Available online at www.sciencedirect.com
SCIENCE DIRECT®
Fuzzy Sets and Systems 157 (2006) 1260–1275
FUZZY sets and systems
www.elsevier.com/locate/fss

Sequential Adaptive Fuzzy Inference System (SAFIS) for nonlinear system identification and prediction

Hai-Jun Rong, N. Sundararajan*, Guang-Bin Huang, P. Saratchandran

School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798, Singapore

Received 7 April 2005; received in revised form 17 November 2005; accepted 19 December 2005
Available online 18 January 2006

Abstract

In this paper, a Sequential Adaptive Fuzzy Inference System called SAFIS is developed based on the functional equivalence between a radial basis function network and a fuzzy inference system (FIS). In SAFIS, the concept of "Influence" of a fuzzy rule is introduced and using this the fuzzy rules are added or removed based on the input data received so far. If the input data do not warrant adding of fuzzy rules, then only the parameters of the "closest" (in a Euclidean sense) rule are updated using an extended kalman filter (EKF) scheme. The performance of SAFIS is compared with several existing algorithms on two nonlinear system identification benchmark problems and a chaotic time series prediction problem. Results indicate that SAFIS produces similar or better accuracies with less number of rules compared to other algorithms.

Keywords: Sequential adaptive fuzzy inference system (SAFIS); GAP–RBF; GGAP–RBF; "Influence" of a fuzzy rule; Extended Kalman filter

1. Introduction

It is well known that Fuzzy Inference Systems (FIS) can be used to approximate closely any nonlinear input–output mapping by means of a series of if–then rules [9]. In the design of FIS, there are two major tasks, viz., the structure identification and the parameter adjustment. Structure identification determines the input–output space partition, antecedent and consequent variables of if–then rules, number of such rules, and initial positions of membership functions. The second task of parameter adjustment involves realizing the parameters for the fuzzy system structure determined in the previous step [15].

Recently, the functional equivalence between a RBF neural network and a FIS (under certain conditions [8]) have been exploited by many researchers for carrying out the above two tasks [14,4,10,18,13]. These schemes utilize the learning capabilities of the RBF for changing the rules as well as adjusting the parameters since the hidden neurons of the RBF networks are related to the fuzzy rules.

These methods can be broadly divided into two classes namely batch learning schemes and sequential learning schemes. In batch learning, it is assumed that the complete training data are available before the training commences. The training usually involves cycling the data over a number of epochs. In sequential learning, the data arrive one by

* Corresponding author. Tel.: +65 6790 5027; fax: +65 6793 3318.
E-mail address: ensundara@ntu.edu.sg (N. Sundararajan).

0165-0114/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.fss.2005.12.011

one and after the learning of each data it is discarded and the notion of epoch does not exist. In practical applications, new training data arrive sequentially and to handle this using batch learning, one has to retrain the network all over again, resulting in large training time.

Hence, in these cases, sequential learning algorithms are generally preferred over batch learning algorithms as they do not require retraining whenever a new data is received. Compared with the batch learning algorithms, the sequential learning algorithm which is discussed in this paper, i.e. SAFIS, has the following distinguishing features [7]:

(1) All the training observations are *sequentially* (one-by-one) presented to the system.
(2) At any time, *only one* training observation is seen and learned.
(3) A training observation is *discarded* as soon as the learning procedure for that particular observation is completed.
(4) The learning system has no *prior* knowledge as to how many total training observations will be presented.

Thus, if one strictly applies the above features of the sequential algorithms, many of the above algorithms are not sequential. One major bottleneck seems to be that they need the entire training data ready for training before the training procedure starts and thus they are not really sequential. This point is highlighted in a brief review of the existing algorithms given below.

Lee and Shin [14] have used the least-square method and genetic algorithm to construct fuzzy systems when all the data are available and hence this method is not sequential.

A significant contribution to sequential learning in RBF network was made by Platt [17] through the development of Resource Allocation Network (RAN). In RAN, the network starts with no hidden neurons but adds hidden neurons based on the novelty of the input data. Most of the recent algorithms for adaptively creating fuzzy systems are based on the ideas of RAN. These algorithms claim to be "on-line" algorithms and if one looks closely at them, they are not sequential as per the above distinguishing features.

A hierarchically self-organizing approach proposed by Cho and Wang [4] automatically generates fuzzy rules without predefining the number of fuzzy rules based on the error and distance criterion of fuzzy basis functions. However, the algorithm requires cycling the whole training data over a number of learning cycles (epochs) and hence it is not a truly sequential learning scheme.

Recently, a self-constructing neural fuzzy inference network (SONFIN) has been proposed by Juang and Lin [10] in which the fuzzy rules are extracted online from the training data. For adding a new fuzzy rule, SONFIN utilizes the distance criterion between the new input data and the center of the Gaussian membership function in the existing fuzzy rules. Although this algorithm is sequential in nature, it does not remove the fuzzy rules once created even though that rule is not effective. This may result in a structure where the number of rules may be large.

In most of the real applications, not all fuzzy rules contribute significantly to the system performance during the entire time period. A fuzzy rule may be active initially, but may later contribute little to the system output. For this reason, the insignificant fuzzy rules have to be removed during learning to realize a compact fuzzy system structure. Using the ideas of adding and pruning hidden neurons to form a minimal RBF network in [20], a hierarchical on-line self-organizing learning algorithm for dynamic fuzzy neural networks (DFNN) has been proposed in [18]. Another on-line self-organizing fuzzy neural network (SOFNN) proposed by Leng et al. [13] also includes a pruning method. The pruning method utilizes the optimal brain surgeon (OBS) approach to determine the importance of each rule. However, in these two algorithms the pruning criteria need all the past data received so far. Hence, they are not strictly sequential and further requires increased memory for storing all the past data.

Angelov and Filev [1] have proposed an evolving Takagi–Sugeno model (eTS) that recursively updates TS model structure based on the *potential* of the input data (defined based on its distances to all other data points received so far). In this algorithm, a new rule is added when the potential of the new data is higher than the potential of the existing rules or a new rule is modified when the potential of the new data is higher than the potential of the existing rules and the new data are close to an old rule. This is a truly sequential learning algorithm. However, the algorithm cannot simplify the rule base during learning by ignoring the rules which may become irrelevant with the future data samples when the data sample sequentially arrives. A simplified version of the eTS learning algorithm that simplifies the rule base, called the simpl_eTS, is proposed by Angelov and Filev [2]. The algorithm utilizes the concept of the scatter, which is similar to the notion of potential but computationally more efficient. The algorithm can simplify the rule base to make the rules representative based on the population of each rule determined by the number of the data samples that belong to a particular cluster. If the population of a rule is less than 1% of the total data at the moment of appearance of a rule, the rule is ignored from the rule base by setting its firing strength to zero.

In this paper, a SAFIS is developed to realize a compact fuzzy system with lesser number of rules. SAFIS uses the idea of functional equivalence between a RBF neural network and a fuzzy inference system. Here, SAFIS uses the GAP–RBF neural network recently proposed by Huang et al. [6]. The SAFIS algorithm consists of two aspects: determination of the fuzzy rules and adjustment of the premise and consequent parameters in fuzzy rules.

SAFIS uses the concept of *influence* of a fuzzy rule to add and remove rules during learning. SAFIS starts with no fuzzy rules and based on the data builds up a compact rule base. During the learning, only the current data are made use of and there is no need to store all the past data. The *influence* of a fuzzy rule is defined as its contribution to the system output in a statistical sense. Here, we have derived an expression for this for the case where the input data are uniformly distributed.

The parameter adjustment is done using a winner rule strategy where the winner rule is defined as the one closest to the input data and the parameter update is done using an EKF mechanism.

It should be noted that SAFIS is a truly sequential learning algorithm.

In order to evaluate the performance of SAFIS algorithm with the others, a comparison is carried out on two nonlinear system identification problems and one chaotic time series (Mackey–Glass) prediction problem. The comparison is done in terms of accuracy and the complexity (the number of rules) of the fuzzy system. Results indicate SAFIS produces similar or better accuracies with a smaller number of rules for the FIS.

This paper is organized as follows. Section 2 presents the structure of SAFIS along with the definition for the "influence" of fuzzy rules. This section also presents the details of the learning algorithm of the proposed SAFIS. Section 3 presents a quantitative performance comparison for SAFIS with other algorithms based on several benchmark problems in the nonlinear identification and prediction area. Section 4 summarizes the conclusions from this study.

2. SAFIS—architecture and algorithm

Generally, a wide class of MIMO nonlinear dynamic systems can be represented by the nonlinear discrete model with an input–output description form:

$$\mathbf{y}(n) = \mathbf{f}[\mathbf{y}(n-1), \mathbf{y}(n-2), \ldots, \mathbf{y}(n-k+1); \mathbf{u}(n), \mathbf{u}(n-1), \ldots, \mathbf{u}(n-p+1)], \tag{1}$$

where $\mathbf{y}$ is a vector containing N_y system outputs, $\mathbf{u}$ is a vector for N_u system inputs; $\mathbf{f}$ is a nonlinear vector function, representing N_y hypersurfaces of the system, and k and p are the maximum lags of the output and input, respectively.

Selecting $[\mathbf{y}(n-1), \ldots, \mathbf{y}(n-k+1); \mathbf{u}(n), \mathbf{u}(n-1), \ldots, \mathbf{u}(n-p+1)], \mathbf{y}(n)$ as the fuzzy system's input–output $\mathbf{x}_n, \mathbf{y}_n$ at time n, the above equation can be put as

$$\mathbf{y}_n = \mathbf{f}(\mathbf{x}_n). \tag{2}$$

The aim of the new SAFIS algorithm is to approximate $\mathbf{f}$ such that

$$\hat{\mathbf{y}}_n = \hat{\mathbf{f}}(\mathbf{x}_n), \tag{3}$$

where $\hat{\mathbf{y}}_n$ is the output of SAFIS. This means that the objective is to minimize the error between the system output and the output of SAFIS, $\|\mathbf{y}_n - \hat{\mathbf{y}}_n\|$. Before describing the details of the algorithm, the structure of SAFIS network is first described below.

The structure of SAFIS illustrated by Fig. 1 consists of five layers to realize the following fuzzy rule model:

Rule k: if $(x_1 \text{ is } A_{1k}) \cdots (x_{N_x} \text{ is } A_{N_x k})$, then $(\hat{y}_1 \text{ is } a_{1k}) \cdots (\hat{y}_{N_y} \text{ is } a_{N_y k})$ where $a_{jk} (j = 1, 2, \ldots, N_y, k = 1, 2, \ldots, N_h)$ is a constant consequent parameter in rule k, $A_{ik} (i = 1, 2, \ldots, N_x)$ is the membership value of the ith input variable x_i in rule k, N_x is the dimension of the input vector $\mathbf{x} (\mathbf{x} = [x_1, \ldots, x_{N_x}]^{\mathrm{T}})$, N_h is the number of fuzzy rules, N_y is the dimension of the output vector $\hat{\mathbf{y}} (\hat{\mathbf{y}} = [\hat{y}_1, \ldots, \hat{y}_{N_y}]^{\mathrm{T}})$. In SAFIS, the number of fuzzy rules N_h varies. Initially, there is no fuzzy rule and then during learning fuzzy rules are added and removed.

Layer 1: In layer 1, each node represents an input variable and directly transmits the input signal to layer 2.

Layer 2: In this layer, each node represents the membership value of each input variable. SAFIS utilizes the function equivalence between a RBF network and a FIS and thus its antecedent part (if part) in fuzzy rules is achieved by Gaussian functions of the RBF network. The membership value $A_{ik}(x_i)$ of the ith input variable x_i in the kth Gaussian

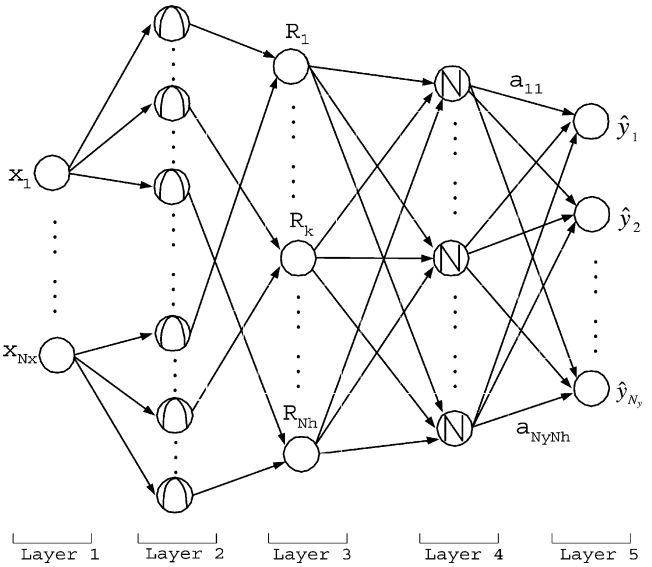

Fig. 1. Structure of SAFIS.

function is given by

$$A_{ik}(x_i) = \exp\left(-\frac{(x_i - \mu_{ik})^2}{\sigma_k^2}\right), \quad k = 1, 2, \ldots, N_h, \tag{4}$$

where N_h is the number of the Gaussian functions, μ_{ik} is the center of the kth Gaussian function for the ith input variable, σ_k is the width of the kth Gaussian function. In SAFIS, the width of all the input variables in the kth Gaussian function is the same.

Layer 3: Each node in the layer represents the if part of if–then rules obtained by the sum–product composition and the total number of such rules is N_h. The firing strength (if part) of the kth rule is given by

$$R_k(\mathbf{x}) = \prod_{i=1}^{N_x} A_{ik}(x_i) = \exp\left(-\sum_{i=1}^{N_x} \frac{(x_i - \mu_{ik})^2}{\sigma_k^2}\right) = \exp\left(-\frac{\|\mathbf{x} - \mu_k\|^2}{\sigma_k^2}\right). \tag{5}$$

Layer 4: The nodes in the layer are named as normalized nodes whose number is equal to the number of the nodes in third layer. The kth normalized node is given by

$$\bar{R}_k = \frac{R_k(\mathbf{x})}{\sum_{k=1}^{N_h} R_k(\mathbf{x})}. \tag{6}$$

Layer 5: Each node in this layer corresponds to an output variable, which is given by the weighted sum of the output of each normalized rule. The system output is calculated by

$$\hat{\mathbf{y}} = \frac{\sum_{k=1}^{N_h} R_k(\mathbf{x}) a_k}{\sum_{k=1}^{N_h} R_k(\mathbf{x})}, \tag{7}$$

where $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{N_y}]^{\mathrm{T}}$, $a_k = [a_{1k}, a_{2k}, \ldots, a_{N_y k}]^{\mathrm{T}}$.

2.1. "Influence" of a fuzzy rule

As per Eq. (7), the contribution of the kth rule to the overall output for an input observation $\mathbf{x}_l$ is given by

$$E(k,l) = |a_k| \frac{R_k(\mathbf{x}_l)}{\sum_{k=1}^{N_h} R_k(\mathbf{x}_l)}. \tag{8}$$

Then the contribution of the kth rule to the overall output based on all input data N received so far is obtained by

$$E(k) = |a_k| \frac{\sum_{l=1}^{N} R_k(\mathbf{x}_l)}{\sum_{k=1}^{N_h} \sum_{l=1}^{N} R_k(\mathbf{x}_l)}. \tag{9}$$

Dividing both the numerator and denominator by N in Eq. (9), the equation becomes

$$E(k) = |a_k| \frac{\sum_{l=1}^{N} R_k(\mathbf{x}_l)/N}{\sum_{k=1}^{N_h} \sum_{l=1}^{N} R_k(\mathbf{x}_l)/N}. \tag{10}$$

Using the significance concept of GAP–RBF [6], the *influence* of the kth fuzzy rule is defined as its statistical contribution to the overall output of SAFIS. When $N \to \infty$ the influence of the kth rule is given by

$$E_{\inf}(k) = \lim_{N\to\infty} E(k) = \lim_{N\to\infty} |a_k| \frac{\sum_{l=1}^{N} R_k(\mathbf{x}_l)/N}{\sum_{k=1}^{N_h} \sum_{l=1}^{N} R_k(\mathbf{x}_l)/N}. \tag{11}$$

Calculation of $E_{\inf}(k)$ using the above equation requires the knowledge of $(\mathbf{x}_l, \mathbf{y}_l), l = 1, \ldots, N$. In the truly sequential learning scheme, this is not possible. An alternate way of calculating $E_{\inf}(k)$ is by using the distribution of the inputs and follows the same approach as introduced in [7]. In order to compute $E_{\inf}(k)$ one has to compute first E_k defined by

$$E_k = \lim_{N\to\infty} \frac{\sum_{l=1}^{N} R_k(\mathbf{x}_l)}{N}. \tag{12}$$

Assume that the observations $(\mathbf{x}_l, \mathbf{y}_l)$, $l = 1, \ldots,$ are drawn from a sampling range X with a sampling density function $p(\mathbf{x})$. Consider a situation where N observations have been learned by the sequential learning scheme. Let the sampling range X be divided into M small spaces Δ_j, $j = 1, \ldots, M$. The size of Δ_j is represented by $S(\Delta_j)$. Since the sampling density function is $p(\mathbf{x})$ there are around $N \cdot p(\mathbf{x_j}) \cdot S(\Delta_j)$ samples in each Δ_j, where $\mathbf{x_j}$ is any point chosen in Δ_j. When the number of input observations N is large, Δ_j is small. From Eq. (12), we have

$$\begin{aligned} E_k &\approx \lim_{M\to\infty} \frac{\sum_{j=1}^{M} R_k(\mathbf{x}_j) \cdot Np(\mathbf{x}_j) \cdot S(\Delta_j)}{N} \\ &= \lim_{M\to\infty} \sum_{j=1}^{M} R_k(\mathbf{x}_j) \cdot p(\mathbf{x}_j) \cdot S(\Delta_j) \\ &= \int_X R_k(\mathbf{x}) p(\mathbf{x})\, \mathrm{d}\mathbf{x} \\ &= \int_X \exp\left(-\frac{\|\mathbf{x} - \mu_k\|^2}{\sigma_k^2}\right) p(\mathbf{x})\, \mathrm{d}\mathbf{x}. \end{aligned} \tag{13}$$

If the distribution of the N_x attributes $(x_1, \ldots, x_i, \ldots, x_{N_x})^{\mathrm{T}}$ of observations $\mathbf{x}'$s are independent from each other, the density function $p(\mathbf{x})$ of $\mathbf{x}$ can be written as: $p(\mathbf{x}) = \prod_{i=1}^{N_x} p_i(x_i)$, where $p_i(x)$ is the density function of the ith attribute x_i of observations. In this case, Eq. (13) can be rewritten as

$$\begin{aligned} E_k &= \int \cdots \int_X \exp\left(-\frac{\|\mathbf{x} - \mu_k\|^2}{\sigma_k^2}\right) p(\mathbf{x})\, \mathrm{d}\mathbf{x} \\ &= \prod_{i=1}^{N_x} \left(\int_{a_i}^{b_i} \exp\left(-\frac{\|x - \mu_{k,i}\|^2}{\sigma_k^2}]\right) p_i(x)\, \mathrm{d}x \right), \end{aligned} \tag{14}$$

where N_x is the dimension of the input space X and (a_i, b_i) the interval of the ith attribute x_i of observations, $i = 1, \ldots, N_x$.

Eq. (14) involves an integration of the probability density function $p(\mathbf{x})$ in the sampling range X. When the input samples are uniformly drawn from a range X, the sampling density function $p(\mathbf{x})$ is given by $p(\mathbf{x}) = 1/S(X)$, where $S(X)$ is the size of the range X given by $S(X) = \int_{\mathbf{x}} 1 \, \mathrm{d}\mathbf{x}$. Substituting for $p(\mathbf{x})$ in Eq. (13) we get

$$E_k = \int_X \exp\left(-\frac{\|\mathbf{x} - \mu_k\|^2}{\sigma_k^2}\right) \frac{1}{S(X)} \, \mathrm{d}\mathbf{x}. \tag{15}$$

Note that in general the width σ_k of a rule k is much less than the size of range X, the above equation can be approximated as

$$\begin{aligned} E_k &\approx \frac{1}{S(X)} \left(2 \int_0^{+\infty} \exp\left(-\frac{x^2}{\sigma_k^2}\right) \mathrm{d}x\right)^{N_x} \\ &= \frac{\pi^{N_x/2} \sigma_k^{N_x}}{S(X)} \\ &\approx \frac{(1.8\sigma_k)^{N_x}}{S(X)}. \end{aligned} \tag{16}$$

Thus, based on Eq. (16) the influence of the kth rule is given by

$$E_{\mathrm{inf}}(k) = |a_k| \frac{(1.8\sigma_k)^{N_x}}{\sum_{k=1}^{N_h} (1.8\sigma_k)^{N_x}}. \tag{17}$$

Remark. The significance of a neuron proposed in GAP–RBF [6] is defined based on the average contribution of an individual neuron to the output of the RBF network. Under this definition, one may need to estimate the input distribution range $S(X)$. However, the influence of a rule introduced in this paper is different from the significance of a neuron proposed in GAP–RBF [6]. In fact, the influence of a neuron is defined as the relevant significance of the neuron compared to summation of significance of all the existing RBF neurons. As seen from Eq. (17), with the introduction of influence one need not estimate the input distribution range $S(X)$ and the implementation has been simplified.

2.2. SAFIS algorithm

The learning algorithm of SAFIS consists of two aspects: determination of fuzzy rules and adjustment of the premise and consequent parameters in fuzzy rules. SAFIS can automatically add and remove fuzzy rules using ideas similar to GAP–RBF [6] for hidden neurons. A description of dynamically adding and removing the fuzzy rules along with the details of parameter adjustment when there are no addition of rules is given below.

2.2.1. Adding of fuzzy rules

SAFIS begins with no fuzzy rules. As inputs $\mathbf{x}_n, \mathbf{y}_n$ (n is the time index) are received sequentially during learning, growing of fuzzy rules is based on the following two criteria which are distance criterion and the influence of the new added fuzzy rule $N_h + 1$:

$$\begin{cases} \|\mathbf{x}_n - \mu_{nr}\| > \varepsilon_n, \\ E_{\mathrm{inf}}(N_h + 1) = |e_n| \dfrac{(1.8\kappa \|\mathbf{x}_n - \mu_{nr}\|)^{N_x}}{\sum_{k=1}^{N_h+1} (1.8\sigma_k)^{N_x}} > e_{\mathrm{g}}, \end{cases} \tag{18}$$

where ε_n, e_{g} are thresholds to be selected appropriately, $\mathbf{x}_n$ is the latest input data, μ_{nr} is the center of the fuzzy rule nearest to $\mathbf{x}_n$. e_{g} is the growing threshold and is chosen according to the desired accuracy of SAFIS. $e_n = \mathbf{y}_n - \hat{\mathbf{y}}_n$, $\mathbf{y}_n$ is the true value, $\hat{\mathbf{y}}_n$ is the approximated value, κ is an overlap factor that determines the overlap of fuzzy rules in the input space, ε_n is the distance threshold which decays exponentially and is given by

$$\varepsilon_n = \max\left\{\varepsilon_{\max} \times \gamma^n, \varepsilon_{\min}\right\}, \tag{19}$$

where $\varepsilon_{\max}$, $\varepsilon_{\min}$ are the largest and smallest length of interest, γ is the decay constant. The equation shows that initially it is the largest length of interest in the input space which allows fewer fuzzy rules to coarsely learn the system and then it decreases exponentially to the smallest length of interest in the input space, which allows more fuzzy rules to finely learn the system.

2.2.2. Allocation of antecedent and consequent parameters

When the new fuzzy rule $N_h + 1$ is added, its corresponding antecedent and consequent parameters are allocated as follows:

$$\begin{cases} a_{N_h+1} = e_n, \\ \mu_{N_h+1} = \mathbf{x}_n, \\ \sigma_{N_h+1} = \kappa\|\mathbf{x}_n - \mu_{nr}\|. \end{cases} \tag{20}$$

2.2.3. Parameter adjustment

In parameter modification, SAFIS utilizes a winner rule strategy similar to the work done by Huang et al. [6]. The key idea of the winner rule strategy is that only the parameters related to the selected winner rule are updated by the EKF algorithm in every step. The 'winner rule' is defined as the rule that is closest (in the Euclidean distance sense) to the current input data as in [6]. As a result, in SAFIS, a fast computation is achieved.

The parameter vector existing in all the fuzzy rules is given by

$$\begin{aligned} \theta_n &= \begin{bmatrix} \theta_1 & \cdots & \theta_{nr} & \cdots & \theta_{N_h} \end{bmatrix}^{\mathrm{T}} \\ &= \begin{bmatrix} a_1, \mu_1, \sigma_1, & \ldots, & a_{nr}, \mu_{nr}, \sigma_{nr}, & \ldots, & a_{N_h}, \mu_{N_h}, \sigma_{N_h} \end{bmatrix}^{\mathrm{T}}, \end{aligned} \tag{21}$$

where $\theta_{nr} = [a_{nr}, \mu_{nr}, \sigma_{nr}]$ is the parameter vector of the nearest fuzzy rule and its gradient is derived as follows:

$$\begin{aligned} \dot{a}_{nr} &= \frac{\partial \hat{\mathbf{y}}_n}{\partial a_{nr}} = \frac{\partial \hat{\mathbf{y}}_n}{\partial R_{nr}} \frac{\partial R_{nr}}{\partial a_{nr}} = \frac{R_{nr}}{\sum_{k=1}^{N_h} R_k}, \\ \dot{\mu}_{nr} &= \frac{\partial \hat{\mathbf{y}}_n}{\partial \mu_{nr}} = \frac{\partial \hat{\mathbf{y}}_n}{\partial R_{nr}} \frac{\partial R_{nr}}{\partial \mu_{nr}} = \frac{a_{nr} - \hat{\mathbf{y}}_n}{\sum_{k=1}^{N_h} R_k} \frac{\partial R_{nr}}{\partial \mu_{nr}}, \\ \dot{\sigma}_{nr} &= \frac{\partial \hat{\mathbf{y}}_n}{\partial \sigma_{nr}} = \frac{\partial \hat{\mathbf{y}}_n}{\partial R_{nr}} \frac{\partial R_{nr}}{\partial \sigma_{nr}} = \frac{a_{nr} - \hat{\mathbf{y}}_n}{\sum_{k=1}^{N_h} R_k} \frac{\partial R_{nr}}{\partial \sigma_{nr}}, \\ \frac{\partial R_{nr}}{\partial \mu_{nr}} &= 2R_{nr} \frac{\mathbf{x}_n - \mu_{nr}}{\sigma_{nr}^2}, \\ \frac{\partial R_{nr}}{\partial \sigma_{nr}} &= 2R_{nr} \frac{\|\mathbf{x}_n - \mu_{nr}\|^2}{\sigma_{nr}^3}. \end{aligned} \tag{22}$$

After obtaining the gradient vector of the nearest fuzzy rule, that is $B_{nr} = [\dot{a}_{nr}, \dot{\mu}_{nr}, \dot{\sigma}_{nr}]^{\mathrm{T}}$, EKF is used to update its parameters as follows:

$$\begin{aligned} K_n &= P_{n-1} B_n [R_n + B_n^{\mathrm{T}} P_{n-1} B_n]^{-1}, \\ \theta_n &= \theta_{n-1} + K_n e_n, \\ P_n &= [I_{Z\times Z} - K_n B_n^{\mathrm{T}}] P_{n-1} + q\, I_{Z\times Z}, \end{aligned} \tag{23}$$

where q is a scalar that determines the allowed step in the direction of the gradient vector, Z is the dimension of parameters to be adjusted. When a new rule is added, the dimension of P_n increases to

$$\begin{pmatrix} P_{n-1} & 0 \\ 0 & p_0 I_{Z_1\times Z_1} \end{pmatrix}, \tag{24}$$

where Z_1 is the dimension of the parameters introduced by the newly added rule, p_0 is an initial value of the uncertainty assigned to the newly allocated rule. In this paper, p_0 is set to 1.0 for all the examples.

2.2.4. Removing of a fuzzy rule

If the influence of rule k is less than a certain pruning threshold e_p, the rule k is insignificant to the output and should be removed. The pruning threshold e_p is chosen a priori. Given the pruning threshold e_p, rule k will be removed if

$$E_{\inf}(k) = |a_k| \frac{(1.8\sigma_k)^{N_x}}{\sum_{k=1}^{N_h} (1.8\sigma_k)^{N_x}} < e_p. \tag{25}$$

Considering the Gaussian function $R(x) = \exp(-x^2/\sigma^2)$, its first and second derivatives will approach zero much faster when x moves away from zero. Thus, in EKF the gradient vector of the parameters for all the rules except the nearest rule will approach zero more quickly than those of the nearest rule:

$$\left(\frac{R_{nr}}{\sum_{k=1}^{N_h} R_k}, \frac{2(a_{nr} - \hat{\mathbf{y}}_n) R_{nr}}{\sum_{k=1}^{N_h} R_k} \frac{\mathbf{x}_n - \mu_{nr}}{\sigma_{nr}^2}, \frac{2(a_{nr} - \hat{\mathbf{y}}_n) R_{nr}}{\sum_{k=1}^{N_h} R_k} \frac{\|\mathbf{x}_n - \mu_{nr}\|^2}{\sigma_{nr}^3} \right).$$

In this case, one may only need to adjust parameters of the nearest rule without adjusting the parameters of all rules when a new observation enters and a new rule need not be added. At the same time, all rules need not be checked for possible pruning. If a new observation arrives and the growing criteria (18) is satisfied, a new rule will be added. The existing rules will maintain their influence because their parameters remain unchanged after learning the new observation. Simultaneously, the newly added rule is also influencing and therefore it is not necessary to check for pruning after a new rule is added. If the growing criteria (18) is not satisfied after a new observation arrives, a new rule will not be added and only the parameters of the nearest rule will be modified. As such only the nearest rule needs to be checked for pruning.

The SAFIS algorithm is summarized below:

Given the growing and pruning thresholds e_g, e_p, for each observation $(\mathbf{x}_n, \mathbf{y}_n)$, where $\mathbf{x}_n \in R^{N_x}$, $\mathbf{y}_n \in R^{N_y}$ and $n = 1, 2, \ldots,$ do

(1) **compute** the overall system output:

$$\hat{\mathbf{y}}_n = \frac{\sum_{k=1}^{N_h} a_k R_k(\mathbf{x}_n)}{\sum_{k=1}^{N_h} R_k(\mathbf{x}_n)},$$

$$R_k(\mathbf{x}_n) = \exp\left(-\frac{1}{\sigma_k^2} \|\mathbf{x}_n - \mu_k\|^2 \right), \tag{26}$$

where N_h is the number of fuzzy rules.

(2) **calculate** the parameters required in the growth criterion:

$$\varepsilon_n = \max\{\varepsilon_{\max} \gamma^n, \varepsilon_{\min}\}, \quad (0 < \gamma < 1),$$

$$e_n = \mathbf{y}_n - \hat{\mathbf{y}}_n \tag{27}$$

(3) **apply** the criterion for adding rules:

If $\|\mathbf{x}_n - \mu_{nr}\| > \varepsilon_n$ and $E_{\inf}(N_h + 1) = |e_n| \frac{(1.8\kappa \|\mathbf{x}_n - \mu_{nr}\|)^{N_x}}{\sum_{k=1}^{N_h+1} (1.8\sigma_k)^{N_x}} > e_g$,

allocate a new rule $N_h + 1$ with

$$a_{N_h+1} = e_n,$$
$$\mu_{N_h+1} = \mathbf{x}_n,$$
$$\sigma_{N_h+1} = \kappa \|\mathbf{x}_n - \mu_{nr}\|. \tag{28}$$

Else

adjust the system parameters $a_{nr}, \mu_{nr}, \sigma_{nr}$ for the nearest rule only by using the EKF method.

Table 1
Effects of parameter e_g on system performance (number of rules and the testing RMS error) under different ε_{max} values and $\kappa = 1.0$

ε_{max}	e_g			
	0.001	0.005	0.01	0.05
1.0	(61, 0.0198)	(17, 0.0385)	(11, 0.0535)	(2, 0.0912)
5.0	(45, 0.0233)	(17, 0.0385)	(11, 0.0535)	(2, 0.0912)
10.0	(41, 0.0249)	(14, 0.0386)	(9, 0.0461)	(2, 0.0912)

Table 2
Effects of parameter e_g on system performance (number of rules and the testing RMS error) under different κ values and $\varepsilon_{max} = 10.0$

κ	e_g			
	0.001	0.005	0.01	0.05
1.0	(41, 0.0249)	(14, 0.0386)	(9, 0.0461)	(2, 0.0912)
1.5	(50, 0.0350)	(18, 0.0586)	(15, 0.0598)	(3, 0.1382)
2.0	(52, 0.0557)	(25, 0.0902)	(15, 0.1384)	(3, 0.1391)

check the criterion for pruning the rule:
If $E_{\inf}(nr) = |a_{nr}| \frac{(1.8\sigma_{nr})^{N_x}}{\sum_{k=1}^{N_h}(1.8\sigma_k)^{N_x}} < e_p$
remove the nrth rule
reduce the dimensionality of EKF
Endif
Endif

2.3. Selection of predefined parameters

In SAFIS, some parameters need to be decided in advance according to the problems considered. They include the distance thresholds (ε_{max}, ε_{min}, γ), the overlap factor (κ) for determining the width of the newly added rule, the growing threshold (e_g) for a new rule and the pruning threshold (e_p) for removing an insignificant rule. A general selection procedure for the predefined parameters is given as follows: ε_{max} is set to around the upper bound of input variables; ε_{min} is set to around 10% of ε_{max}; γ is set to around 0.99. e_p is set to around 10% of e_g. The overlap factor (κ) is utilized to initialize the width of the newly added rule and chosen according to different problems. κ is suggested to be chosen in the range [1.0, 2.0]. The growing threshold e_g is chosen according to the system performance. The smaller the e_g, the better the system performance, but the resulting system structure is more complex.

An example is given to illustrate the effects of the parameters (e_g, κ, ε_{max}) on the system structure and performance. Consider the following two-dimensional sinc function:

$$z = \mathrm{sinc}(x, y) = \frac{\sin(x)\sin(y)}{xy}. \tag{29}$$

In the simulation, 2500 training data pairs (x, y) are drawn from the input range $[-10, 10] \times [-10, 10]$. At the same time, 100 testing data pairs (x, y) are drawn from the same input range.

The general rule for choosing the parameters (ε_{min}, γ, e_p) are obeyed. ε_{min} is set to 10% of ε_{max}; γ is set to 0.997; e_p is set to the 10% of e_g. The parameters e_g, ε_{max} and κ are observed in the range [0.001, 0.05], [1.0, 10.0] and [1.0, 2.0] respectively, to illustrate their effect on the resulting system structure and testing accuracy. Tables 1 and 2 give the effects of parameter e_g on system performance in terms of number of rules and the testing RMS error under different κ or ε_{max} values. From the two tables, it is easy to find that with the increase of e_g the number of rules is decreased and also system performance (testing RMS error) becomes worse with the same κ or ε_{max} value. Furthermore, it can be found from the two tables that the resulting system structure and testing accuracy have does not very big change when the parameter κ or ε_{max} appears as different values.

3. Performance evaluation of SAFIS

In this section, the performance of SAFIS is evaluated based on two nonlinear system identification problems and one chaotic time series (Mackey–Glass) prediction problem. For the first system identification problem, performance of SAFIS is compared with other well known sequential algorithms such as MRAN [20], RANEKF [11], eTS [1], Simpl_eTS [2] and Hybrid Algorithm (HA) [19]. For the second benchmark system identification problem, performance of SAFIS is compared with MRAN [20], RANEKF [11], eTS [1], Simpl_eTS [2] and SONFIN [10]. For the chaotic time series prediction problem, the comparison is done with eTS [1], Simpl_eTS [2], RAN [17], ESOM [5] and DENFIS [12]. In all the studies, the parameters (r, Ω) for eTS and Simpl_eTS are tuned to obtain the best performance.

Performance comparison is done in terms of accuracy and the complexity (the number of rules) of the fuzzy system. For these problems, the SAFIS algorithm goes through the training data sequentially in a single pass and builds up the fuzzy inference system by adding and removing the rules along with their parameters. Then its performance is evaluated on the unseen test data. It should be noted that for other algorithms like HA, ESOM and DENFIS, the training is done over a number of passes and/or requires the complete data for training (batch training).

3.1. Nonlinear dynamic system identification

Narendra and Parthasarathy [16] have suggested two special forms of the nonlinear system model given in Eqs. (30) and (31).

Model I:

$$y(n+1) = f[y(n), y(n-1), \ldots, y(n-k+1)] + \sum_{i=0}^{p-1} \beta_i u(n-i). \tag{30}$$

Model II:

$$y(n+1) = f[y(n), y(n-1), \ldots, y(n-k+1)] + g[u(n), u(n-1), \ldots, u(n-p+1)]. \tag{31}$$

These two models of nonlinear systems have been used here for performance comparison.

Selecting $[y(n), y(n-1), \ldots, y(n-k+1), u(n), u(n-1), \ldots, u(n-p+1)]$ and $y(n+1)$ as the input–output of SAFIS, the identified model is given by the equation

$$\hat{y}(n+1) = \hat{f}(y(n), y(n-1), \ldots, y(n-k+1), u(n), u(n-1), \ldots, u(n-p+1)), \tag{32}$$

where $\hat{f}$ is the SAFIS approximation and $\hat{y}(n+1)$ is the output of the SAFIS.

3.1.1. Identification problem 1

The first nonlinear dynamic system to be identified represents model I and is described by Wang and Yen [19]

$$y(n) = \frac{y(n-1)y(n-2)(y(n-1)-0.5)}{1+y^2(n-1)+y^2(n-2)} + u(n-1). \tag{33}$$

The equilibrium state of the unforced system given by Eq. (33) is (0, 0). As [19], the input $u(n)$ is uniformly selected in the range $[-1.5, 1.5]$ and the test input $u(n)$ is given by $u(n) = \sin(2\pi n/25)$. For the purpose of training and testing 5000 and 200 observation data are produced. The different parameter values for SAFIS are chosen as: $\gamma = 0.997$, $\varepsilon_{\max} = 1.0$, $\varepsilon_{\min} = 0.1$, $\kappa = 1.0$, $e_g = 0.05$, $e_p = 0.005$.

The average performance comparison of SAFIS with MRAN, RANEKF, eTS, Simpl_eTS and HA is shown in Table 3 based on 50 experimental trials. From the table it can be seen that SAFIS obtains similar testing accuracy compared to MRAN, RANEKF, eTS, Simpl_eTS and HA. However, SAFIS achieves this accuracy with smallest number of rules. It is worth noting that HA is based on GA iterative learning and is not sequential. The evolution of the fuzzy rules for SAFIS, MRAN, RANEKF, eTS and Simpl_eTS for a typical run is shown by Fig. 2(a). It can be seen from the figure

Table 3
Results of nonlinear identification problem 1

Methods	No. of rules	Training RMSE	Testing RMSE
SAFIS	17	0.0539	0.0221
MRAN	22	0.0371	0.0271
RANEKF	35	0.0273	0.0297
Simpl_eTS ($r = 2.0$, $\Omega = 10^6$)	22	0.0528	0.0225
eTS ($r = 1.8$, $\Omega = 10^6$)	49	0.0292	0.0212
HA [19]	28	0.0182	0.0244

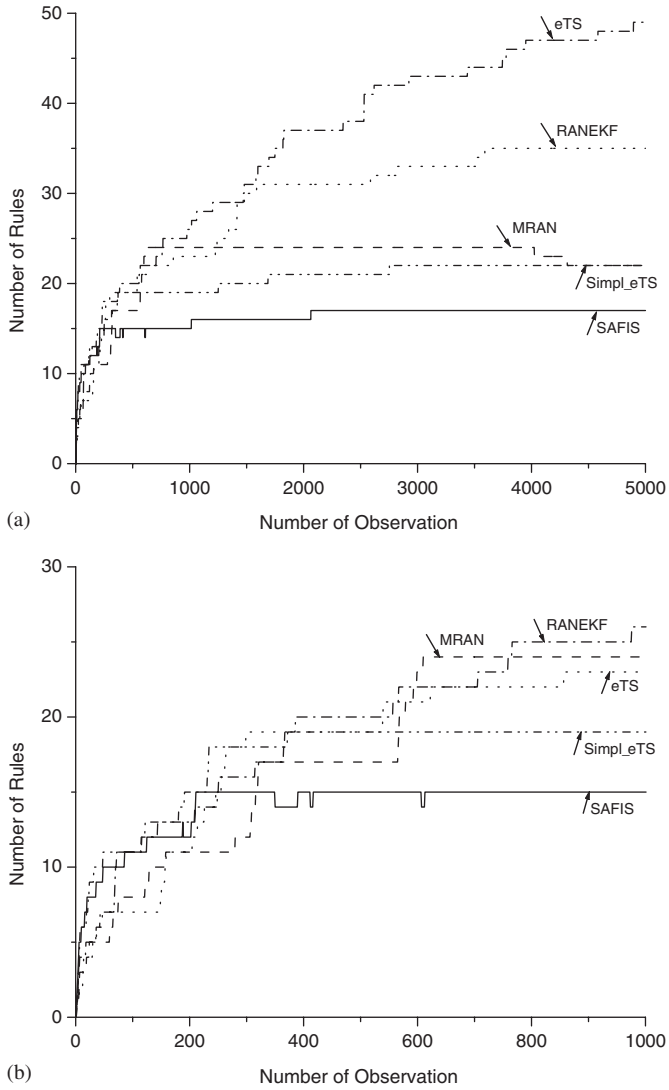

Fig. 2. (a) Rule update process between different algorithms for nonlinear identification problem 1 during the whole observation; (b) Rule update process between different algorithms for nonlinear identification problem 1 between 0 and 1000 observations.

that SAFIS produces least number of rules. Besides, Fig. 2(b) gives a clear illustration for the rule evolution tendency between 0 and 1000 observation and shows that SAFIS can automatically add and delete a rule during learning, which is manifested by increasing and reducing the number of rules by one.

Table 4
Results of nonlinear identification problem 2

Methods	No. of rules	Testing RMSE
SAFIS	8	0.0116
MRAN	10	0.0129
RANEKF	11	0.0184
Simpl_eTS ($r = 0.075$, $\Omega = 10^6$)	18	0.0122
eTS ($r = 1.0$, $\Omega = 10^6$)	19	0.0082
SONFIN [10]	10	0.0130

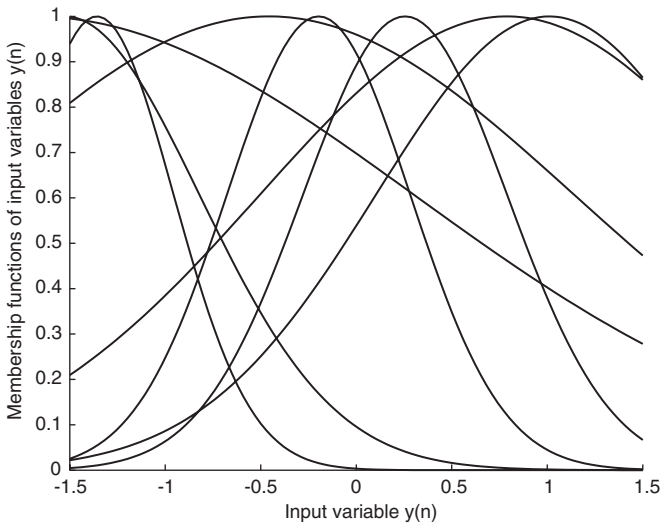

Fig. 3. Membership functions of input variable $y(n)$ for nonlinear identification problem 2.

3.1.2. Identification problem 2

The second nonlinear dynamic system to be identified represents model II and is described by Juang and Lin [10]

$$y(n+1) = \frac{y(n)}{1+y^2(n)} + u^3(n). \tag{34}$$

In accordance with [10], the input signal $u(n)$ is given by $\sin(2\pi n/100)$. For the purpose of training and testing 50 000 and 200 observation data are produced. The SAFIS parameter values chosen are: $\gamma = 0.997$, $\varepsilon_{\max} = 2.0$, $\varepsilon_{\min} = 0.2$, $\kappa = 2.0$, $e_g = 0.03$, $e_p = 0.003$. The input variables $y(n)$, $u(n)$, respectively, follow the uniform sample distribution in the range $[-1.5, 1.5]$ and $[-1.0, 1.0]$.

Table 4 shows the performance comparison of SAFIS with MRAN, RANEKF, eTS, Simpl_eTS and SONFIN [10]. It can be seen from the table that SAFIS achieves similar accuracy with a lesser number of rules. Figs. 3 and 4 show the final membership functions of input variables $y(n)$, $u(n)$ achieved by SAFIS. From the two figures, one can clearly see that the input variable membership functions are distributed in their own entire range. Besides the testing accuracy of SAFIS is slightly better than those of MRAN, RANEKF, Simpl_eTS and SONFIN, which verifies that the learning performance of SAFIS is not lost by only modifying the nearest fuzzy rule instead of all fuzzy rules during the learning.

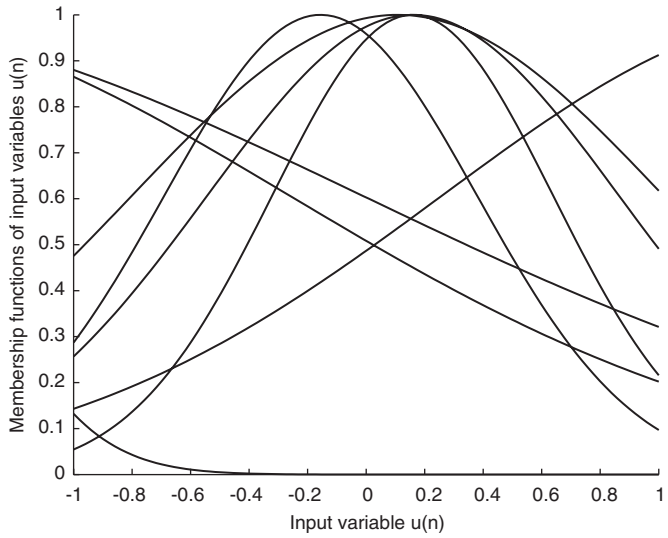

Fig. 4. Membership functions of input variable $u(n)$ for nonlinear identification problem 2.

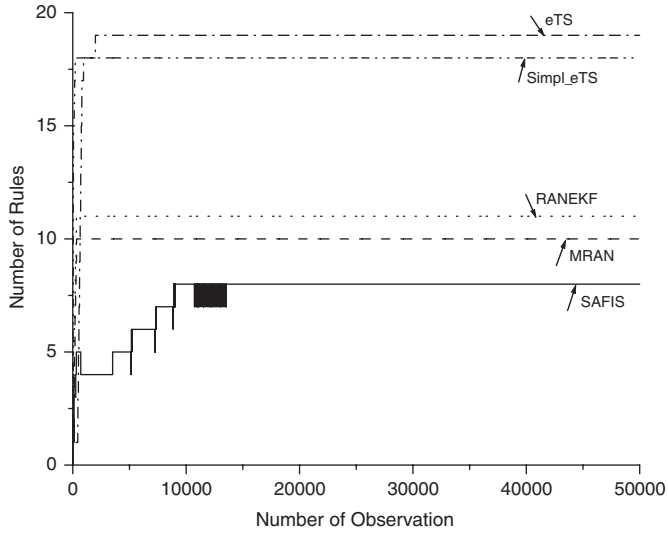

Fig. 5. Rule update process between different algorithms for nonlinear identification problem 2.

The evolution of the fuzzy rules for SAFIS, MRAN, RANEKF, eTS and Simpl_eTS is shown by Fig. 5. It can be seen from the figure that SAFIS is able to add and delete rules during learning and produces least number of rules.

3.2. Mackey–Glass time series prediction

The chaotic Mackey–Glass time series is generated from the following differential equation [12]:

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t), \tag{35}$$

where $\tau = 17$ and $x(0) = 1.2$. For the purpose of training and testing, 6000 samples are produced by means of the fourth-order Runge–Kutta method with the step size 0.1. The prediction task is to predict the value $x(t + 85)$ from

Table 5
Results of Mackey–Glass time series prediction (without normalizing inputs) using truly sequential learning algorithms

Methods	No. of rules	Testing NDEI
SAFIS	21	0.380
Simpl_eTS($r = 0.292$, $\Omega = 10^6$)	21	0.376
eTS($r = 0.65$, $\Omega = 10^6$)	99	0.356
RAN [17]	113	0.373

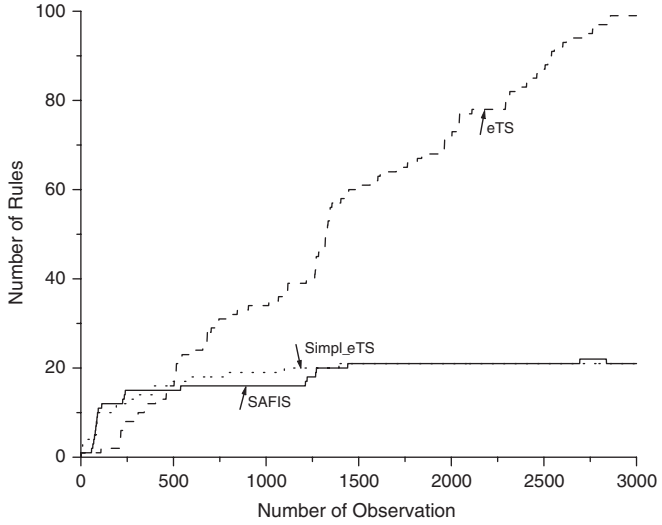

Fig. 6. Rule update process between different algorithms for Mackey–Glass time series prediction (without normalizing inputs) with a single pass.

the input vector $[x(t-18)\ \ x(t-12)\ \ x(t-6)\ \ x(t)]$ for any value of the time t. In order to compare with the results of [12], the observations between $t = 201$ and 3200 and the observations between $t = 5001$ and 5500 are extracted from the series and used as training and testing data. For this problem, the parameters for SAFIS are selected as follows: $\gamma = 0.98$, $\varepsilon_{\max} = 1.6$, $\varepsilon_{\min} = 0.16$, $\kappa = 1.68$, $e_g = 0.0005$, $e_p = 0.00005$. The data follow a uniform sample distribution in the range [0.4, 1.4].

Table 5 shows the prediction accuracies and the number of rules obtained by SAFIS, eTS, Simpl_eTS, and RAN without normalizing inputs. For comparison purposes, the prediction accuracy is based on the Non-Dimensional Error Index (NDEI) defined as the RMSE divided by the standard deviation of the true output values. From the table, it is clear that all the algorithms produce similar accuracies. But SAFIS and Simpl_eTS achieve this with the smallest number of fuzzy rules. The evolution of the fuzzy rules for SAFIS, eTS and Simpl_eTS for a single pass is shown in Fig. 6.

It should be noted that both DENFIS and ESOM required the training data to be used several times (more number of passes) and also need the complete data (not sequential). In order to compare with ESOM and DENFIS, the three algorithm SAFIS, eTS, Simpl_eTS have also been run with multiple passes and the results are shown in Table 6. It can be seen from Table 6, all the algorithms produce comparable prediction accuracies but SAFIS obtains this with the smallest number of rules.

In order to compare with the result of Angelov et al. [2,3] where the inputs are normalized, we have also normalized the inputs and show the results in Table 7 and Fig. 7. As observed from Table 7, all the algorithms produce similar accuracies; however, SAFIS obtains the smallest number of fuzzy rules. The evolution of the fuzzy rules for SAFIS, eTS and Simpl_eTS for a single pass is shown in Fig. 7.

Table 6
Results of Mackey–Glass time series prediction (without normalizing inputs) using multi-pass learning algorithms

Methods	No. of rules	Testing NDEI
SAFIS (10 pass)	45	0.271
Simpl_eTS (10 pass) ($r = 0.292$, $\Omega = 10^6$)	50	0.325
eTS (10 pass) ($r = 0.65$, $\Omega = 10^6$)	99	0.202
ESOM [5]	114	0.320
DENFIS [12]	58	0.276

Table 7
Results of Mackey–Glass time series prediction (with normalized inputs) using truly sequential learning algorithms

Methods	No. of rules	Testing NDEI
SAFIS	6	0.376
Simpl_eTS ($r = 0.25$, $\Omega = 750$) [2]	11	0.394
eTS ($r = 0.25$, $\Omega = 750$) [3]	9	0.380

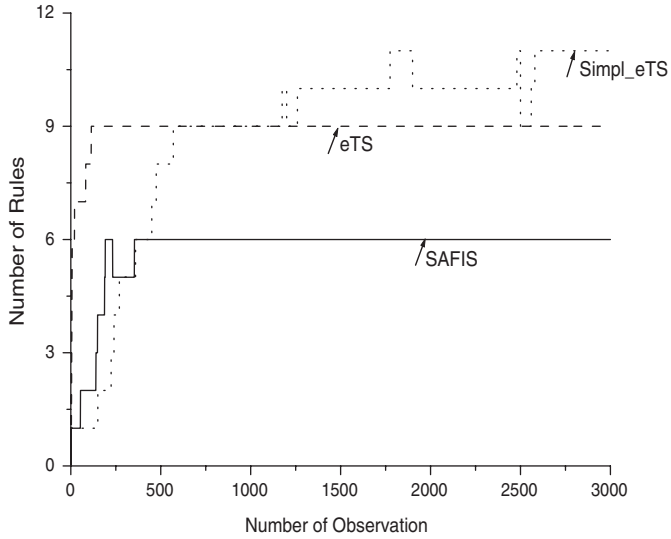

Fig. 7. Rule update process between different algorithms for Mackey–Glass time series prediction (with normalized inputs) with a single pass.

4. Conclusions

In this paper, a sequential fuzzy inference system called SAFIS is developed based on the functional equivalence between a RBF (in this case, GAP–RBF network) and a FIS. In SAFIS, the fuzzy rules are added or removed based on the concept of "influence" of the rule. When there are no additions, only the parameters of the "closest" rule is updated using an EKF scheme. SAFIS is truly a sequential learning algorithm and produces a compact fuzzy inference system.

The performance of SAFIS has been compared with several existing algorithms on two nonlinear systems identification benchmark problems and a chaotic time series prediction problem. Results indicate that SAFIS produces similar or better accuracies with less number of rules compared to other algorithms.

Acknowledgements

We wish to thank reviewers for all the constructive comments and suggestions. We also wish to thank P. Angelov for providing the details of eTS and Simpl_eTS and also the source codes of eTS.

References

[1] P.P. Angelov, D.P. Filev, An approach to online identification of Takagi–Sugeno fuzzy models, IEEE Trans. Systems Man Cybernet.—Part B: Cybernet. 34 (1) (2004) 484–498.
[2] P. Angelov, D. Filev, Simpl_eTS: a simplified method for learning evolving Takagi–Sugeno fuzzy models, in: The 14th IEEE Internat. Conf. on Fuzzy Systems (2005) 1068–1073.
[3] P. Angelov, J. Victor, A. Dourado, D. Filev, On-line evolution of Takagi–Sugeno fuzzy models, in: Proc. Second IFAC Workshop on Advanced Fuzzy/Neural Control, Oulu, Finland, 2004, pp. 67–72.
[4] K.B. Cho, B.H. Wang, Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction, Fuzzy Sets and Systems 83 (1996) 325–339.
[5] D. Deng, N. Kasabov, Evolving self-organizing maps for online learning, data analysis and modeling, in: S.-I. Amari, C.L. Giles, M. Gori, V. Piuri (Eds.), Proc. IJCNN' 2000 on Neural Networks Neural Computing: New Challenges Perspectives New Millennium, vol. VI, IEEE Press, New York, 2000, pp. 3–8.
[6] G.-B. Huang, P. Saratchandran, N. Sundararajan, An efficient sequential learning algorithm for growing and pruning RBF (GAP–RBF) networks, IEEE Trans. Systems Man Cybernet.—Part B: Cybernet. 34 (6) (2004) 2284–2292.
[7] G.-B. Huang, P. Saratchandran, N. Sundararajan, A generalized growing and pruning RBF (GGAP–RBF) neural network for function approximation, IEEE Trans. Neural Networks 16 (1) (2005) 57–67.
[8] J.-S.R. Jang, C.-T. Sun, Functional equivalence between radial basis function networks and fuzzy inference systems, IEEE Trans. Neural Networks 4 (1) (1993) 156–159.
[9] J.-S.R. Jang, C.-T. Sun, E. Mizutani, Neuro-Fuzzy and Soft Computing: a Computational Approach to Learning and Machine Intelligence, Prentice-Hall, New York, 1997.
[10] C.-F. Juang, C.-T. Lin, An on-line self-constructing neural fuzzy inference network and its applications, IEEE Trans. Fuzzy Systems 10 (2) (2002) 144–154.
[11] V. Kadirkamanathan, M. Niranjan, A function estimation approach to sequential learning with neural networks, Neural Comput. 5 (6) (1993) 954–975.
[12] N. Kasabov, Q. Song, DENFIS: dynamic evolving neural-fuzzy inference system and its application for time series prediction, IEEE Trans. Fuzzy Systems 10 (2) (2002) 144–154.
[13] G. Leng, T.M. McGinnity, G. Prasad, An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network, Fuzzy Sets and Systems 150 (2) (2005) 211–243.
[14] C.W. Lee, Y.C. Shin, Construction of fuzzy systems using least-squares method and genetic algorithm, Fuzzy Sets and Systems 137 (2003) 297–323.
[15] S. Mitra, Y. Hayashi, Neuro-fuzy rule generation: survey in soft computing framework, IEEE Trans. Neural Networks 11 (3) (2000) 748–768.
[16] K.S. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Trans. Neural Networks 1 (1) (1990) 4–27.
[17] J. Platt, A resource allocating network for function interpolation, Neural Comput. 3 (2) (1991) 213–225.
[18] S. Wu, M.J. Er, Dynamic fuzzy neural networks—a novel approach to function approximation, IEEE Trans. Systems Man Cybernet.—Part B: Cybernet. 30 (2) (2000) 358–364.
[19] L. Wang, J. Yen, Extracting fuzzy rules for system modeling using a hybrid of genetic algorithm and Kalman filter, Fuzzy Sets and Systems 101 (1999) 353–362.
[20] L. Yingwei, N. Sundararajan, P. Saratchandran, A sequential learning scheme for function approximation using minimal radial basis function (RBF) neural networks, Neural Comput. 9 (1997) 461–478.