Engineering Applications of Artificial Intelligence 25 (2012) 295–308

Contents lists available at SciVerse ScienceDirect

Engineering Applications of Artificial Intelligence

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

A new ARIMA-based neuro-fuzzy approach and swarm intelligence for time series forecasting

Chunshien Li*, Jhao-Wun Hu

Laboratory of Intelligent Systems and Applications, Department of Information Management, National Central University, Taiwan (R.O.C)

A R T I C L E I N F O

Article history:
Received 11 January 2011
Received in revised form
13 August 2011
Accepted 17 October 2011
Available online 8 November 2011

Keywords:
Time series forecasting
Hybrid learning
Neuro-fuzzy system (NFS)
Particle swarm optimization (PSO)
Recursive least-squares estimator (RLSE)
Auto-regressive integrated moving average model (ARIMA)

A B S T R A C T

Time series forecasting is an important and widely interesting topic in the research of system modeling. We propose a new computational intelligence approach to the problem of time series forecasting, using a neuro-fuzzy system (NFS) with auto-regressive integrated moving average (ARIMA) models and a novel hybrid learning method. The proposed intelligent system is denoted as the NFS–ARIMA model, which is used as an adaptive nonlinear predictor to the forecasting problem. For the NFS–ARIMA, the focus is on the design of fuzzy If-Then rules, where ARIMA models are embedded in the consequent parts of If-Then rules. For the hybrid learning method, the well-known particle swarm optimization (PSO) algorithm and the recursive least-squares estimator (RLSE) are combined together in a hybrid way so that they can update the free parameters of NFS–ARIMA efficiently. The PSO is used to update the If-part parameters of the proposed predictor, and the RLSE is used to adapt the Then-part parameters. With the hybrid PSO–RLSE learning method, the NFS–ARIMA predictor may converge in fast learning pace with admirable performance. Three examples are used to test the proposed approach for forecasting ability. The results by the proposed approach are compared to other approaches. The performance comparison shows that the proposed approach performs appreciably better than the compared approaches. Through the experimental results, the proposed approach has shown excellent prediction performance.

© 2011 Elsevier Ltd. All rights reserved.

1. Introduction

Using an intelligent modeling approach, a model can be set up for a target system, with which the relationship of input–output behavior can be approximated. This is known as system modeling. For function approximation, optimization, and forecasting, system modeling has been widely investigated for years. Time series forecasting is one of the momentous applications in system modeling. In a time series, time is usually a very important factor to make decision or prediction. Data in past history recorded in time sequence is called a 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 useful decision. For a more refined definition, time series is a group of statistics, according to the order which events are occurred in time sequence. For instances, 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 change 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.

In the past four decades, various approaches have been presented for time series forecasting (Kim et al., 1997; Xiong, 2001; Box and Jenkins, 1976; Chen et al., 1991; Jang, 1993; Cho and Wang, 1996; Kasabov et al., 1997; Kim and Kasabov, 1999; Nauck and Kruse, 1999; Paul and Kumar, 2002; Sfetsos and Siriopoulos, 2004; Chen et al., 2005; Gao and Er, 2005; Chen et al., 2006; Rong et al., 2006; Herrera et al., 2007; Rojas et al., 2008; Sugiarto and Natarajan, 2007; Zounemat-Kermani and Teshnehlab, 2008; Deng and Wang, 2009; Graves and Pedrycz, 2009; Zhao and Yang, 2009), including powerful auto-regressive moving average (ARMA), fuzzy-based paradigm, neural-net computing model, neural fuzzy hybrid system, and others. In the recent twenty years, fuzzy system is one of the most frequently used methods (Kim et al., 1997; Xiong, 2001; Jang, 1993; Cho and Wang, 1996; Kim and Kasabov, 1999; Nauck and Kruse, 1999; Paul and Kumar, 2002; Rong et al., 2006; Herrera et al., 2007; Sugiarto and Natarajan, 2007; Zounemat-Kermani and Teshnehlab, 2008).

* Corresponding author.
E-mail address: jamesli@mgt.ncu.edu.tw (C. Li).

0952-1976/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2011.10.005

Many kinds of optimization algorithms have been used for fuzzy systems. The design of fuzzy systems to forecasting has been also proposed in related researches. For instance, Takagi and Sugeno (T–S) fuzzy model with fuzzy *C*-regression clustering model was used to determine number of fuzzy rules and gradient descent algorithm was applied for tuning the free parameters of T–S fuzzy model (Kim et al., 1997). A fuzzy model with genetic-based premise learning was proposed by Xiong (2001), for the well-known gas furnace time series dataset of Box and Jenkins. Hybrid neural fuzzy inference system (HyFIS) was developed by Kim and Kasabov (1999), for building and optimizing the fuzzy model. Adaptive-network-based fuzzy inference system (ANFIS) was proposed, where the backpropagation (BP) method and the recursive least-squares estimator (RLSE) were combined to adapt free parameters (Jang, 1993). Almost all of these studies used general T–S fuzzy model (Takagi and Sugeno, 1985) for system modeling and time series forecasting. Besides, neural networks are another major approach to time series forecasting in artificial intelligence field (Chen et al., 1991; Kasabov et al., 1997; Chen et al., 2005, 2006; Deng and Wang, 2009). Most of these studies focused on designing new structure of neural network and applying different learning methods to improve the accuracy of prediction. For example, Chen et al. (2006) presented local linear wavelet neural network (LLWNN). A hybrid training algorithm of particle swarm optimization (PSO) with both diversity learning and gradient descent method was introduced for the training of the LLWNN. Deng and Wang (2009) proposed a novel incremental learning approach (ILA) based on a hybrid fuzzy neural net framework.

However, most of these researches used stationary time series to verify the forecasting performance of their approaches or models. Basically, a time series can be viewed as a set of observations from a stochastic process. A time series is considered stationary if the stochastic process is with the property that probability distribution does not change with time shift, so that the statistics such as mean, variance, and covariance of the time series are of finite constant. In contrast, a non-stationary time series whose probability distribution is changing with time. Thus, the mean, variance and covariance vary with time. This class of time series is usually hard to find the regularity for a prediction model. Traditional mathematical approaches to non-stationary time series are usually feeble and hard to attack such problems with satisfactory performance. Practically, time series in real world usually are non-stationary, especially in economic and finance field. The fluctuations and changes of these non-stationary time series are quite enormous, such as daily closing stock price. Accordingly, the prediction by forecasting models may deviate from the future value easily. For non-stationary time series, the auto-regressive integrated moving average (ARIMA) model, first proposed by Box and Jenkins, has been used in order to make forecasting more accurate. For ARIMA process, models that describe homogeneous non-stationary behavior can be obtained by supposing some suitable difference of the process so that the differenced version of the time series may become stationary (Box and Jenkins, 1976). An ARIMA model combines three different processes including an auto-regressive (AR) process, integration part, and a moving average (MA) process. In previous researches, the concept of ARMA model was used in the fuzzy inference system or neural network. Gao and Er (2005) proposed a ARMA-like model using fuzzy neural network method , called the nonlinear autoregressive moving average with exogenous inputs (NARMAX). They used TSK-type weight vector to mimic ARMA model with exogenous inputs, and cast the NARMAX model into a fuzzy neural network (FNN). Basically, the NARMAX approach is based on the so-called G-FNN framework, whose functionality is equivalent to a TSK-type fuzzy inference system. A hybrid methodology combining an artificial neural network (ANN) and ARMA models was proposed (Rojas et al., 2008) to investigate time series forecasting. However, the forecasting performance and accuracy were not good enough in these previous works. In our study, we propose an innovative hybrid computing paradigm, integrating both neuro-fuzzy system (NFS) and ARIMAs together to achieve ARIMA-based neuro-fuzzy nonlinear-mapping ability for accurate forecasting. Our approach expands ARIMA from its linear modeling ability to the horizon of nonlinear modeling by neuro-fuzzy method, so that excellent performance for accurate forecasting can be achieved. With the neuro-fuzzy system (NFS) theory, ARIMA models are embedded into fuzzy If-Then rules to construct a new NFS–ARIMA predictor to the problem of time series forecasting, equipped with a novel hybrid learning ability. NFS has been a useful modeling tool to represent and process linguistic information and to deal with uncertainty and imprecision (Jang et al., 1997). In fuzzy theory, fuzzy sets can be used to reflect human concepts and thoughts, which tend to be incomplete and vague (Zadeh, 1965, 1975; Fukami et al., 1980; Klir and Yuan, 1995). Besides the viewpoint of learning and adaption, there are perspectives for the use of neuro-fuzzy framework. Basically, neural networks (NNs) and fuzzy systems (FSs) have been proven universal approximators, which can theoretically approximate any function to any degree of accuracy on a compact set. Neuro-fuzzy systems have been successfully applied in various applications. The integration of them tends to adopt their merits to become a unified framework of computing model for information processing, where NN can provide capability for flexible adaptive low-level structure for learning from examples and pattern recognition, and FS can provide high-level comprehensive rule inference for imprecise information processing and decision-making. In this paper, the basic idea of combining fuzzy system and neural network is to design a neuro-fuzzy framework that uses a fuzzy inference system to represent knowledge in an interpretable manner, embedded in the adaptive distributed structure of a neural network. Although the PSO method does not need the support of NN, we think this neuro-fuzzy integration can augment the value of fuzzy system in the perspective of neural network. They complement to one another. Through the connectionist structure by NN, FS can be transformed into a neuro-fuzzy system, providing an insight to the formation of knowledge-base structure and possible design types for FS in the view point of neural network. Therefore, we adopt the participation of NN that can positively provide adaptive flexibility and unified model framework to the proposed approach.

In this paper, a novel intelligent approach is presented for time series forecasting, using Takagi–Sugeno (T–S) neuro-fuzzy model (Jang et al., 1997; Sugeno and Kang, 1988) and ARIMAs. The implementation of a T–S neuro-fuzzy model is very similar to that of a fuzzy logic inference system, except the consequents are described by functions of crisp inputs. 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 an NFS is significantly critical to forecasting. The proposed neuro-fuzzy approach is denoted as NFS–ARIMA, where ARIMA models are embedded in the fuzzy rules. For different time series, different order of NFS–ARIMA can be conducted to make forecasting as good as possible. Furthermore, because there are usually many unknown parameters in the NFS–ARIMA, the selection of learning algorithm plays an extremely pivotal position. In order to adapt the free parameters, a hybrid learning algorithm is proposed in this paper, using both the particle swarm optimization (PSO) (Kennedy and Eberhart, 1995) and the recursive least-squares estimator (RLSE) (Jang, 1993; Jang et al., 1997). The PSO is used for the update of the premise parameters of the NFS–ARIMA; the RLSE for the consequence parameters; they work together in a hybrid way. The focus is to find the optimal or near-optimal solution for the NFS–ARIMA, so that the forecasting performance can be as accurate as possible. The PSO is a swarm based heuristic method, which is useful for global optimization problems, and the RLSE can estimate a linear model optimal in a very efficient way. With the hybrid learning method, the NFS–ARIMA is expected to have accurate forecasting.

In Section 2, we present the theory of NFS–ARIMA. In Section 3, we describe the hybrid learning method, combining the PSO and the RLSE for the proposed NFS–ARIMA. In Section 4, three examples are used to test the proposed approach, including the Mackey-Glass chaos time series, the star brightness time series and the daily IBM stock closing price time series. Finally, the paper is concluded in Section 5 for the proposed approach and experimental results.

2. ARIMA based neuro-fuzzy system

To specify the proposed multiple-input–single-output ARIMA based neuro-fuzzy system, we begin with the first-order T–S fuzzy model. To generate fuzzy rules from a given input–output data set, the first T–S fuzzy model was proposed by Takagi and Sugeno (1985) to develop a systematic approach, where the consequents of fuzzy rules are functions of crisp inputs. Let us consider an M-input–one-output fuzzy system with K fuzzy rules in the rule base. The form of first order T–S fuzzy rules can be given as follows:

$$\text{Rule } i: \text{IF } x_1 \text{ is } {}^{i}s_1(h_1) \text{ and } x_2 \text{ is } {}^{i}s_2(h_2)\ldots\text{and } x_M \text{ is } {}^{i}s_M(h_M) \text{ Then } {}^{i}y={}^{i}a_0+{}^{i}a_1h_1+\cdots+{}^{i}a_Mh_M \tag{1}$$

for $i=1,2,\ldots,K$, where $\{x_j, j=1,2,\ldots,M\}$ are input linguistic variables and$\{h_j, j=1,2,\ldots,M\}$ are their corresponding base variables (or called numerical variables), which can be obtained from observations of a time series; ${}^{i}y$ is the output variable of the ith fuzzy rule; $\{{}^{i}s_1,{}^{i}s_2,\ldots{}^{i}s_M\}$ are the fuzzy sets of the ith rule; $\{{}^{i}a_1,{}^{i}a_2,\ldots{}^{i}a_M\}$ are the consequent parameters of the ith rule. The fuzzy model can be cast into neural structure to be a neuro-fuzzy system. The proposed approach is based on the NFS methodology. In the study, input data are from the observations of a time series.

In time series forecasting, the approach of autoregressive moving average (ARMA) model, sometimes called Box–Jenkins model, is the most used approach to predict future value. For an ARMA model, the observation of a variable in interest for a time series can be expressed as a linear combination of past observations and shocks. Basically, an ARMA process consists of two parts, an auto-regressive (AR) process and a moving average (MA) process. Based on an AR process of pth order, the observation $X(t)$ for a time series can be expressed as follows:

$$X(t)=a_0+a_1X(t-1)+a_2X(t-2)+\cdots+a_pX(t-p)+e(t) \tag{2}$$

where t is the time index, a_0 is a constant term, $\{a_1,a_2,\ldots,a_p\}$ are the process parameters, and $e(t)$ is a random shock at time t, which is assumed to be independently and identically distributed (iid) with zero mean and a variance of σ^2. For a MA process of qth order, the observation $X(t)$ is given below

$$X(t)=u+e(t)-\theta_1e(t-1)-\theta_2e(t-2)-\cdots-\theta_qe(t-q) \tag{3}$$

where u is the expectation of $X(t)$, $\{\theta_1,\theta_2,\ldots,\theta_q\}$ are the process parameters and $\{e(t),e(t-1),\ldots,e(t-q)\}$ are random shocks with the iid property. For an ARMA model, the observation $X(t)$ can be expressed as follows:

$$\begin{aligned}X(t)&=a_0+a_1X(t-1)+a_2X(t-2)+\cdots+a_pX(t-p)+e(t)\\&\quad-\theta_1e(t-1)-\cdots-\theta_qe(t-q)\\&=a_0+\sum_{i=1}^{p}a_iX(t-i)-\sum_{j=1}^{q}\theta_je(t-j)+e(t)\end{aligned} \tag{4}$$

For nonstationary time series, auto-regressive integrated moving average (ARIMA) models are the most common class to tranquilize time series, using the differencing transformation. The first ARIMA was presented by Box and Jenkins (1976). The ARIMA and its variants have been used prosperously in many areas of time series forecasting. Let us define the operations below

$$B^nX(t)\equiv X(t-n) \tag{5}$$

$$\nabla\equiv(1-B) \tag{6}$$

where B is called the backward shift operator and ∇ is called the difference operator. Based on (5) and (6), we further make the following definition:

$$\psi(t)\equiv\nabla^dX(t)=(1-B)^dX(t) \tag{7}$$

where d is called the difference order, which is a non-negative integer. Based on (4), the general form of an ARIMA process is expressed below

$$\psi(t)=a_0+\sum_{i=1}^{p}a_i\psi(t-i)-\sum_{j=1}^{q}\theta_je(t-j)+e(t) \tag{8}$$

Based on (8), an ARIMA predictor can be setup. Assume the random shocks $\{e(t-1),e(t-2),\ldots,e(t-q)\}$ be treated as forecast errors in the forecasting process. The general form of the ARIMA prediction model can be expressed below

$$\begin{aligned}\hat{\psi}(t)&=a_0+\sum_{i=1}^{p}a_i\psi(t-i)-\sum_{j=1}^{q}\theta_je(t-j)\\&=a_0+\sum_{i=1}^{p}a_iB^i\psi(t)-\sum_{j=1}^{q}\theta_jB^je(t)\\&\equiv\text{ARIMA}(p,d,q)\end{aligned} \tag{9}$$

where $\hat{\psi}(t)$ is the forecast for $\psi(t)$, ARIMA(p,d,q) indicates the order of the prediction model, and $\{a_i, i=0,1,2,\ldots,p\}$ and $\{\theta_j, j=1,2,\ldots,q\}$ are the model parameters. Note that if $d=0$, ARIMA(p,0,q) degenerates to ARMA(p,q). Similarly, ARIMA(p,0,0) degenerates to AR(p), and ARIMA(0,0,q) to MA(q). The forecast for $X(t)$ is denoted as $\hat{X}(t)$. The forecast error is defined below

$$e(t)\equiv X(t)-\hat{X}(t) \tag{10}$$

Based on (7), the relationship between $\hat{X}(t)$ and $\hat{\psi}(t)$ is expressed as

$$\hat{\psi}(t)=\nabla^d\hat{X}(t)=(1-B)^d\hat{X}(t) \tag{11a}$$

For the case of $d=0$, $\hat{\psi}(t)=\hat{X}(t)$. For the difference order $d=1$, we have $\hat{\psi}(t)=\nabla\hat{X}(t)=(1-B)\hat{X}(t)=\hat{X}(t)-\hat{X}(t-1)$. Because the observation $X(t-1)$ is already known at time t, it can be used to replace $\hat{X}(t-1)$. Thus, the expression for the forecast $\hat{X}(t)$ can be given as follows:

$$\hat{X}(t)=\hat{\psi}(t)+X(t-1) \tag{11b}$$

For $d=2$, the forecast $\hat{X}(t)$ can be expressed as follows:

$$\hat{X}(t)=\hat{\psi}(t)+X(t-1)+\nabla X(t-1) \tag{11c}$$

For $d=3$, the forecast $\hat{X}(t)$ can be given below

$$\hat{X}(t)=\hat{\psi}(t)+X(t-1)+2\nabla X(t-1)-\nabla X(t-2) \tag{11d}$$

For higher difference order ($d\geq4$), based on (11a), similar operation can be applied to obtain the forecast $\hat{X}(t)$.

The consequent in (1) can be viewed as AR(p) model, where $p=M$. With the consideration of time, the form of consequent part can be extended with ARMA(p,q) model, given as follows:

$$ {}^{i}y(t) = {}^{i}a_0 + {}^{i}a_1 h_1(t) + \cdots + {}^{i}a_M h_M(t) - {}^{i}\theta_1 e(t-1) - \cdots - {}^{i}\theta_q e(t-q) \tag{12} $$

Furthermore, the ARIMA(p,d,q) in (9) can be embedded to T–S fuzzy rules whose consequents are expressed in the following form:

$$ {}^{i}y(t) = {}^{i}\mathrm{ARIMA}(p,d,q) = {}^{i}a_0 + \sum_{j=1}^{p} {}^{i}a_j B^j \psi(t) - \sum_{k=1}^{q} {}^{i}\theta_k B^k e(t) \tag{13} $$

for $i=1,2,\ldots,K$, where $\{{}^{i}a_j, j=0,1,\ldots p\}$ and $\{{}^{i}\theta_k, k=1,2,\ldots,q\}$ are the consequent parameters of the ith fuzz rule; $\{e(t-k), k=1,2,\ldots q\}$ are prediction errors; $\{\psi(t-j), j=1,2,\ldots,p\}$ are the ^{i}ARIMA (p,d,q) state.

For example, if an ARIMA(2,1,2) model is used, the form of ${}^{i}y(t)$ for the ith fuzzy rule can be expressed as follows:

$$ \begin{aligned} {}^{i}y(t) &= {}^{i}a_0 + {}^{i}a_1 \overbrace{(X(t-1)-X(t-2))}^{\psi(t-1)} + {}^{i}a_2 \overbrace{(X(t-2)-X(t-3))}^{\psi(t-2)} \\ &\quad - {}^{i}\theta_1 e(t-1) - {}^{i}\theta_2 e(t-2) \\ &= {}^{i}\mathrm{ARIMA}(2,1,2) \end{aligned} \tag{14} $$

For convenience, the proposed predictor is denoted by **NFS–ARIMA(p,d,q)**, where (p,d,q) indicates the model order. Based on (13), the general form of If-Then rules for the NFS–ARIMA(p,d,q) is written as follows:

$$ \begin{aligned} &\text{Rule } i: \text{IF } x_1 \text{ is } {}^{i}s_1(h_1(t)) \text{ and } x_2 \text{ is } {}^{i}s_2(h_2(t))\ldots\text{and } x_M \text{ is} \\ &\quad {}^{i}s_M(h_M(t)) \text{ Then } {}^{i}y(t) = {}^{i}\mathrm{ARIMA}(p,d,q) \end{aligned} \tag{15} $$

for $i=1,2,\ldots,K$. The fuzzy inference is cast into neural structure to be an ARIMA based neuro-fuzzy system, shown in Fig. 1, which is a six-layer 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 a crisp variable of the input vector $H(t)=[h_1(t), h_2(t),\ldots,h_M(t)]$.

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. For the design of fuzzy sets, we select Gaussian membership functions, which have the following advantages. The class of Gaussian membership functions (MFs) only needs less number of parameters than other class of MFs, such as triangular MF (using three parameters) and trapezoidal MF (using four parameters). A Gaussian MF may only need the parameters of mean and spread to define its function. Gaussian MFs have smooth and continuously differentiable property, providing good transition behavior from full membership to zero membership for the definition of fuzzy set. And, Gaussian MFs are with very long tails theoretically, which are good for the design of proposed models, providing excellent completeness for input-space partition and good overlap of fuzzy sets. In general, the properties of completeness and fuzzy-set overlap are very important to NFS-based applications. The general form of Gaussian membership function is given as follows:

$$ \mathrm{gaussmf}(h;m,\sigma) = \exp(-0.5((h-m)/\sigma)^2) \tag{16} $$

where h is a input base variable; m and σ are the parameters of mean and spread, respectively. The parameters of mean and spread for all the fuzzy sets to define the premises of the proposed NFS–ARIMA are called the premise parameters.

Layer 2: the layer is for the firing strengths of the K fuzzy rules. The nodes perform the *fuzzy-and* operations for the premise parts of the fuzzy rules. Each node output indicates a firing strength for its corresponding fuzzy rule. The firing strength for the ith rule,

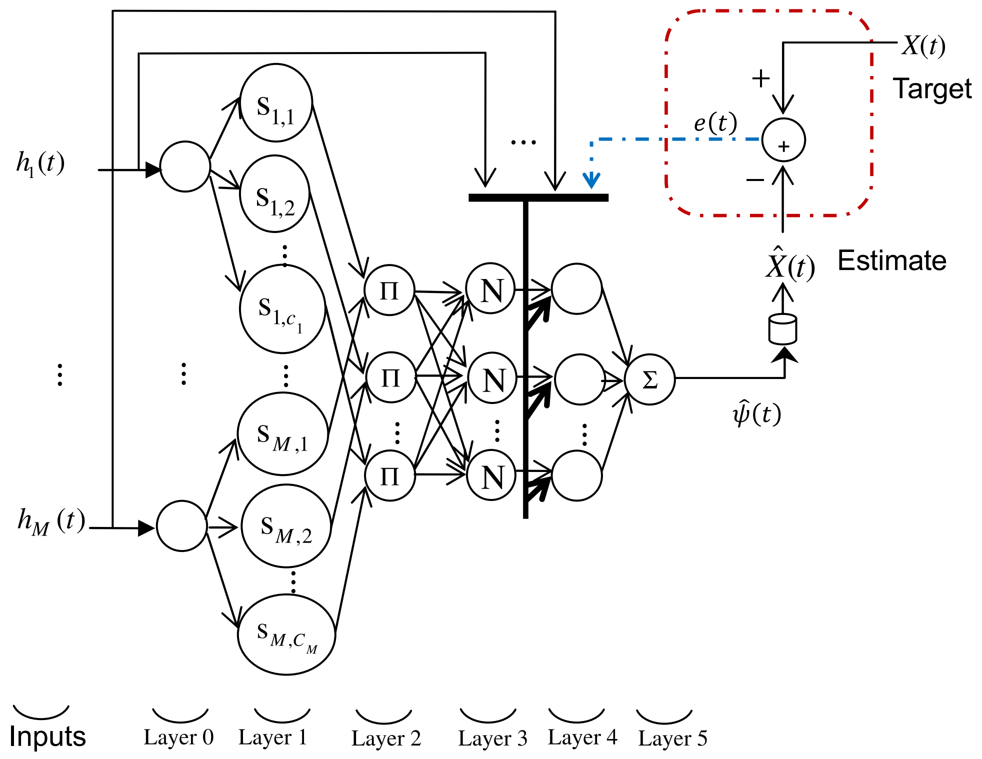

Fig. 1. ARIMA-based neuro-fuzzy system, where $s_{j,q}$ is the qth fuzzy set of the jth input for$j=1,2,\ldots,M$ and $q=1,2,\ldots,c_j$; c_j indicates the cardinality of the term set of the jth input. The dotted portion is applied if prediction errors are involved in ARIMAs in the consequents of fuzzy rules. Note that as shown in (1) and (15), ${}^{i}s_j$ the jth fussy set of the ith fuzzy rule is from one of the term set $\{s_{j,q}, q=1,2,\ldots,c_j\}$ for the jth input. The antecedents of fuzzy rules begin to form in Layer 2.

denoted as ${}^{i}\beta$, is given below

$$ {}^{i}\beta(t) = \prod_{j=1}^{M} {}^{i}\mu_j(h_j(t)) \tag{17}$$

where ${}^{i}\mu_j(h_j(t))$ indicates the membership degree for the jth fuzzy set of the ith rule.

Layer 3: the normalization of the firing strengths of the fuzzy rules is performed in this layer. The normalized firing strength of the ith fuzzy rule is given as follows:

$$ {}^{i}\gamma(t) = \frac{{}^{i}\beta(t)}{\sum_{i=1}^{K} {}^{i}\beta(t)} \tag{18}$$

Layer 4: the layer is called the consequent layer. The nodes in the layer perform the normalized consequents of all the fuzzy rules. With (13), the actual output by the ith fuzzy rule is given as follows:

$$ {}^{i}\gamma(t){}^{i}y(t) = {}^{i}\gamma(t)({}^{i}\text{ARIMA}(p,d,q)) = {}^{i}\gamma(t)\left({}^{i}a_0 + \sum_{j=1}^{p} {}^{i}a_j B^j \psi(t) + \sum_{k=1}^{q} {}^{i}\theta_k B^k e(t)\right) \tag{19}$$

where $\{{}^{i}a_j, j=0,1,2,\ldots,p\}$ and $\{{}^{i}\theta_k, k=1,2,\ldots,q\}$ are the consequent parameters of the ith fuzzy rule; B is the backward shift operator.

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 inferred result. The inferred result is denoted by $\hat{\psi}(t)$, given as follows:

$$ \hat{\psi}(t) = \sum_{i=1}^{K} {}^{i}\gamma(t){}^{i}y(t) = \frac{\sum_{i=1}^{K} {}^{i}\beta(t){}^{i}y(t)}{\sum_{i=1}^{K} {}^{i}\beta(t)} \tag{20}$$

Once the consequent parameters are determined, the inferred result by the NFS–ARIMA can be expressed as follows:

$$ \hat{\psi}(t) = \sum_{i=1}^{K} {}^{i}\gamma(t)\left({}^{i}a_0 + \sum_{j=1}^{p} {}^{i}a_j B^j \psi(t) + \sum_{k=1}^{q} {}^{i}\theta_k B^k e(t)\right) \tag{21}$$

for $t=1,2,\ldots,N$. With (11a), the relationship between the inferred result $\hat{\psi}(t)$ and the forecast $\hat{X}(t)$ for the time series is established. The expression of $\hat{X}(t)$ for the cases of $d=1$, $d=2$, and $d=3$ can be obtained by (11b), (11c), and (11d), respectively.

Assume that there are observations for a time series, which can be collected to be used as training data for the proposed neuro-fuzzy predictor. The training data (TD) is given as follows:

$$ \text{TD} = \{X(i), i=1,2,\ldots\} \tag{22}$$

To overview the prediction approach to the problem of time series forecasting, an illustrative diagram is given in Fig. 2.

The input vector $H(t)$ is from the TD in (22). The NFS–ARIMA predictor generates an inferred result, which is then transformed to forecast value $\hat{X}(t)$, using (11a). A prediction error in (10) is produced. Assume there are N prediction errors $\{e(t), t=1,2,\ldots,N\}$ to involve in the training process, where these errors are used to design a cost function in terms of root mean squared error (RMSE) for training purpose. The cost function is to be minimized so that the proposed NFS–ARIMA predictor can be optimized. The definition of RMSE is given as follows:

$$ \text{RMSE} = \sqrt{\frac{\sum_{t=1}^{N} e(t)^2}{N}} \tag{23a}$$

The RMSE is used as a performance measure. Note that mean squared error (MSE), defined as $\text{MSE}=\text{RMSE}^2$, can also be used for performance measure. Based on RMSE in (23a), the non-dimensional error index (NDEI) can be defined as follows:

$$ \text{NDEI} = \frac{\text{RMSE}}{\sigma_X} \tag{23b}$$

where σ_X is the standard deviation of the target time series. The premise parameters in Layer 1 and the consequent parameters in Layer 4 can be viewed as the two subsets of the free parameters. These parameters are called the system parameters of the NFS–ARIMA. They can be adapted by machine learning algorithms such as PSO and RLSE. In the following section, we specify the PSO–RLSE hybrid learning method for the proposed NFS–ARIMA.

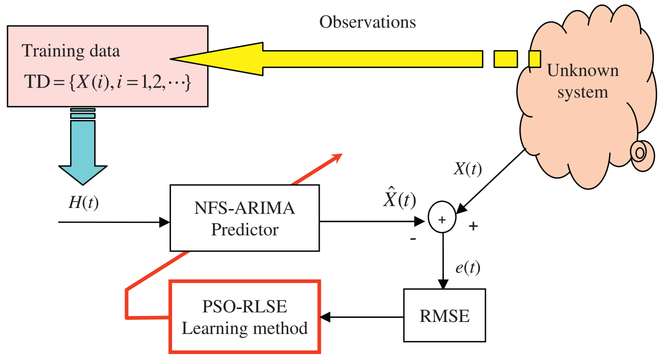

Fig. 2. ARIMA based neuro-fuzzy prediction approach.

3. Hybrid PSO–RLSE learning method

3.1. Particle swarm optimization(PSO)

The original PSO was first proposed by Kennedy and Eberhart (1995). The method of PSO is an excellent approach, having collective wisdom concept to the evolution of optimization search. The PSO and its variants have been applied successfully to a variety of application problems (Shi and Eberhart, 1999; Parsopoulos and Vrahatis, 2002a, 2002b; Juang, 2004; Coello et al., 2004; Shi and Eberhart, 1998). For the PSO algorithm, a particle is viewed as a bird, searching for food (a better solution), and all the particles comprise a population, called a “swarm” of the birds. Each particle moves toward its own best position and the swarm-best position. All particles compete with each other to become the swarm best. By this way, PSO combines the 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. Iteratively, particles search for the optimal solution, using the concept of fitness function which is designed with RMSE. A fitness function sometimes is called a cost function. Particles change their search directions by means of two search memories, which are **Pbest** (the best location of individual particle) and **Gbest** (the best location of the swarm). A vector illustration for a PSO particle is shown in Fig. 3(a), and an illustration of position evolution in the search process by a PSO swarm is shown conceptually in Fig. 3(b), where the target position denotes the location of the optimal solution. For simplicity, there are only two particles in the swarm, 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 particle B is the **Gbest** of the swarm at $t=2$. Assume the problem space is with Q dimensions. The particles' velocities are updated as follows:

$$ \mathbf{V}_i(t+1) = w \times \mathbf{V}_i(t) + c_1 \times \xi_1 \times (\mathbf{Pbest}_i(t) - \mathbf{L}_i(t)) + c_2 \times \xi_2 \times (\mathbf{Gbest}(t) - \mathbf{L}_i(t)) \tag{24}$$

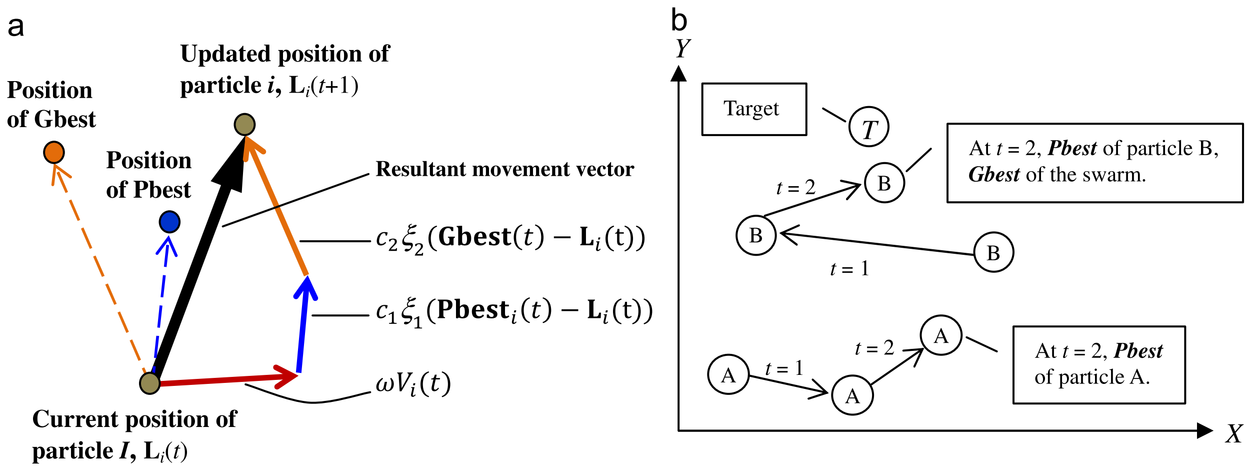
a
Position of Gbest
Updated position of particle i, L$_i$(t+1)
Position of Pbest
Resultant movement vector
$c_2\xi_2(\mathbf{Gbest}(t) - \mathbf{L}_i(\mathrm{t}))$
$c_1\xi_1(\mathbf{Pbest}_i(t) - \mathbf{L}_i(\mathrm{t}))$
$\omega V_i(t)$
Current position of particle I, L$_i$(t)
b
Y
Target
T
At t = 2, Pbest of particle B, Gbest of the swarm.
t = 2
B
B
t = 1
B
A
t = 1
t = 2
A
A
At t = 2, Pbest of particle A.
X

Fig. 3. Illustration of PSO. (a) Viewpoint in vectors for particle *i* of PSO. (b) Position evolution for PSO particles.

Step 1. Initial settings for swarm size, dimensions of particles, maximal search iterations, and stopping criteria.
Step 2. Initialize positions and velocities for the particles of the swarm.
Step 3. Calculate fitness for each particle.
Step 4. Update velocity and position for each particle.
Step 5. If $f(L_i) < f(\boldsymbol{Pbest}_i)$, then update the ***Pbest****ᵢ*. If $f(\boldsymbol{Pbest}_i) < f(\boldsymbol{Gbest})$, update the ***Gbest*** of the swarm.
Step 6. If any stopping criterion is satisfied, ***Gbest*** is the optimal solution. Otherwise, go back to **Step 3** to continue the PSO procedure.

Fig. 4. Implementation procedure of PSO.

where $\boldsymbol{V}_i(t)=[v_{i,1}(t),v_{i,2}(t),\ldots,v_{i,Q}(t)]$ is the velocity of the ith particle at time t, $\{c_1,c_2\}$ are the parameters for PSO, $\{\xi_1, \xi_2\}$ are random numbers in [0,1], and w is the inertia weight (Shi and Eberhart, 1998). The particles locations are updated as follows:

$$\mathbf{L}_i(t+1)=\mathbf{L}_i(t)+\mathbf{V}_i(t+1) \tag{25}$$

where $\boldsymbol{L}_i(t)=[l_{i,1}(t),l_{i,2}(t),\ldots,l_{i,Q}(t)]$ is the location of the ith particle at time t.

The implementation procedure of PSO is given in Fig. 4, where $f(\boldsymbol{L}_i)$ indicates the cost in RMSE for the current location of the ith particle; $f(\mathbf{Pbest}_i)$ indicates the cost for $\mathbf{Pbest}_i$; $f(\mathbf{Gbest})$ represents the cost for **Gbest**.

3.2. *Recursive least-squares estimator (RLSE)*

In general, the least squares estimation (LSE) problem (Jang et al., 1997) can be specified with a linear model given as follows:

$$y=f_1(u)\theta_1+f_2(u)\theta_2+\cdots+f_m(u)\theta_m+\varepsilon \tag{26}$$

where y is the target; u is the model's input; $\{f_i(.), i=1,2,\ldots,m\}$ are known functions of u; $\{\theta_i, i=1,2,\ldots,m\}$ are unknown parameters to be estimated; ε indicates model error. Substituting training data pairs $\{(u_i,y_i),i=1,2,\ldots,N\}$ into (26), we have a set of N linear equations, given as follows:

$$\begin{aligned}
y_1 &= f_1(u_1)\theta_1+f_2(u_1)\theta_2+\cdots+f_m(u_1)\theta_m+\varepsilon_1\\
y_2 &= f_1(u_2)\theta_1+f_2(u_2)\theta_2+\cdots+f_m(u_2)\theta_m+\varepsilon_2\\
&\vdots\qquad\quad \vdots\qquad\qquad\quad \vdots\qquad \vdots\\
y_N &= f_1(u_N)\theta_1+f_2(u_N)\theta_2+\cdots+f_m(u_N)\theta_m+\varepsilon_N
\end{aligned} \tag{27}$$

The LSE problem can be written in a concise matrix form below

$$\mathbf{Y}=\mathbf{A}\boldsymbol{\theta}+\boldsymbol{\varepsilon} \tag{28a}$$

where

$$\mathbf{A}=\begin{bmatrix} f_1(u_1) & f_2(u_1) & \cdots & f_m(u_1)\\ f_1(u_2) & f_2(u_2) & \cdots & f_m(u_2)\\ \vdots & \vdots & \cdots & \vdots\\ f_1(u_N) & f_2(u_N) & & f_m(u_N)\end{bmatrix} \tag{28b}$$

$$\boldsymbol{\theta}=[\theta_1,\theta_2,\ldots,\theta_m]^T \tag{28c}$$

$$\mathbf{Y}=[y_1,y_2,\ldots,y_N]^T \tag{28d}$$

$$\boldsymbol{\varepsilon}=\begin{bmatrix}\varepsilon_1 & \varepsilon_2 & \cdots & \varepsilon_N\end{bmatrix}^T \tag{28e}$$

Note that **A** is the input matrix; $\boldsymbol{\theta}$ is the parameter vector to be estimated; **Y** is the target vector; $\boldsymbol{\varepsilon}$ is the error vector.

By the recursive least-squares estimator (RLSE) method (Jang et al., 1997), the optimal solution for $\boldsymbol{\theta}$ can be obtained sequentially and recursively. At iteration k, the transient estimator is denoted as $\boldsymbol{\theta}_k$, which can be calculated recursively below

$$\mathbf{P}_{k+1}=\mathbf{P}_k-\frac{\mathbf{P}_k\mathbf{b}_{k+1}\mathbf{b}_{k+1}^{\mathbf{T}}\mathbf{P}_k}{1+\mathbf{b}_{k+1}^{\mathbf{T}}\mathbf{P}_k\mathbf{b}_{k+1}} \tag{29a}$$

$$\boldsymbol{\theta}_{k+1}=\boldsymbol{\theta}_k+\mathbf{P}_{k+1}\mathbf{b}_{k+1}(y_{k+1}-\mathbf{b}_{k+1}^{\mathbf{T}}\boldsymbol{\theta}_k) \tag{29b}$$

for $k=0,1,2,\ldots,(N-1)$, where $[\mathbf{b}_k^T,y_k]$ is the kth row of $[\mathbf{A}, \mathbf{Y}]$ in (28a). And then, the $\boldsymbol{\theta}_N$ is the estimate of $\boldsymbol{\theta}$ for the consequent parameters of the proposed NFS–ARIMA. Before the RLSE in (29a) and (29b) is started, $\boldsymbol{\theta}_0$ can be initially set to zero and $\mathbf{P}_0$ is usually set below

$$\mathbf{P}_0=\alpha\mathbf{I} \tag{30}$$

where α is a large positive value and **I** is the identity matrix.

3.3. Hybrid PSO–RLSE learning algorithm

A hybrid PSO–RLSE method is devised to train the proposed NFS–ARIMA for the purpose of fast learning, where the PSO is used to update the premise parameters and the RLSE is used to update the consequent parameters. The implementation flow-chart of the PSO–RLSE is shown in Fig. 5.

The procedure in steps for training the NFS–ARIMA is given below.

Step 1. Collect sample data. Some portion of the data is used for training, and the rest is for testing.

Step 2. Update the premise parameters by the PSO.

Step 3. Update the consequent parameters by the RLSE. The vectors $\mathbf{b}_{k+1}$ and $\boldsymbol{\theta}_k$ in the RLSE equations are given below

$$\begin{cases} \mathbf{b}_{k+1} = [{}^1\mathbf{bb}(k+1)\ \ {}^2\mathbf{bb}(k+1)\cdots{}^K\mathbf{bb}(k+1)] \\ {}^i\mathbf{bb}(k+1) = [{}^i\gamma\ \ h_1(k+1){}^i\gamma\cdots h_M(k+1){}^i\gamma] \end{cases} \tag{31}$$

$$\begin{cases} \boldsymbol{\theta}_k = [{}^1\boldsymbol{\omega}(k)\ \ {}^2\boldsymbol{\omega}(k)\cdots{}^K\boldsymbol{\omega}(k)]^T \\ {}^i\boldsymbol{\omega}(k) = [{}^ia_0(k)\ \ {}^ia_1(k)\cdots{}^ia_M(k)]^T \end{cases} \tag{32}$$

for $k=0,1,2,\ldots,(N-1)$ and $i=1,2,\ldots,K$, where ${}^i\gamma$ is the normalized firing strength of the ith fuzzy rule.

Step 4. After the update of all parameters for the NFS–ARIMA, calculate the forecast $\hat{X}(t)$.

Step 5. Calculate fitness in RMSE for each PSO particle.

Step 6. Update **Pbest** for each particle and **Gbest** for the PSO swarm.

Step 7. If any stopping criterion is satisfied, stop the training procedure. And, **Gbest** is used for the optimum premise parameters of the NFS–ARIMA. Otherwise, go back to *Step 2* and continue the procedure.

4. Experimentation

Example 1. Mackey-Glass chaos time series

The well-known Mackey-Glass chaos time series is used to test the proposed approach for performance comparison to other approaches in the literature. The time series is defined as follows:

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{33}$$

where $\tau=17$. 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 prediction form by the proposed NFS–ARIMA predictor for the chaos time series is given below:

$$x(t+\overline{P}) = f(x(t),x(t-\Delta),\ldots,x(t-(\overline{D}-1)\Delta)) \tag{34}$$

where t is the time index, $\overline{P}=\Delta=6$, and $\overline{D}=4$. From the Mackey-Glass time series $x(t)$, the form of data pairs $(H(t), X(t))$ for the training of NFS–ARIMA is given as follows:

$$H(t) = [x(t-18),x(t-12),x(t-6),x(t)] \tag{35}$$

$$X(t) = x(t+6) \tag{36}$$

for $t=118$ to 1117, where $H(t)$ is the input vector to the predictor and $X(t)$ is the corresponding target. There are 1000 data pairs generated. The first 500 data pairs are used for training and the remaining data pairs are used for testing. Two proposed predictors, NFS–ARIMA(4,0,0) and NFS–ARIMA(4,1,0), are designed and tested for the example. Each predictor has four inputs. Each input variable has two fuzzy sets. There are 16 fuzzy rules. For each predictor, there are 16 premise parameters and 80 consequent parameters. All of the fuzzy sets are with the form of Gaussian

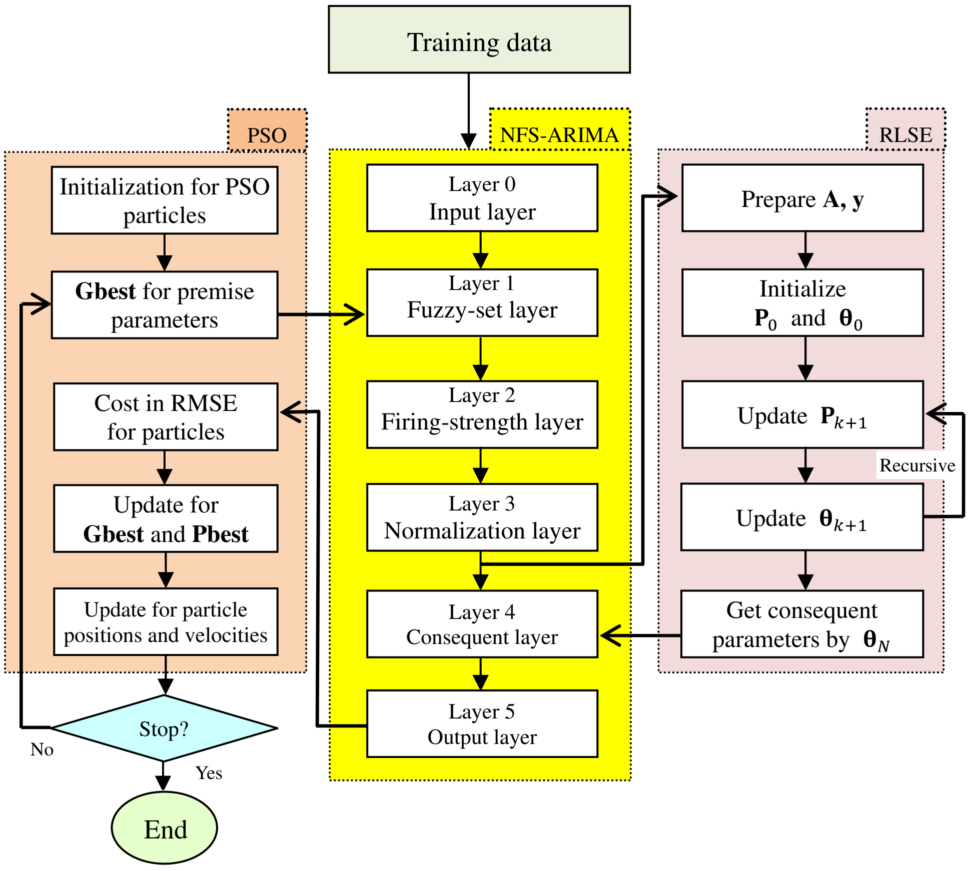
Training data
PSO
NFS-ARIMA
RLSE
Initialization for PSO particles
Gbest for premise parameters
Cost in RMSE for particles
Update for Gbest and Pbest
Update for particle positions and velocities
Stop?
No
Yes
End
Layer 0 Input layer
Layer 1 Fuzzy-set layer
Layer 2 Firing-strength layer
Layer 3 Normalization layer
Layer 4 Consequent layer
Layer 5 Output layer
Prepare A, y
Initialize P_0 and θ_0
Update P_k+1
Recursive
Update θ_k+1
Get consequent parameters by θ_N

Fig. 5. Flowchart of the PSO–RLSE method for training of NFS–ARIMA.

Table 1
Settings of the PSO–RLSE hybrid learning method (Mackey-Glass chaos time series).

PSO	
Dimensions of PSO particles	16
Swarm size	100
Initialization of particle positions	Random in $[0,1]^{16}$
Initialization of particle velocities	Random in $[0,1]^{16}$
Learning rate (c_1,c_2)	(2,2)
Inertia weight w	0.8
Maximum iterations	400
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

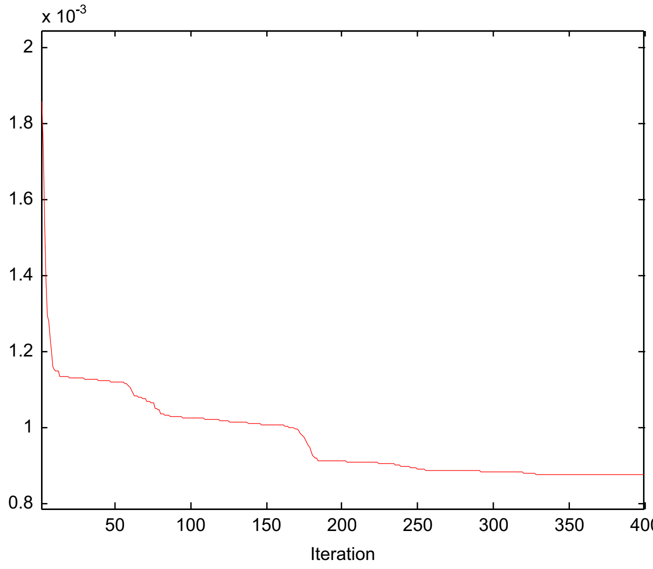

Fig. 6. Learning curve of the proposed NFS–ARIMA(4,1,0) (Mackey-Glass chaos time series).

membership function in (16). For the training of the two NFS–ARIMA predictors, the settings of the PSO–RLSE hybrid method are given in Table 1. After training, both of the NFS–ARIMA predictors performed very well, and the NFS–ARIMA(4,1,0) performed better than the NFS–ARIMA(4,0,0). The prediction performance by the NFS–ARIMA(4,1,0) is 8.7×10^{-4} in RMSE for training phase and 8.6×10^{-4} for testing phase, while the NFS–ARIMA(4,0,0) has the prediction performance 1.4×10^{-3} and 1.3×10^{-3} for training and testing phases, respectively.

The learning curve of NFS–ARIMA(4,1,0) is shown in Fig. 6. The prediction result is shown in Fig. 7(a), where the Mackey-Glass time series is indicated by black line and the forecast is by red dot line. Practically, the proposed approach has very accurate performance, whose prediction response is very close to the real time series. The prediction and real time series curves are almost coincided with each other. For clear display, we enlarge a portion of prediction response in Fig. 7(b), showing they are still close to one another, even in this enlargement scale. After training, the fuzzy sets of the four inputs are shown in Fig. 8.

The performance comparison to other approaches in the literature is given in Table 2, where the proposed approach shows very excellence performance superior to the compared approaches, especially the NFS–ARIMA(4,1,0).

After training, the parameters of NFS–ARIMA(4,1,0) are given in Table 3.

Example 2. Star brightness time series

For 600 successive days, the brightness observations at midnight of a variable star were recorded to be the star brightness time series. The dataset is obtained from Hyndman, denoted as $\{y(i), i=1,2,\ldots,600\}$. The time series was scaled by a factor of 1/30. The first 300 samples are used for training and the remaining 300 samples are used for testing. The data range is normalized into the interval of 0 and 1. The input–output data pairs $\{H(t), X(t)\}$ for the predicting model are selected as follows:

$$H(t)=[y(t-1),y(t-2),y(t-3)] \tag{37}$$

$$X(t)=y(t) \tag{38}$$

Two predictors NFS–ARIMA(3,0,0) and NFS–ARIMA(3,2,2) are used in the example. Each NFS–ARIMA predictor is with three

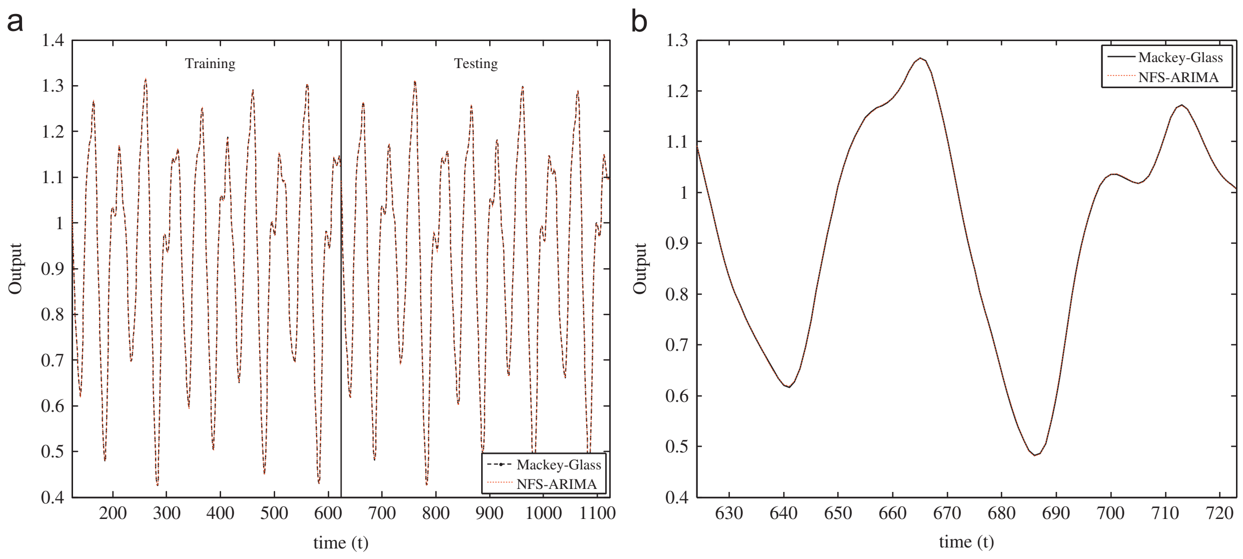

Fig. 7. (a) Prediction response by NFS–ARIMA(4,1,0) for the Mackey-Glass chaos time series. (b) Enlargement for a portion of the prediction response, showing that the two curves are close to each other.

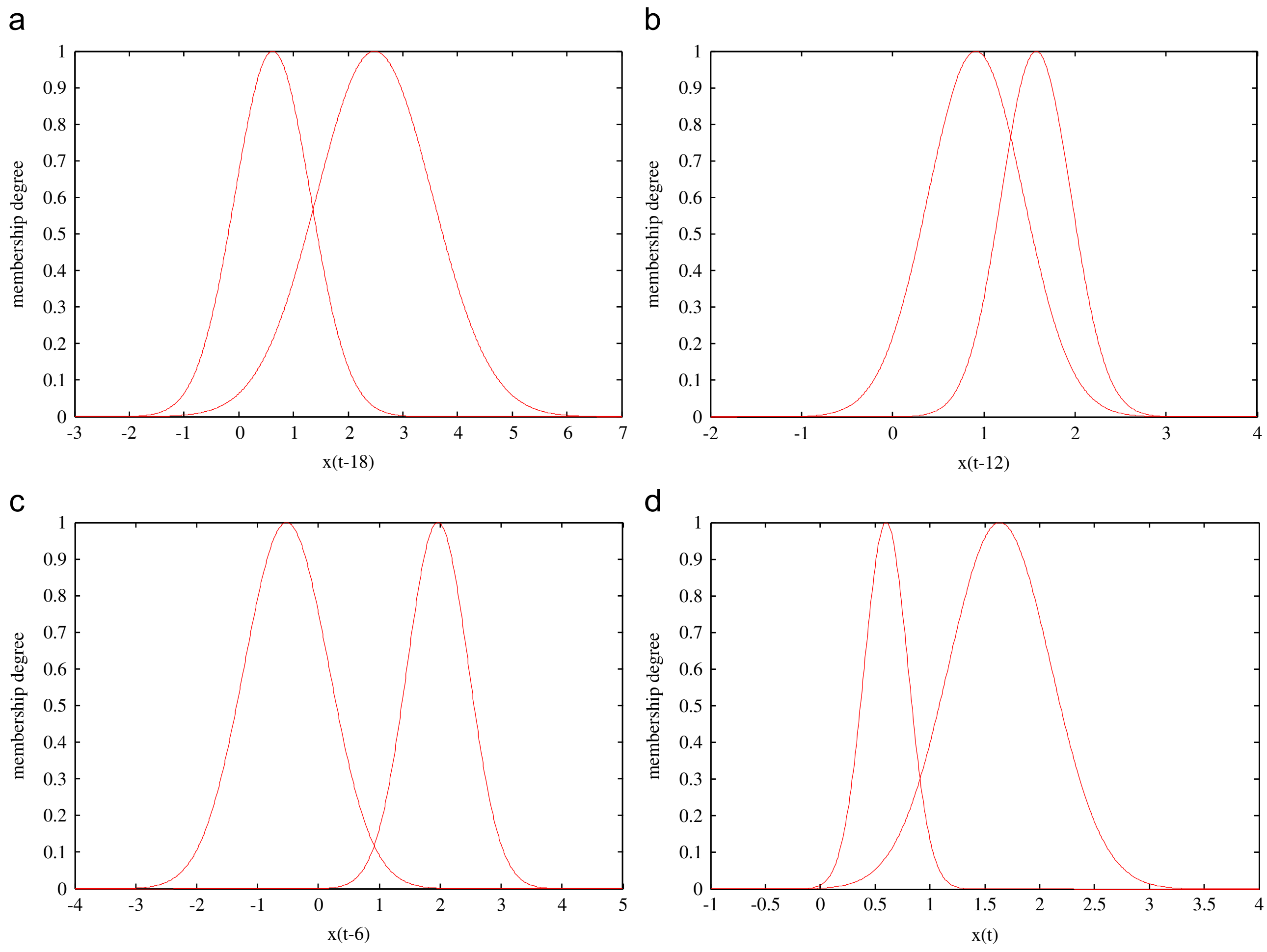

Fig. 8. Fuzzy sets after learning for the four inputs of NFS–ARIMA(4,1,0) (Mackey-Glass chaos time series).

Table 2
Performance comparison (Mackey-Glass chaos time series).

Method	RMSE (training)	RMSE (testing)	NDEI (testing)	Rules
Chen et al. (1991)	0.0158	0.0163	N/A	13
Cho and Wang (1996) (Table 1)	0.0096	0.0114	N/A	23
Cho and Wang (1996) (Table 2)	0.0107	0.0128	N/A	21
Cho and Wang (1996) (Table 3)	0.0119	0.0131	N/A	13
ANFIS (Jang, 1993)	0.0016	0.0015	0.007	16
Nauck and Kruse (1999)	0.1070	0.1080	N/A	26
Paul and Kumar (2002)	0.0053	0.0055	N/A	9
Paul and Kumar (2002)	0.0056	0.0057	N/A	10
HyFIS (Kim and Kasabov, 1999)	0.0021	0.00208	0.0101	16
WNN+gradient (Chen et al., 2006)	0.0067	0.0071	N/A	N/A
WNN+hybrid (Chen et al., 2006)	0.0056	0.0059	N/A	N/A
LLWNN+gradient (Chen et al., 2006)	0.0038	0.0041	N/A	N/A
LLWNN+hybrid (Chen et al., 2006)	0.0033	0.0036	N/A	N/A
FNT model (case 1) (Chen et al., 2005)	0.0069	0.0071	N/A	N/A
ILA (Deng and Wang, 2009)	0.0051	0.0066	N/A	12
G-FNN (Gao and Er, 2005)	0.0063	0.0056	0.0243	10
Rojas et al. (2008)	N/A	0.0025	N/A	N/A
NFS–ARIMA(4,0,0) (proposed)	0.0014	0.0013	0.0063	16
NFS–ARIMA(4,1,0) (proposed)	0.00087	0.00086	0.0038	16

inputs. Each input variable has two fuzzy sets. For each predictor, there are 8 fuzzy rules. The fuzzy sets are with the form of Gaussian membership function. The cost function is designed by MSE. To train the predictors, the settings for the PSO–RLSE hybrid learning method are given in Table 4.

Both of the proposed predictors did well, especially for NFS–ARIMA(3,2,2), whose prediction performance is 1.3977×10^{-4} in MSE for training, 1.9932×10^{-4} in MSE for testing and 0.0536 in NDEI for testing. The learning curve of NFS–ARIMA(3,2,2) is shown in Fig. 9.

Table 3
Parameters of NFS–ARIMA(4,1,0) after learning (Mackey-Glass chaos time series).

Premise-part	$h_1(t)=x(t-18)$		$h_2(t)=x(t-12)$		$h_3(t)=x(t-6)$		$h_4(t)=x(t)$	
Fuzzy set	m	σ	m	σ	m	σ	m	σ
s_1	0.6213	0.6909	1.5747	0.3827	1.9620	0.5052	0.6011	0.1994
s_2	2.4847	1.0569	0.9131	−0.5230	−0.5230	0.6961	1.6365	0.4691

Consequent-part	$a_0+a_1(\psi(t-1))+a_2(\psi(t-2))+a_3(\psi(t-3))+a_4(\psi(t-4))$				
	a_0	a_1	a_2	a_3	a_4
Rule 1	6.8042	8.4777	−10.9620	−2.2863	−8.8171
Rule 2	1.5872	0.1438	−0.9181	−1.1951	−1.9727
Rule 3	−4.3816	7.0817	6.2733	4.1302	−3.3248
Rule 4	8.7253	14.1543	15.9221	3.9280	3.6392
Rule 5	3.0703	−3.6888	0.9658	−3.9910	0.3319
Rule 6	0.1986	0.0718	0.1479	0.0128	−0.2815
Rule 7	0.4397	−0.6756	−0.9697	−0.3098	0.1242
Rule 8	−1.2101	−5.0284	0.3558	−0.3311	−1.7398
Rule 9	−9.7790	7.2091	18.1424	2.5073	8.5484
Rule 10	−3.8518	2.5437	8.3609	5.2280	4.0723
Rule 11	6.5820	−17.8930	−10.6284	−11.4605	5.0834
Rule 12	−11.6517	10.8732	−19.7875	5.5411	−7.9599
Rule 13	−6.3557	8.1249	4.6487	6.8948	−0.9329
Rule 14	−3.4497	−1.0100	−0.7265	1.3667	2.5262
Rule 15	−0.1802	2.2336	0.8853	1.4293	1.3466
Rule 16	3.7691	2.2890	−11.6961	−4.2587	2.3552

Table 4
Settings of the PSO–RLSE hybrid learning method (star brightness time series).

PSO	**NFS–ARIMA(3,0,0)**	**NFS–ARIMA(3,2,2)**
Dimensions of PSO particles	12	12
Swarm size	50	50
Initialization of particle positions	Random in $[0,1]^{12}$	Random in $[0,1]^{12}$
Initialization of particle velocities	Random in $[0,1]^{12}$	Random in $[0,1]^{12}$
Learning rate (c_1,c_2)	(2,2)	(2,2)
Inertia weight w	0.8	0.8
Maximum iterations	100	100
RLSE	**NFS–ARIMA(3,0,0)**	**NFS–ARIMA(3,2,2)**
Number of consequent-part parameters	32	48
$\boldsymbol{\theta}_0$	32×1 Zero vector	48×1 Zero vector
$\mathbf{P}_0$	$\mathbf{P}_0=\alpha\mathbf{I}$	$\mathbf{P}_0=\alpha\mathbf{I}$
α	10^8	10^8
$\mathbf{I}$	32×32 Identity matrix	48×48 Identity matrix

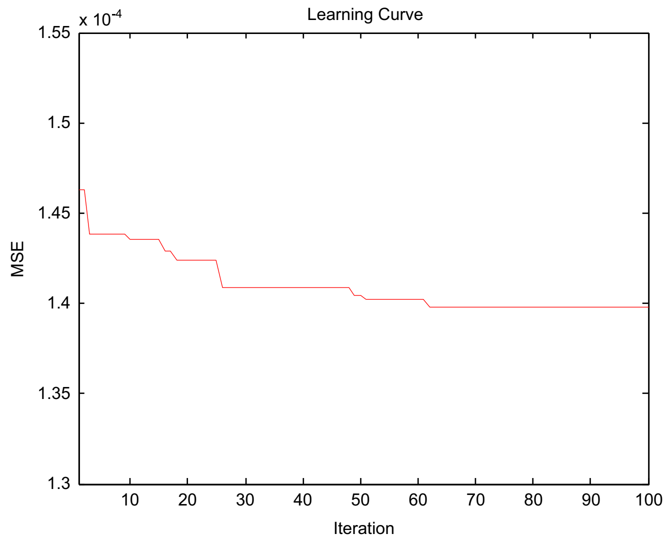

Fig. 9. Learning curve for the proposed NFS–ARIMA(3,2,2) (star brightness time series).

Obviously, the proposed hybrid learning method shows very fast convergence speed. In few iterations, it already reaches near to the converged MSE. The prediction response by the NFS–ARIMA(3,2,2) is shown in Fig. 10.

The performance comparison to other approaches in the literature is given in Table 5, where the NFS–ARIMA(3,2,2) shows outstanding performance to the compared approaches. The parameters of NFS–ARIMA(3,2,2) after training are given in Table 6.

Example 3. Time series of daily IBM stock closing price

The 1000 data of daily IBM stock closing price from March 3, 2006 to February 25, 2010 are obtained from Yahoo Website, denoted as $\{y(i),\ i=1,2,\ldots,1000\}$. This time series exhibits vibrating fluctuation dramatically and the range variety of stock closing prices is severe, from 70 to 130 largely. The data is normalized to the interval of 0 and 1. The first 500 data are used for training and the rest data are for testing. The time series is highly non-stationary. Three predictors NFS–ARIMA(2,0,0), NFS–ARIMA (2,1,0) and NFS–ARIMA(2,1,1) are designed in the example. Because the stock closing price is usually correlated to previous days' stock closing prices, the form of data pairs $(H(t), X(t))$ for the predictors is given as follows:

$$H(t)=[y(t-1),y(t-2)] \tag{39}$$

$$X(t)=y(t) \tag{40}$$

Each of the proposed predictors has two inputs, each of which has two Gaussian fuzzy sets. There are 4 fuzzy rules for each predictor. The cost function is designed by RMSE. The settings of the PSO–RLSE hybrid learning method are given in Table 7.

After learning, the three predictors are compared for their performances. The performance comparison for the three predictors is shown in Table 8, where the NFS–ARIMA(2,1,0) outperforms the NFS–ARIMA(2,0,0) and shows slightly better than NFS–ARIMA(2,1,1).

The prediction responses by the three predictors are shown in Figs. 11–13, respectively. The learning curves for the NFS–ARIMA(2,1,0) and the NFS–ARIMA(2,1,1) are shown in Figs. 14 and 15, respectively.

The parameters of NFS–ARIMA(2,1,0) and NFS–ARIMA(2,1,1) after learning are shown in Tables 9 and 10, respectively.

Remarks. The difference between NFS–ARIMA(2,1,0) and NFS–ARIMA(2,1,1) is only by the 1st-order moving average (MA), which is the information of prediction error at last time step. The information by MA may become either error-correction information to improve prediction or the inquisitive signal to degrade performance, depending on the nature of time series. Through the experiment, NFS–ARIMA(2,1,0) outperforms NFS–ARIMA(2,1,1) slightly, showing the time series of the daily IBM stock could tend to the latter nature. But, the performances by the NFS–ARIMA(2,1,0) and NFS–ARIMA(2,1,1) are very close.

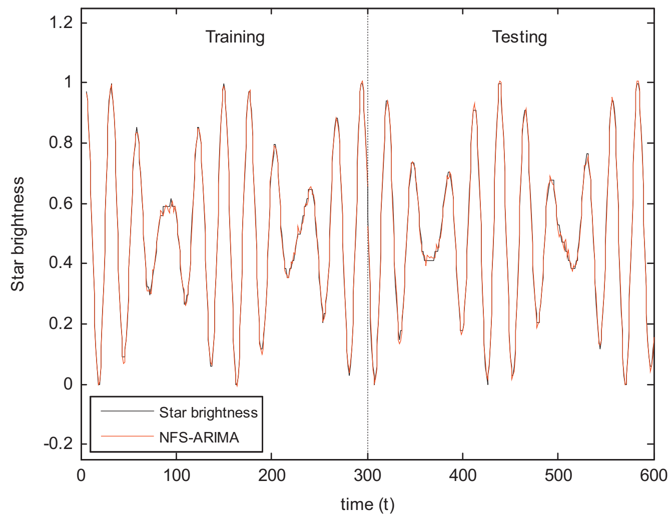

Fig. 10. Prediction response by the proposed NFS–ARIMA(3,2,2) (star brightness time series).

Table 7
Settings of the PSO–RLSE hybrid learning method (IBM stock).

PSO	**NFS–ARIMA(2,0,0), NFS–ARIMA(2,1,0)**	**NFS–ARIMA(2,1,1)**
Dimensions of PSO particles	8	8
Swarm size	25	25
Initialization of particle positions	Random in $[0,1]^8$	Random in $[0,1]^8$
Initialization of particle velocities	Random in $[0,1]^8$	Random in $[0,1]^8$
Learning rate (c_1,c_2)	(2,2)	(2,2)
Inertia weight w	0.8	0.8
Maximum iterations	300	300
RLSE	**NFS–ARIMA(2,0,0), NFS–ARIMA(2,1,0)**	**NFS–ARIMA(2,0,2)**
Number of consequent-part parameters	12	16
$\boldsymbol{\theta}_0$	12×1 Zero vector	16×1 Zero vector
$\boldsymbol{P}_0$	$\boldsymbol{P}_0=\alpha\boldsymbol{I}$	$\boldsymbol{P}_0=\alpha\boldsymbol{I}$
α	10^8	10^8
$\boldsymbol{I}$	12×12 Identity matrix	16×16 Identity matrix

Table 5
Performance comparison (star brightness time series).

Method	**MSE (training)**	**MSE (testing)**	**NDEI (training)**	**NDEI (testing)**
TSK-NFIS (Graves and Pedrycz, 2009)	N/A	3.31×10^{-4}	N/A	0.0609
AR (Graves and Pedrycz, 2009)	N/A	3.22×10^{-4}	N/A	0.0601
NAR (Graves and Pedrycz, 2009)	N/A	3.12×10^{-4}	N/A	0.0592
Neural Net (Graves and Pedrycz, 2009)	N/A	3.11×10^{-4}	N/A	0.0591
NFS–ARIMA(3,0,0) (proposed)	2.0932×10^{-4}	2.6431×10^{-4}	0.0552	0.0617
NFS–ARIMA(3,2,2) (proposed)	1.3977×10^{-4}	1.9932×10^{-4}	0.0449	0.0536

Table 6
Parameters of NFS–ARIMA(3,2,2) after learning (star brightness time series).

Premise-part	$h_1(t)=y(t-1)$		$h_2(t)=y(t-2)$		$h_3(t)=y(t-3)$	
Fuzzy set	m	σ	m	σ	m	σ
s_1	0.9411	0.9089	0.8324	0.5233	0.6538	0.6963
s_2	0.4794	0.6765	0.5253	0.4714	0.1462	0.4550
Consequent-part	$a_0+a_1\psi(t-1)+a_2\psi(t-2)+a_3\psi(t-3)-\theta_1 e(t-1)-\theta_2 e(t-2)$					
	a_0	a_1	a_2	a_3	θ_1	θ_2
Rule 1	−0.5353	−8.0854	4.3155	−1.1188	−7.8592	−26.7271
Rule 2	2.0301	31.5472	75.9568	−57.4514	6.5547	2.4587
Rule 3	−2.7831	16.5157	−58.4757	−2.5255	73.7917	27.6293
Rule 4	0.5993	−89.0049	−45.6803	84.8065	−63.8086	31.2774
Rule 5	2.3748	−9.4537	23.1149	15.0161	−36.5175	52.8312
Rule 6	−2.9898	10.6107	−69.9463	−12.2336	−27.7629	−58.6007
Rule 7	0.4698	4.7646	25.1901	−3.3348	−2.4298	−58.3556
Rule 8	1.0332	37.0406	43.3424	−36.2705	41.0952	24.8450

Table 8
Prediction performance (IBM stock).

Model	RMSE (training)	RMSE (testing)	NDEI (training)	NDEI (testing)
NFS–ARIMA(2,0,0)	0.0191	0.0365	0.0920	0.1439
NFS–ARIMA (2,1,0)	**0.0187**	**0.0318**	**0.0900**	**0.1253**
NFS–ARIMA (2,1,1)	0.0192	0.0320	0.0924	0.1261

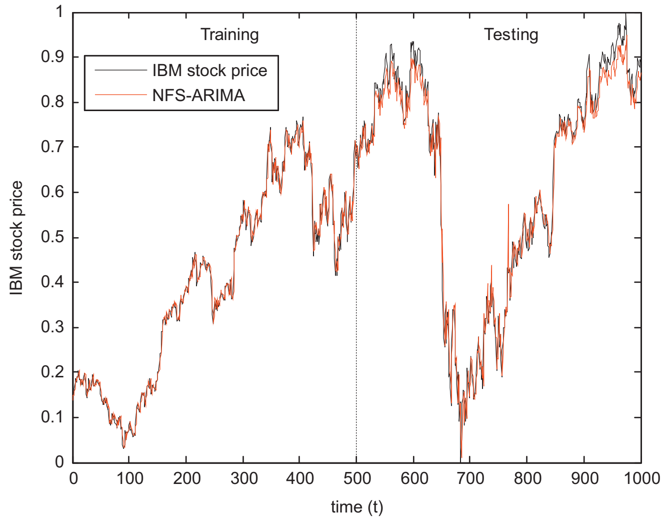

Fig. 11. Prediction response by the proposed NFS–ARIMA(2,0,0) (IBM stock).

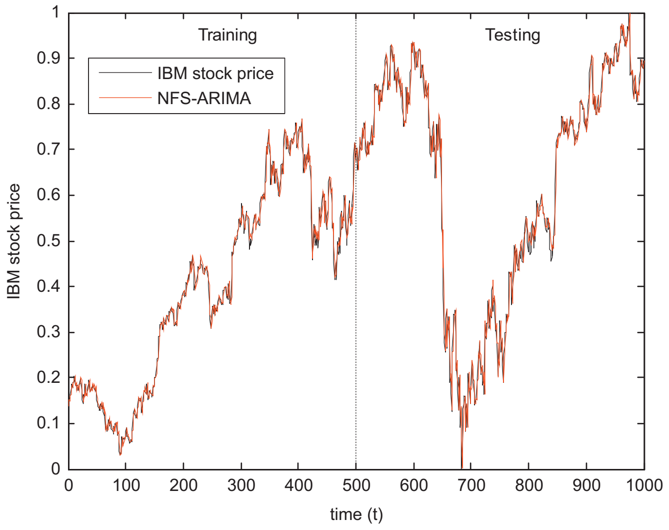

Fig. 12. Prediction response by the proposed NFS–ARIMA(2,1,0) (IBM stock).

5. Discussion and conclusion

The proposed NFS–ARIMA approach with the novel PSO–RLSE hybrid learning method has been presented for the problem of time series forecasting. Three examples have been demonstrated for the proposed approach in the paper. The rationales of neuro-fuzzy system and ARIMAs are combined to form the proposed NFS–ARIMA prediction computing model. An NFS is a hybrid of fuzzy logic and neural network, where fuzzy logic can be used to handle ambiguous information and perform inference to produce useful result for decision-making while neural network is famous for its learning ability and fault tolerance. Both fuzzy inference system (FIS) and neural network (NN) have been proved universal

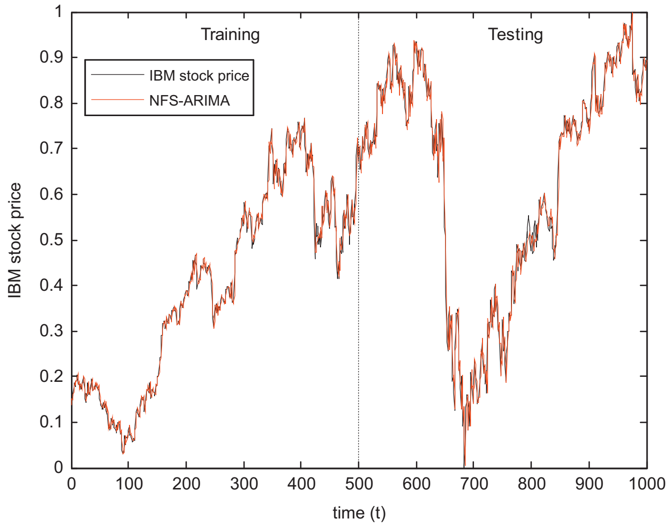

Fig. 13. Prediction response by the proposed NFS–ARIMA(2,1,1) (IBM stock).

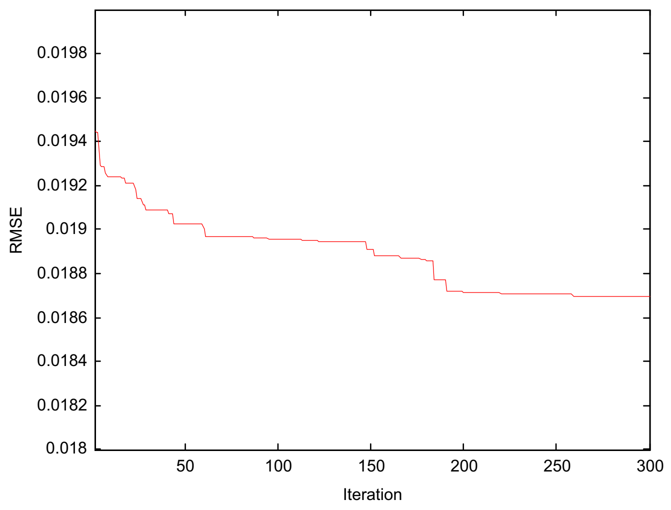

Fig. 14. Learning curve of NFS–ARIMA(2,1,0) (IBM stock).

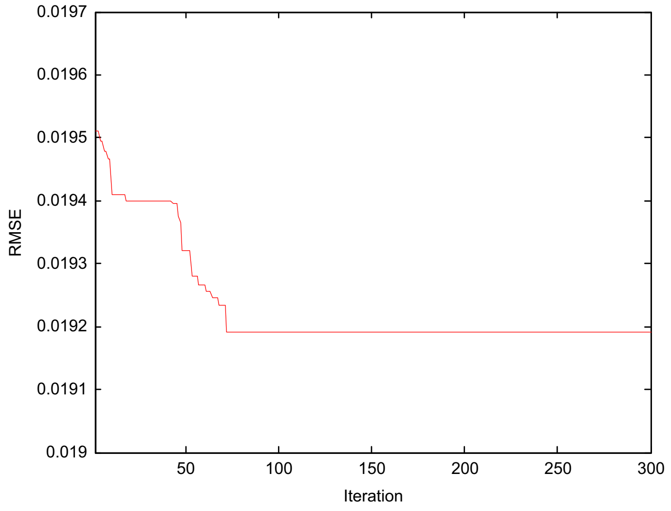

Fig. 15. Learning curve of NFS–ARIMA(2,1,1) (IBM stock).

approximators that the fusion of FIS and NN become natural to have their advantages in applications. Moreover, ARIMA is well-known for its ability to deal with non-stationary time series and

Table 9
Parameters of NFS–ARIMA(2,1,0) after learning (IBM stock).

Premise-part	$h_1(t)=y(t-1)$		$h_2(t)=y(t-2)$	
Fuzzy set	m	σ	m	σ
s_1	0.1835	0.0110	0.9905	0.7835
s_2	0.5219	0.0031	0.8923	0.1325
Consequent-part	$a_0+a_1(\psi(t-1))+a_2(\psi(t-2))$			
	a_0	a_1	a_2	
Rule 1	0.0013	0.0303	−0.0524	
Rule 2	−0.1161	−136.6935	258.6373	
Rule 3	0.0173	−0.4383	−0.1769	
Rule 4	−0.3899	6.1224	4.2057	

Table 10
Parameters of NFS–ARIMA(2,1,1) after learning (IBM stock).

Premise-part	$h_1(t)=y(t-1)$		$h_2(t)=y(t-2)$	
Fuzzy set	m	σ	m	σ
s_1	0.7861	0.4566	0.3980	0.7747
s_2	0.5861	0.8469	0.4930	0.0312
Consequent-part	$a_0+a_1(\psi(t-1))+a_2(\psi(t-2))-\theta_1 e(t-1)$			
	a_0	a_1	a_2	θ_1
Rule 1	−0.0138	2.3891	0.1131	1.9712
Rule 2	−0.0084	127.7507	−3.6844	129.6702
Rule 3	0.0094	−1.7722	−0.0320	−1.6474
Rule 4	0.02763	−105.8003	2.6838	−104.8827

its capability of coordinating the models of AR, its integration part and MA. Therefore, the proposed NFS–ARIMA model becomes an excellent adaptive predictor with great flexibility in structure selection, in terms of AR, ARI, MA, ARMA, or ARIMA. And, it also has great potential to be an accurate predictor, if appropriate learning method is applied to adapt the predictor to optimal state (or near optimal state).

For the parameter learning perspective, the set of NFS–ARIMA parameters is divided into two subsets, the subset of If-part parameters and the subset of Then-part parameters. The division of the parameter set of NFS–ARIMA makes it much easier to evolve the model to the optimal (or near optimal) state. The novel PSO–RLSE hybrid learning method has been devised for this purpose. The well-known PSO is excellent in swarm-based search for optimization. With appropriate settings, PSO has much less chance of being trapped by a local minimum in the problem space. On the other hand, for the linear least squares estimation problem, the RLSE is famous for its capability of speedy optimization, using much less computational overhead than its traditional LSE version, in terms of computational time and resource. As a result, the hybrid PSO–RLSE method shows very fast learning convergence for the proposed NFS–ARIMA, for example, the learning curves in Figs. 6, 9, 14, and 15 for the three examples, for each of which within few iterations the cost is near to its converged value. And, the prediction performance by the proposed approach is superior or comparable to those by the compared researches, for example, as shown in Tables 2 and 5. In Example 3, with 4 fuzzy rules only, the proposed approach has been tested, using the IBM stock time series to show its capability for real-world application. As a result, the NFS–ARIMA(2,1,0) and the NFS–ARIMA(2,1,1) performed better than the NFS–ARIMA(2,0,0), as shown in Table 8. In this case the NFS–ARIMA(2,1,0) and the NFS–ARIMA(2,1,1) have shown over 12% more accurate than the NFS–ARIMA(2,0,0) in testing. This confirms our thought that with difference processing (integration part for the model) the proposed approach may perform better prediction. Moreover, in Example 1, with NFS–ARIMA(4,1,0) comparing to NFS–ARIMA(4,0,0), even much better (more than 33%) result has been shown in Table 2 for Mackey-Glass chaos time series forecasting. In Example 2, similar effect is found as well.

The main advantages of the proposed approach are specified. First, the new NFS–ARIMA hybrid model has been proposed to time series forecasting, with which more accurate forecasting can be made. Second, the novel PSO–RLSE hybrid learning method has been presented to make accurate prediction possible. The parameters of NSF-ARIMA are divided into two subsets, the If-part subset and the Then-part subset. With this division strategy, the PSO–RLSE method has evolved the proposed NFS–ARIMA with excellent performance. Third, the prediction performances by the proposed approach are great, as shown in the three examples. In Tables 2 and 5, the performances by the proposed approach are generally much better than the compared approaches.

Acknowledgments

This research work is supported by the National Science Council, Taiwan (R.O.C.), under the Grant contract no. NSC98-2221-E-008-086. The authors thank the anonymous reviewers for their constructive comments.

References

Box, G.E.P., Jenkins, G.M., 1976. Time Series Analysis: Forecasting and ControlHolden-Day, San Francisco, CA, USA.
Chen, S., Cowan, C.F.N., Grant, P.M., 1991. Orthogonal least squares learning algorithm for radial basis function networks. IEEE Trans. Neural Networks 2, 302–309.
Cho, K.B., Wang, B.H., 1996. Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction. Fuzzy Sets Syst. 83, 325–339.
Chen, Y., Yang, B., Dong, J., Abraham, A., 2005. Time-series forecasting using flexible neural tree model. Inf. Sci. 174, 219–235.
Chen, Y., Yang, B., Dong, J., 2006. Time-series prediction using a local linear wavelet neural network. Neurocomputing 69, 449–465.
Coello, C.A., Pulido, G.T., Lechuga, M.S., 2004. Handling multiple objectives with particle swarm optimization. IEEE Trans. Evol. Comput. 8, 256–279.
Deng, X., Wang, X., 2009. Incremental learning of dynamic fuzzy neural networks for accurate system modeling. Fuzzy Sets Syst. 160, 972–987.
Fukami, S., Mizumoto, M., Tanaka, K., 1980. Some considerations on fuzzy conditional inference. Fuzzy Sets Syst. 4, 243–273.
Gao, Y., Er, M.J., 2005. NARMAX time series model prediction: feedforward and recurrent fuzzy neural network approaches. Fuzzy Sets Syst. 150, 331–350.
Graves, D., Pedrycz, W., 2009. Fuzzy prediction architecture using recurrent neural networks. Neurocomputing 72, 1668–1678.
Herrera, L.J., Pomares, H., Rojas, I., Guillen, A., Gonzalez, J., Awad, M., Herrera, A., 2007. Multigrid-based fuzzy systems for time series prediction: CATS competition. Neurocomputing 70, 2410–2425.
Hyndman, R.J. Time Series Data Library. Physics. Daily Brightness of a Variable Star on 600 Successive Midnights. Available: ⟨http://www-personal.buseco.monash.edu.au/hyndman/TSDL/⟩, 2010.
Jang, J.S.R., 1993. ANFIS: adaptive-network-based fuzzy inference system. IEEE Trans. Syst., Man, Cybern. 23, 665–685.
Jang, J.S.R., Sun, C.T., Mizutani, E., 1997. Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine IntelligencePrentice Hall, Upper Saddle River, NJ,USA.
Juang, C.-F., 2004. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Trans. Syst., Man, Cybern., Part B: Cybern. 34, 997–1006.
Kim, E., Park, M., Ji, S., Park, M., 1997. A new approach to fuzzy modeling. IEEE Trans. Fuzzy Syst. 5, 328–337.
Kasabov, N.K., Kim, J., Watts, M.J., Gray, A.R., 1997. FuNN/2—a fuzzy neural network architecture for adaptive learning and knowledge acquisition. Inf. Sci. 101, 155–175.
Kim, J., Kasabov, N., 1999. HyFIS: adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems. Neural Networks 12, 1301–1319.
Klir, G.J., Yuan, B., 1995. Fuzzy Sets and Fuzzy Logic: Theory and ApplicationsPrentice Hall, Upper Saddle River, NJ, USA.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948.

Nauck, D., Kruse, R., 1999. Neuro-fuzzy systems for function approximation. Fuzzy Sets Syst. 101, 261–271.

Paul, S., Kumar, S., 2002. Subsethood-product fuzzy neural inference system (SuPFuNIS). IEEE Trans. Neural Networks 13, 578–599.

Parsopoulos, K., Vrahatis, M.N., 2002a. Recent approaches to global optimization problems through particle swarm optimization. Nat. Comput. 1, 235–306.

Parsopoulos, K.E., Vrahatis, M.N., 2002b. Particle swarm optimization method for constrained optimization problems. Intell. Technol.—Theory Appl.: New Trends Intell. Technol. 76, 214–220.

Rong, H.J., Sundararajan, N., Huang, G.B., Saratchandran, P., 2006. Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction. Fuzzy Sets Syst. 157, 1260–1275.

Rojas, I., Valenzuela, O., Rojas, F., Guillen, A., Herrera, L.J., Pomares, H., Marquze, L., Pasadas, M., 2008. Soft-computing techniques and ARMA model for time series prediction. Neurocomputing 71, 519–537.

Sfetsos, A., Siriopoulos, C., 2004. Time series forecasting with a hybrid clustering scheme and pattern recognition. IEEE Trans. Syst., Man Cybern., Part A: Syst. Hum. 34, 399–405.

Sugiarto, I., Natarajan, S., 2007. Parameter estimation using least square method for MIMO Takagi–Sugeno neuro-fuzzy in time series forecasting. J. Tek. Elektro. 7 (2), 82–87.

Sugeno, M., Kang, G.T., 1988. Structure identification of fuzzy model. Fuzzy Sets Syst. 28, 15–33.

Shi, Y., Eberhart, R.C., 1999. Empirical study of particle swarm optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation, CEC 99.

Shi, Y., Eberhart, R., 1998. Parameter selection in particle swarm optimization. Lect. Notes Comput. Sci. 1447, 591–600.

Takagi, T., Sugeno, M., 1985. Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. Syst., Man, Cybern. 15, 116–132.

Xiong, N., 2001. Evolutionary learning of rule premises for fuzzy modelling. Int. J. Syst. Sci. 32, 1109–1118.

Yahoo Website. IBM Stocks. Available: 〈http://ichart.finance.yahoo.com/table.csv?s=IBM&a=02&b=08&c=2006&d=01&e=25&f=2010&g=d&ignore=.csv〉, 2010.

Zounemat-Kermani, M., Teshnehlab, M., 2008. Using adaptive neuro-fuzzy inference system for hydrological time series prediction. Appl. Soft Comput. 8, 928–936.

Zhao, L., Yang, Y., 2009. PSO-based single multiplicative neuron model for time series prediction. Expert Syst. Appl. 36, 2805–2812.

Zadeh, L.A., 1965. Fuzzy sets. Inf. Control 8, 338–353.

Zadeh, L.A., 1975. Fuzzy logic and approximate reasoning. Synthese 30, 407–428.