Chapter 10
Sequential Adaptive Fuzzy Inference System for Function Approximation Problems

Hai-Jun Rong

Abstract In the classic approaches to design a fuzzy inference system, the fuzzy rules are determined by a domain expert a priori and then they are maintained unchanged during the learning. These fixed fuzzy rules may not be appropriate in real-time applications where the environment or model often meets unpredicted disturbances or damages. Hence, poor performance may be observed. In comparison to the conventional methods, fuzzy inference systems based on neural networks, called fuzzy-neural systems, have begun to exhibit great potential for adapting to the changes by utilizing the learning ability and adaptive capability of neural networks. Thus, a fuzzy inference system can be built using the standard structure of neural networks. Nevertheless, the determination of the number of fuzzy rules and the adjustment of the parameters in the if-then fuzzy rules are still open issues. A sequential adaptive fuzzy inference system (SAFIS) is developed to determine the number of fuzzy rules during learning and modify the parameters in fuzzy rules simultaneously. SAFIS uses the concept of influence of a fuzzy rule for adding and removing rules during learning. The influence of a fuzzy rule is defined as its contribution to the system output in a statistical sense when the input data is uniformly distributed. When there is no addition of fuzzy rules, 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 evaluated based on some function approximation problems, via, nonlinear system identification problems and a chaotic time-series prediction problem. Results indicate that SAFIS produces similar or better accuracies with lesser number of rules compared to other algorithms.

H.-J. Rong (✉)
State Key Laboratory of Strength and Vibration, School of Aerospace,
Xi'an Jiaotong University, No.28 Xianning West Road, Xi'an, ShaanXi, China
e-mail: hjrong@mail.xjtu.edu.cn

M. Sayed-Mouchaweh and E. Lughofer (eds.), *Learning in Non-Stationary Environments: Methods and Applications*, DOI 10.1007/978-1-4419-8020-5_10,
© Springer Science+Business Media New York 2012

10.1 Introduction

A fuzzy inference system can model the qualitative aspects of human knowledge and reasoning processes using fuzzy if-then rules. Based on this, many ill-defined and uncertain systems in some disciplines such as engineering, economics, and other areas [6, 10, 16, 18, 20, 23, 24, 29, 31] can be handled without employing precise quantitative analysis. The experiment results have demonstrated that the fuzzy inference systems are very useful to solve some practical problems involving a high level of uncertainty, complexity, or nonlinearity compared with the conventional modeling methods. However, the superior performance of fuzzy inference systems mainly depends on the fuzzy rules. If the fuzzy rules are not appropriate and deviate from the requirement of the system itself, this may result in poor performance. On the other hand, although the rules are correct, it is hard to determine the appropriate parameters for the fuzzy rules. Inappropriate parameters also may result in poor performance.

To solve these problems, many researchers have built fuzzy-neural systems by incorporating the fuzzy inference process in the structure of neural networks and then the learning ability of neural networks are used to adjust the fuzzy rules. Except for some special fuzzy-neural systems which made use of fuzzy neurons and fuzzy weights [7, 25], most of the recent fuzzy-neural systems [1, 3, 5, 11, 13, 17, 19, 32] have been built based on the standard feed-forward network with local fields to approximate the fuzzy inference systems with local properties. In these fuzzy-neural systems, the neurons with local fields correspond to the fuzzy rules and the proposed algorithms for designing the fuzzy-neural systems have considered two issues, that is, 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 task of parameter adjustment involves realizing the parameters for the fuzzy system structure determined in the previous step [21].

The researchers [5, 13, 15, 19, 27, 28, 30, 32] have tried to develop many efficient approaches for solving the two issues. 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 is available before the training commences. The training usually involves cycling the data over a number of epochs. In sequential learning, the data arrives one by 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 arrives 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. The sequential fuzzy-neural scheme, which is discussed in this chapter, has the following distinguishing features:

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 existing 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.

Jang [11] has developed an adaptive-network-based fuzzy inference system (ANFIS) where a hybrid learning method was utilized to identify the system parameters. The parameters in the membership functions were updated by a gradient descent method, and the parameters in the consequent parts were adjusted by means of a least-square error method. The number of fuzzy rules was determined according to a grid-type partition which resulted in the exponential increase of the number of fuzzy rules as the input variables increased. Chiu [4] solved this problem by selecting some significant input variables from all the input variables as the input of the fuzzy systems. However, these algorithms require cycling the whole training data over a number of learning cycles (epochs). Thus, they are batch learning algorithms. Besides, in these algorithms, the number of fuzzy rules are determined beforehand and cannot be varied according to learning process.

Many approaches [5, 13, 19, 32] have been proposed based on the functional equivalence between a radial basis function (RBF) neural network and a fuzzy inference system to achieve the determination of the number of fuzzy rules and parameter adjustment simultaneously during learning. 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 [12]. A significant contribution to sequential learning in RBF network was made by Platt [26] 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 [5] automatically generated fuzzy rules without predefining the number of fuzzy rules based on the error and distance criterion of fuzzy basis functions. The parameters in the fuzzy rules were modified by the gradient descent algorithm. 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.

Juang and Lin [13] have proposed a self-constructing neural fuzzy inference network (SONFIN) in which the fuzzy rules were extracted online from the training data together with the parameter update for all existing fuzzy rules using the gradient descent method. For adding a new fuzzy rule, SONFIN utilized 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 [33], a hierarchical on-line self-organizing learning algorithm for dynamic fuzzy neural networks (DFNN) has been proposed in [32]. Another on-line self-organizing fuzzy neural network (SOFNN) proposed by Leng et al. [19] also included a pruning method. The pruning method utilized the optimal brain surgeon (OBS) approach to determine the importance of each rule. In the two algorithms, the least-square error method was utilized to update the parameters for all the existing fuzzy rules. 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.

A dynamic evolving neural-fuzzy inference system (DENFIS) was proposed by Kasabov and Song [17] where the fuzzy rules were created depending on the position of the input vector in the input space and the output was dynamically calculated based on *m*-most active fuzzy rules which have been created during the past learning process. Angelov and Filev [3] proposed an evolving Takagi–Sugeno model (eTS) that recursively updated 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 was added when the potential of the new data was higher than the potential of the existing rules, or a new rule was modified when the potential of the new data was higher than the potential of the existing rules and the new data was close to an old rule. These two algorithms are truly sequential learning algorithms. However, the algorithms 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 simplified the rule base, called the simpl_eTS, was proposed by Angelov and Filev [1]. The algorithm utilized the concept of the scatter which was similar to the notion of potential but computationally more efficient. The algorithm could 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 belonged to a particular cluster. If the population of a rule was less than 1% of the total data at the moment of appearance of a rule, the rule was ignored from the rule base by setting its firing strength to zero. Besides, these algorithms employed the least-square error method to modify the parameters of the existing fuzzy rules.

In this chapter, a sequential adaptive fuzzy inference system (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 growing and pruning RBF (GAP-RBF)

neural network proposed by Huang et al. [8]. 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 is 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 is uniformly distributed. The parameter adjustment is done using a winner rule strategy where the winner rule is defined as the one closest to the latest input data, and the parameter update is done using an extended Kalman filter (EKF) mechanism.

10.2 Architecture of SAFIS

A function approximation problem can be described as follows. Suppose the sample data, $\{(\mathbf{x}_n, \mathbf{y}_n) : n = 1, 2, \ldots\}$, are observed, where $\mathbf{x}_n$ is a N_x-dimensional features of observation n and $\mathbf{y}_n$ is its target output of dimension N_y. It is assumed that the observation data are free of noise, and an underlying function f exists between the target output $\mathbf{y}_n$ and feature space $\mathbf{x}_n$ from the known set of data:,

$$\mathbf{y}_n = f(\mathbf{x}_n). \tag{10.1}$$

The aim of the SAFIS algorithm is to approximate f such that:

$$\hat{\mathbf{y}}_n = \hat{f}(\mathbf{x}_n), \tag{10.2}$$

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. 10.1 consists of five layers to realize the following fuzzy rule model:

Rule k : if $(x_1$ is $A_{1k})$ ... $(x_{N_x}$ is $A_{N_x k})$, then $(\hat{y}_1$ is $a_{1k})$ ... $(\hat{y}_{N_y}$ 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 $\mathbf{x}(\mathbf{x} = [x_1, \ldots, x_{N_x}]^T)$, N_h is the number of fuzzy rules, and N_y is the dimension of the output vector $\hat{\mathbf{y}}(\hat{\mathbf{y}} = [\hat{y}_1, \ldots, \hat{y}_{N_y}]^T)$. 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.

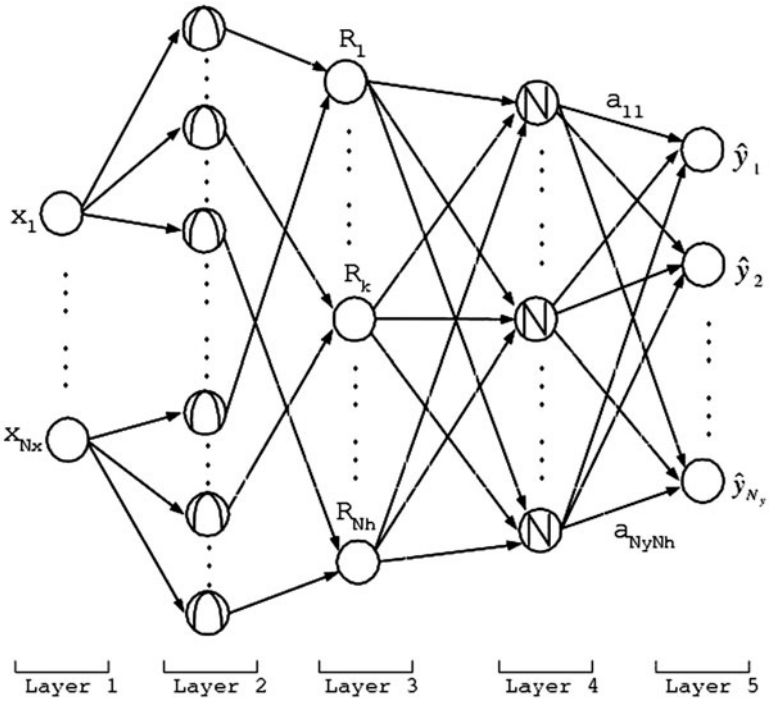

Fig. 10.1 Structure of SAFIS

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 function is given by:

$$A_{ik}(x_i) = \exp\left(-\frac{(x_i - \mu_{ik})^2}{\sigma_k^2}\right), k = 1, 2, \ldots, N_h, \tag{10.3}$$

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, and 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:

$$R_k(\mathbf{x}) = \prod_{i=1}^{N_x} A_{ik}(x_i) = \exp\left(-\sum_{i=1}^{N_x} \frac{(x_i - \mu_{ik})^2}{\sigma_k^2}\right) = \exp\left(-\frac{\|\mathbf{x} - \mu_k\|^2}{\sigma_k^2}\right). \tag{10.4}$$

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:

$$\bar{R}_k = \frac{R_k(\mathbf{x})}{\sum_{k=1}^{N_h} R_k(\mathbf{x})}. \tag{10.5}$$

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:

$$\hat{\mathbf{y}} = \frac{\sum_{k=1}^{N_h} R_k(\mathbf{x})\mathbf{a}_k}{\sum_{k=1}^{N_h} R_k(\mathbf{x})}, \tag{10.6}$$

where $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{N_y}]^T, \mathbf{a}_k = [a_{k1}, a_{k2}, \ldots, a_{kN_y}]^T$.

Similar to the significance concept of a neuron in GAP-RBF [8], the SAFIS algorithm uses the concept of "influence" of a rule to realize the growing and pruning of fuzzy rules. It is described below.

10.2.1 "Influence" of a Fuzzy Rule

As per (10.6), the contribution of the kth rule to the overall output for an input observation $\mathbf{x}_l$ is given by:

$$E(k,l) = \|\mathbf{a}_k\| \frac{R_k(\mathbf{x}_l)}{\sum_{k=1}^{N_h} R_k(\mathbf{x}_l)}. \tag{10.7}$$

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

$$E(k) = \|\mathbf{a}_k\| \frac{\sum_{l=1}^{N} R_k(\mathbf{x}_l)}{\sum_{k=1}^{N_h}\sum_{l=1}^{N} R_k(\mathbf{x}_l)}. \tag{10.8}$$

Dividing both the numerator and denominator by N in (10.8), the equation becomes:

$$E(k) = \|\mathbf{a}_k\| \frac{\sum_{l=1}^{N} R_k(\mathbf{x}_l)/N}{\sum_{k=1}^{N_h}\sum_{l=1}^{N} R_k(\mathbf{x}_l)/N}. \tag{10.9}$$

Using the significance concept of GAP-RBF [8], 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:

$$E_{\inf}(k) = \lim_{N\to\infty} E(k) = \lim_{N\to\infty} \|\mathbf{a}_k\| \frac{\sum_{l=1}^{N} R_k(\mathbf{x}_l)/N}{\sum_{k=1}^{N_h}\sum_{l=1}^{N} R_k(\mathbf{x}_l)/N}. \tag{10.10}$$

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 [9]. In order to compute $E_{\inf}(k)$, one has to compute first E_k defined by:

$$E_k = \lim_{N\to\infty} \frac{\sum_{l=1}^{N} R_k(\mathbf{x}_l)}{N}. \tag{10.11}$$

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 $\Delta_j, j = 1, \ldots, M$. 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 and Δ_j is small from (10.11), we have:

$$\begin{aligned} E_k &\approx \lim_{M\to\infty} \frac{\sum_{j=1}^{M} R_k(\mathbf{x}_j) \cdot Np(\mathbf{x}_j) \cdot S(\Delta_j)}{N} \\ &= \lim_{M\to\infty} \sum_{j=1}^{M} R_k(\mathbf{x}_j) \cdot p(\mathbf{x}_j) \cdot S(\Delta_j) \\ &= \int_X R_k(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \\ &= \int_X \exp\left(-\frac{\|\mathbf{x} - \mu_k\|^2}{\sigma_k^2}\right) p(\mathbf{x}) d\mathbf{x}. \end{aligned} \tag{10.12}$$

If the distribution of the N_x attributes $(x_1, \ldots, x_i, \ldots, x_{N_x})^T$ of observations $\mathbf{x}$'s are independent from each other, the density function $p(\mathbf{x})$ of $\mathbf{x}$ can be written as follows: $p(\mathbf{x}) = \prod_{i=1}^{N_x} p_i(x_i)$, where $p_i(x)$ is the density function of the i-th attribute x_i of observations. In this case, (10.12) can be re-written as:

$$\begin{aligned} E_k &= \int \cdots \int_X \exp\left(-\frac{\|\mathbf{x} - \mu_k\|^2}{\sigma_k^2}\right) p(\mathbf{x}) \mathrm{d}\mathbf{x} \\ &= \prod_{i=1}^{N_x} \left(\int_{a_i}^{b_i} \exp\left(-\frac{\|x - \mu_{k,i}\|^2}{\sigma_k^2}\right) p_i(x) \mathrm{d}x \right), \end{aligned} \tag{10.13}$$

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

Equation (10.13) involves the 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 $p(\mathbf{x}) = \frac{1}{S(X)}$, 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 (10.12), we get:

$$E_k = \int_X \exp\left(-\frac{\|\mathbf{x} - \mu_k\|^2}{\sigma_k^2}\right) \frac{1}{S(X)} \mathrm{d}\mathbf{x}. \tag{10.14}$$

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:

$$\begin{aligned} E_k &\approx \frac{1}{S(X)} \left(2 \int_0^{+\infty} \exp\left(-\frac{x^2}{\sigma_k^2}\right) \mathrm{d}x\right)^{N_x} \\ &= \frac{\pi^{N_x/2} \sigma_k^{N_x}}{S(X)} \\ &= \frac{(1.8\sigma_k)^{N_x}}{S(X)}. \end{aligned} \tag{10.15}$$

Thus, based on (10.15), the influence of the kth rule is given by:

$$E_{\inf}(k) = \|\mathbf{a}_k\| \frac{(1.8\sigma_k)^{N_x}}{\sum_{k=1}^{N_h} (1.8\sigma_k)^{N_x}}. \tag{10.16}$$

It is noteworthy that the significance of a neuron proposed in GAP-RBF [8] 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 here is different from the significance of a neuron proposed in GAP-RBF [8]. 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. Seen from (10.16), with the introduction of influence, one need not estimate the input distribution range S(X) and the implementation has been simplified.

Influence of a rule is utilized for the addition and deletion of a fuzzy rule in SAFIS algorithm as indicated below.

10.2.2 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 [8] 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.

10.2.2.1 Adding of Fuzzy Rules

SAFIS begins with no fuzzy rules. When the first input $\mathbf{x}_1, \mathbf{y}_1$ is received, it is translated into the first rule whose parameters are given as, $\mu_1 = \mathbf{x}_1, \mathbf{a}_1 = \mathbf{y}_1, \sigma_1 = \kappa\|\mathbf{x}_1\|$. Then, as inputs $\mathbf{x}_n, \mathbf{y}_n$ ($n > 1$ 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 $N_h + 1$:

$$\begin{aligned} &\|\mathbf{x}_n - \mu_{nr}\| > \varepsilon_n \\ &E_{\inf}(N_h+1) = \|\mathbf{e}_n\| \frac{(1.8\kappa\|\mathbf{x}_n - \mu_{nr}\|)^{N_x}}{\sum_{k=1}^{N_h+1} (1.8\sigma_k)^{N_x}} > e_{\mathrm{g}}, \end{aligned} \tag{10.17}$$

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 $\mathbf{x}_n$, and e_{g} is the growing threshold and is chosen according to the desired accuracy of SAFIS. $\mathbf{e}_n = \mathbf{y}_n - \hat{\mathbf{y}}_n$, $\mathbf{y}_n$ 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, and ε_n is the distance threshold which decays exponentially and is given by:

$$\varepsilon_n = \max\left\{\varepsilon_{\max} \times \gamma^{\mathrm{n}}, \varepsilon_{\min}\right\}, \tag{10.18}$$

where $\varepsilon_{\max}, \varepsilon_{\min}$ are the largest and smallest length of interest and 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.

10.2.2.2 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:

$$\begin{cases} \mathbf{a}_{N_h+1} &= \mathbf{e}_n \\ \mu_{N_h+1} &= \mathbf{x}_n \\ \sigma_{N_h+1} &= \kappa\|\mathbf{x}_n - \mu_{nr}\| \end{cases} \tag{10.19}$$

10.2.2.3 Parameter Adjustment

In parameter modification, SAFIS utilizes a winner rule strategy similar to the work done by Huang et al. [8]. 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 [8]. As a result, a fast computation is achieved in SAFIS.

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

$$\begin{aligned}\theta_n &= \left[\, \theta_1 \ \ldots\ \theta_{nr} \ \ldots\ \theta_{N_h} \,\right]^T \\ &= \left[\, \mathbf{a}_1, \mu_1, \sigma_1, \ \ldots,\ \mathbf{a}_{nr}, \mu_{nr}, \sigma_{nr}, \ \ldots,\ \mathbf{a}_{N_h}, \mu_{N_h}, \sigma_{N_h} \,\right]^T, \end{aligned} \tag{10.20}$$

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

$$\begin{aligned}
\dot{\mathbf{a}}_{nr} &= \frac{\partial \hat{\mathbf{y}}_n}{\partial \mathbf{a}_{nr}} = \frac{\partial \hat{\mathbf{y}}_n}{\partial R_{nr}} \frac{\partial R_{nr}}{\partial \mathbf{a}_{nr}} = \frac{R_{nr}}{\sum_{k=1}^{N_h} R_k} \\
\dot{\mu}_{nr} &= \frac{\partial \hat{\mathbf{y}}_n}{\partial \mu_{nr}} = \frac{\partial \hat{\mathbf{y}}_n}{\partial R_{nr}} \frac{\partial R_{nr}}{\partial \mu_{nr}} = \frac{\mathbf{a}_{nr} - \hat{\mathbf{y}}_n}{\sum_{k=1}^{N_h} R_k} \frac{\partial R_{nr}}{\partial \mu_{nr}} \\
\dot{\sigma}_{nr} &= \frac{\partial \hat{\mathbf{y}}_n}{\partial \sigma_{nr}} = \frac{\partial \hat{\mathbf{y}}_n}{\partial R_{nr}} \frac{\partial R_{nr}}{\partial \sigma_{nr}} = \frac{\mathbf{a}_{nr} - \hat{\mathbf{y}}_n}{\sum_{k=1}^{N_h} R_k} \frac{\partial R_{nr}}{\partial \sigma_{nr}} \\
\frac{\partial R_{nr}}{\partial \mu_{nr}} &= 2R_{nr} \frac{\mathbf{x}_n - \mu_{nr}}{\sigma_{nr}^2} \\
\frac{\partial R_{nr}}{\partial \sigma_{nr}} &= 2R_{nr} \frac{\|\mathbf{x}_n - \mu_{nr}\|^2}{\sigma_{nr}^3}.
\end{aligned} \tag{10.21}$$

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

$$\begin{aligned}
\mathbf{K}_n &= \mathbf{P}_{n-1}\mathbf{B}_n[\mathbf{R}_n + \mathbf{B}_n^T \mathbf{P}_{n-1}\mathbf{B}_n]^{-1} \\
\theta_n &= \theta_{n-1} + \mathbf{K}_n \mathbf{e}_n \\
\mathbf{P}_n &= [\mathbf{I}_{Z\times Z} - \mathbf{K}_n \mathbf{B}_n^T]\mathbf{P}_{n-1} + q\mathbf{I}_{Z\times Z},
\end{aligned} \tag{10.22}$$

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

$$\begin{pmatrix} \mathbf{P}_{n-1} & 0 \\ 0 & p_0 \mathbf{I}_{Z_1 \times Z_1} \end{pmatrix}, \tag{10.23}$$

where Z_1 is the dimension of the parameters introduced by the newly added rule and p_0 is an initial value of the uncertainty assigned to the newly allocated rule.

10.2.2.4 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:

$$E_{\inf}(k) = \|\mathbf{a}_k\| \frac{(1.8\sigma_k)^{N_x}}{\sum_{k=1}^{N_h} (1.8\sigma_k)^{N_x}} < e_\mathrm{p}. \tag{10.24}$$

In SAFIS, only the nearest rule instead of all the existing rules will be considered for removing. This is explained as follows. Considering the Gaussian function $R(x) = \exp(-\frac{x^2}{\sigma^2})$, 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 that are given by:

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

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 needs 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 (10.17) 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 (10.17) 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:

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

$$R_k(\mathbf{x}_n) = \exp\left(-\frac{1}{\sigma_k^2}\|\mathbf{x}_n - \mu_k\|^2\right) \tag{10.25}$$

where N_h is the number of fuzzy rules.

2. **calculate** the parameters required in the growth criterion:

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

$$\mathbf{e}_n = \mathbf{y}_n - \hat{\mathbf{y}}_n \tag{10.26}$$

3. **apply** the criterion for adding rules:

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

allocate a new rule $N_h + 1$ with

$$\mathbf{a}_{N_h+1} = \mathbf{e}_n$$

$$\mu_{N_h+1} = \mathbf{x}_n$$

$$\sigma_{N_h+1} = \kappa\|\mathbf{x}_n - \mu_{nr}\| \tag{10.27}$$

Else

adjust the system parameters $\mathbf{a}_{nr}$, μ_{nr}, σ_{nr} for the nearest rule only by using the EKF method.

check the criterion for pruning the rule:

If $E_{\inf}(nr) = \|\mathbf{a}_{nr}\| \frac{(1.8\sigma_{nr})^{N_x}}{\sum_{k=1}^{N_h}(1.8\sigma_k)^{N_x}} < e_p$

remove the nr-th rule

reduce the dimensionality of EKF

Endif

Endif

10.2.3 Selecting of Predefined Parameters

In SAFIS, some parameters need to be decided in advance according to the problems considered. They include the distance thresholds ($\varepsilon_{\max}$, $\varepsilon_{\min}$, γ), the overlap factor (κ) for determining the width of the newly added rule, the growing threshold (e_g)

Table 10.1 Effects of parameter e_g on system performance (number of rules and the testing RMS error) under different ε_{max} values and $\kappa = 1.0$

$\varepsilon_{max} \backslash e_g$	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 10.2 Effects of parameter e_g on system performance (number of rules and the testing RMS error) under different κ values and $\varepsilon_{max} = 10.0$

$\kappa \backslash e_g$	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)

for a new rule, and the pruning threshold (e_p) for removing an insignificant rule. Based on the observation from many experiments, 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; and 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 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-dimension sinc function:

$$z = \mathrm{sinc}(x, y) = \frac{\sin(x)\sin(y)}{xy}. \tag{10.28}$$

In the simulation, 2,500 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 $(\varepsilon_{min}, \gamma, e_p)$ are obeyed. ε_{min} is set to 10% of ε_{max}; γ is set to 0.997; and 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 10.1 and 10.2 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 no very big change when

the parameter κ or $\varepsilon_{\max}$ appears different values. However, these parameters are problem dependent and need to be determined according to the problem considered. Besides the above guidelines for setting the parameters, the optimal parameters can be determined using search techniques like GA for some complex problems in the future work.

10.3 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 [33], RANEKF [14], eTS [3], Simpl_eTS [1], and hybrid algorithm (HA) [30]. For the second system identification problem performance of SAFIS is compared with MRAN [33], RANEKF [14], eTS [3], Simpl_eTS [1], and SONFIN [13]. For the chaotic time-series prediction problem, the comparison is done with MRAN [33], RANEKF [14], eTS [3], and Simpl_eTS [1]. In all the studies, the parameters (r, Ω) for eTS and Simpl_eTS where r is the distance and Ω is the least-square error parameter [1, 3] 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.

10.3.1 Nonlinear Dynamic System Identification

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

$$\mathbf{y}(n) = \mathbf{f}[\mathbf{y}(n-1), \mathbf{y}(n-2), \ldots, \mathbf{y}(n-k+1); \mathbf{u}(n), \mathbf{u}(n-1), \ldots, \mathbf{u}(n-p+1)], \tag{10.29}$$

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:

$$\mathbf{y}_n = \mathbf{f}(\mathbf{x}_n). \tag{10.30}$$

The SAFIS algorithm is used to approximate $\mathbf{f}$ such that:

$$\hat{\mathbf{y}}_n = \hat{\mathbf{f}}(\mathbf{x}_n), \tag{10.31}$$

and the error between the system output $\mathbf{y}_n$ and the output of SAFIS $\hat{\mathbf{y}}_n$, $\|\mathbf{y}_n - \hat{\mathbf{y}}_n\|$ is minimized.

Narendra and Parthasarathy [22] have suggested two special forms of the nonlinear system model given in (10.32) and (10.33).

Model I:

$$y(n+1) = f[y(n), y(n-1), \ldots, y(n-k+1)] + \sum_{i=0}^{p-1} \beta_i u(n-i), \tag{10.32}$$

where β_i is the constant unknown parameter.

Model II:

$$y(n+1) = f[y(n), y(n-1), \ldots, y(n-k+1)] + g[u(n), u(n-1), \ldots, u(n-p+1)]. \tag{10.33}$$

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 this equation:

$$\hat{y}(n+1) = \hat{f}(y(n), y(n-1), \ldots, y(n-k+1), u(n), u(n-1), \ldots, u(n-p+1)), \tag{10.34}$$

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

10.3.1.1 Identification Problem 1

The first nonlinear dynamic system to be identified represents model I and is described by Wang and Yen [30]:

$$y(n) = \frac{y(n-1)y(n-2)(y(n-1)-0.5)}{1+y^2(n-1)+y^2(n-2)} + u(n-1). \tag{10.35}$$

The equilibrium state of the unforced system given by (10.35) is $(0,0)$. As in [30], the input $u(n)$ is uniformly selected in the range $[-1.5, 1.5]$ and the test input $u(n)$ is given by $u(n) = \sin(2\pi n/25)$; 5,000 and 200 observation data are produced for the purpose of training and testing. The different parameter values for SAFIS are chosen as follows: $\gamma = 0.997, \varepsilon_{\max} = 1.0, \varepsilon_{\min} = 0.1, \kappa = 1.0, e_g = 0.05$, and $e_p = 0.005$.

The average performance comparison of SAFIS with MRAN, RANEKF, eTS, Simpl_eTS, and HA is shown in Table 10.3 based on 50 experimental trials. From the table, it can be seen that SAFIS obtains similar testing accuracy compared

Table 10.3 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 [30]	28	0.0182	0.0244

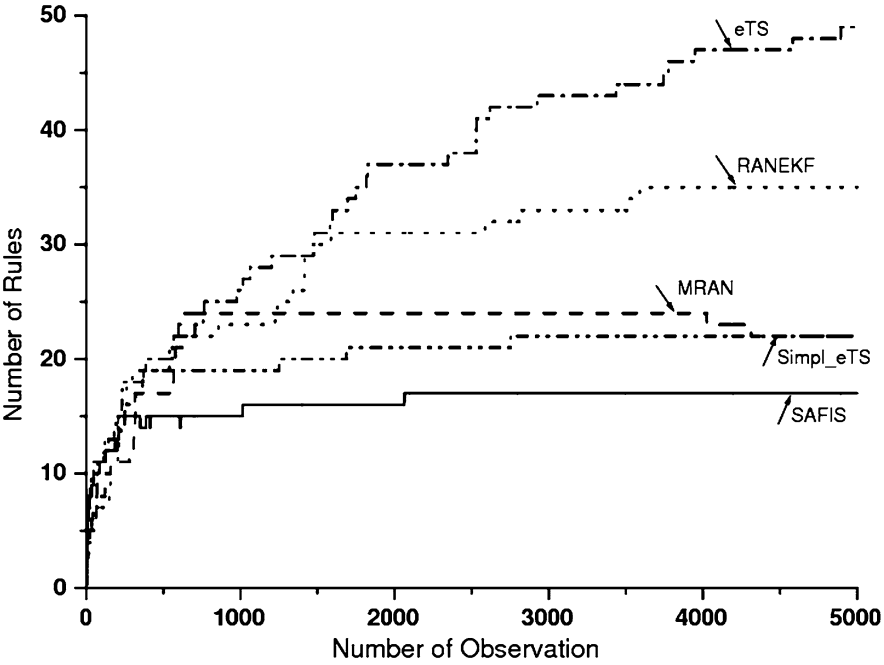

Fig. 10.2 Rule update process between different algorithms for nonlinear identification problem 1 during the whole observation

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. 10.2. It can be seen from the figure that SAFIS produces least number of rules. Besides, Fig. 10.3 gives a clear illustration for the rule evolution tendency between 0 and 1,000 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. The fuzzy rules for the typical run are listed in Table 10.4, where G(.) represents the Gaussian membership function. The first and second values in G(.) indicate the center and the width of the Gaussian function, respectively.

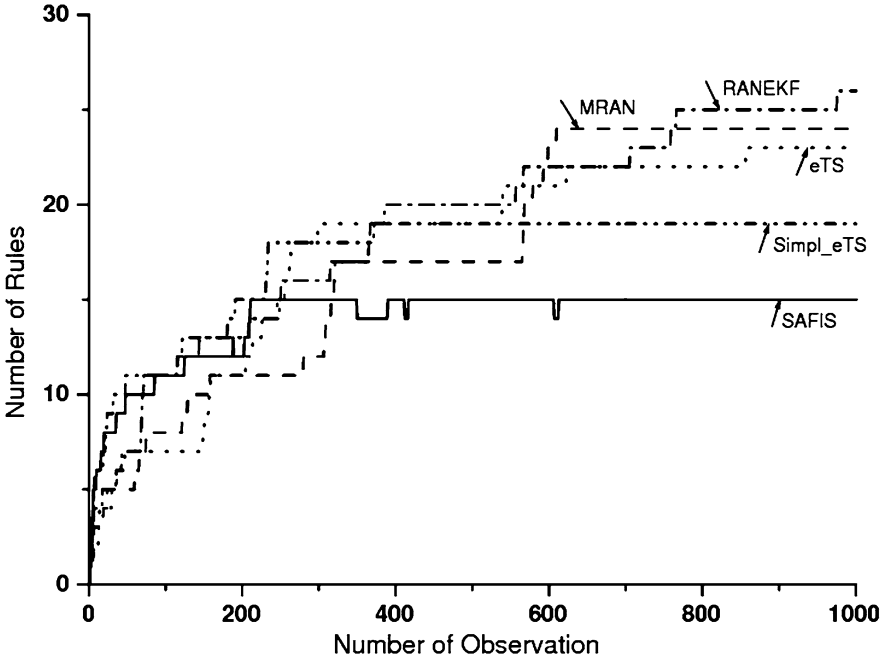

Fig. 10.3 Rule update process between different algorithms for nonlinear identification problem 1 between 0 and 1,000 observations

Table 10.4 Fuzzy rules of SAFIS for nonlinear identification problem 1

No. of rules	Antecedent parameters			Consequent parameters
	y(n-1)	y(n-2)	u(n-1)	
1	G(0.6883,0.8504)	G(0.3062,0.8504)	G(−2.0115, 0.8504)	a = −1.6137
2	G(−1.6117, 0.9709)	G(1.0091,0.9709)	G(1.7480,0.9709)	a = 2.5292
3	G(−0.2325, 1.1461)	G(−1.5054, 1.1461)	G(0.2722,1.1461)	a = 0.5625
4	G(−0.1653, 1.1633)	G(−1.0712, 1.1633)	G(1.9589,1.1633)	a = 1.5430
5	G(1.8338,1.2341)	G(0.3378,1.2341)	G(1.2441,1.2341)	a = 2.0280
6	G(1.5042,1.3481)	G(−0.5239, 1.3481)	G(0.0087,1.3481)	a = −0.4277
7	G(0.3110,0.9829)	G(−1.0793, 0.9829)	G(1.7363,0.9829)	a = 1.9069
8	G(−0.8126, 0.6423)	G(−1.2611, 0.6423)	G(−0.5928, 0.6423)	a = −1.1355
9	G(−0.6152, 0.9283)	G(−2.0362, 0.9283)	G(−1.4239, 0.9283)	a = −1.4374
10	G(−1.3413, 0.7751)	G(−1.0834, 0.7751)	G(−1.8843, 0.7751)	a = −2.4472
11	G(1.8475,1.1035)	G(−1.0128, 1.1035)	G(−1.4383, 1.1035)	a = −2.1617
12	G(0.7468,0.7356)	G(2.2865,0.7356)	G(1.4947,0.7356)	a = 1.8152
13	G(−2.3833, 1.9263)	G(−1.8191, 1.9263)	G(−1.0221, 1.9263)	a = −2.9007
14	G(−0.4007, 1.7921)	G(1.9986,1.7921)	G(−1.6721, 1.7921)	a = −2.0982
15	G(−1.6354, 1.8484)	G(1.8161,1.8484)	G(0.1940,1.8484)	a = 1.6630
16	G(−2.8360, 1.9440)	G(−2.0148, 1.9440)	G(1.0852,1.9440)	a = 0.9242
17	G(1.7938,0.9549)	G(1.4088,0.9549)	G(−0.3515, 0.9549)	a = 0.2905

Table 10.5 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 [13]	10	0.0130

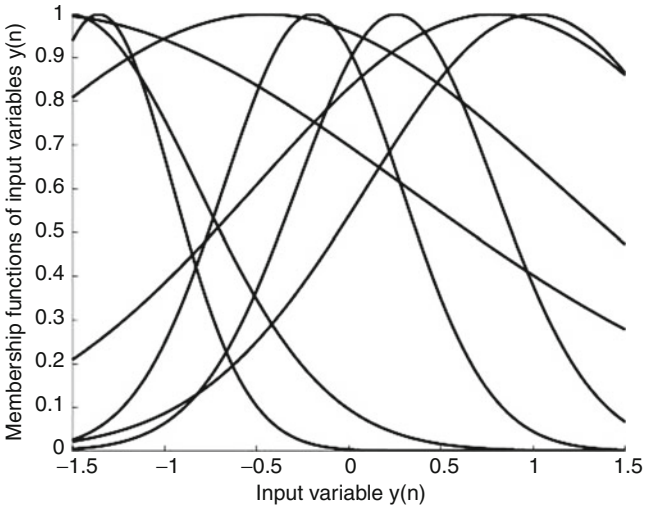

Fig. 10.4 Membership functions of input variable $y(n)$ for nonlinear identification problem 2

10.3.1.2 Identification Problem 2

The second nonlinear dynamic system to be identified represents model II and is described by Juang and Lin [13]:

$$y(n+1) = \frac{y(n)}{1+y^2(n)} + u^3(n). \tag{10.36}$$

In accordance with [13], the input signal $u(n)$ is given by $\sin(2\pi n/100)$; 50,000 and 200 observation data are produced for the purpose of training and testing. The SAFIS parameter values chosen are as follows: $\gamma = 0.997, \varepsilon_{\max} = 2.0, \varepsilon_{\min} = 0.2, \kappa = 2.0, e_g = 0.03$, and $e_p = 0.003$. 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 10.5 shows the performance comparison of SAFIS with MRAN, RANEKF, eTS, Simpl_eTS, and SONFIN [13]. It can be seen from the table that SAFIS achieves similar accuracy with a lesser number of rules. Figures 10.4 and 10.5 show the final membership functions of input variables $y(n), u(n)$ achieved by

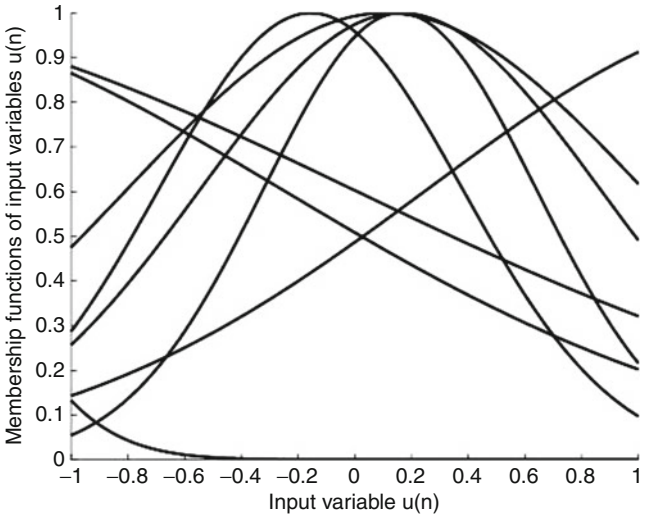

Fig. 10.5 Membership functions of input variable $u(n)$ for nonlinear identification problem 2

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. The evolution of the fuzzy rules for SAFIS, MRAN, RANEKF, eTS, and Simpl_eTS is shown by Fig. 10.6. It can be seen from the figure that SAFIS is able to add and delete rules during learning and produces least number of rules. The details of the fuzzy rules are depicted in Table 10.6.

10.3.2 Mackey-Glass Time-Series Prediction

In this example, the SAFIS is applied to predict complex time series, a special function approximation problem. The time-series prediction is very important in solving real-world problems such as the detection of arrhythmia in heartbeats. The chaotic Mackey-Glass time series is recognized as one of the time series benchmark problems, which is generated from the following differential equation [2]:

$$\frac{\mathrm{d}x(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t), \tag{10.37}$$

where $\tau = 17$ and $x(0) = 1.2$. For the purpose of training and testing, 6,000 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 the input

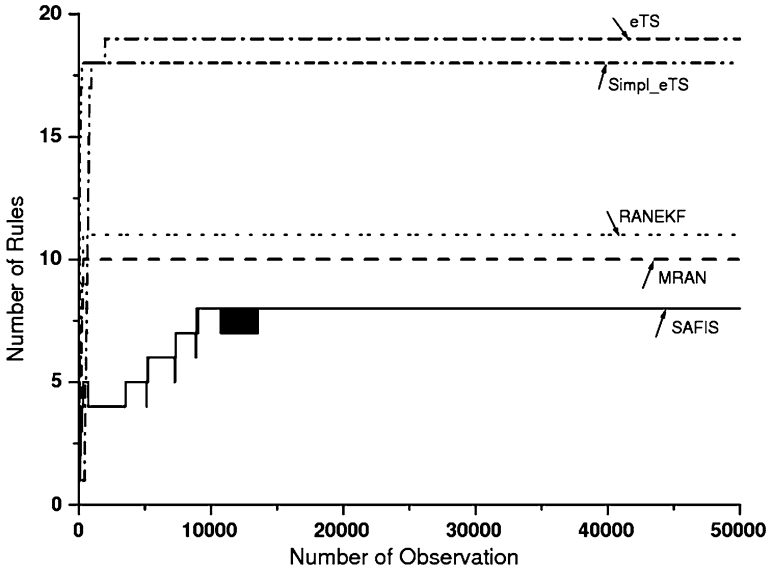

Fig. 10.6 Rule update process between different algorithms for nonlinear identification problem 2

Table 10.6 Fuzzy rules of SAFIS for nonlinear identification problem 2

No. of rules	Antecedent parameters		Consequent parameters
	y(n)	u(n)	
1	G(−0.9729, 1.0670)	G(−1.2625, 1.0670)	a = −6.3086
2	G(0.2731,1.4623)	G(1.9572,1.4623)	a = 4.5472
3	G(0.3706,0.8085)	G(−0.0884, 0.8085)	a = 3.1725
4	G(−0.1623, 1.1271)	G(0.1100,1.1271)	a = −6.4255
5	G(−0.8399, 1.1393)	G(−0.5162, 1.1393)	a = 4.6764
6	G(1.9988,2.1081)	G(1.8151,2.1081)	a = 1.8266
7	G(1.4377,2.3607)	G(1.2834,2.3607)	a = 2.2549
8	G(1.2992,0.8117)	G(0.3417,0.8117)	a = −1.8183

vector $[x(t-18)\ \ x(t-12)\ \ x(t-6)\ \ x(t)]$ for any value of the time t. As in [2], the observations between $t = 201$ and $t = 3{,}200$ and the observations between $t = 5{,}001$ and $t = 5{,}500$ are extracted from the series and used as training and testing data. For this problem, the parameters for SAFIS are selected as follows: $\gamma = 0.98, \varepsilon_{\max} = 1.6, \varepsilon_{\min} = 0.16, \kappa = 1.68, e_g = 0.0005$, and $e_p = 0.00005$. The data follow a uniform sample distribution in the range $[0.4, 1.4]$.

Table 10.7 shows the prediction accuracies and the number of rules obtained by SAFIS, MRAN, RANEKF, eTS, and Simpl_eTS. 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. As observed

Table 10.7 Results of Mackey-Glass time-series prediction

Methods	No. of rules	Testing NDEI
SAFIS	6	0.376
MRAN	14	0.375
RANEKF	18	0.378
Simpl_eTS($r = 0.25$, $\Omega = 750$) [1]	11	0.394
eTS($r = 0.25$, $\Omega = 750$) [2]	9	0.380

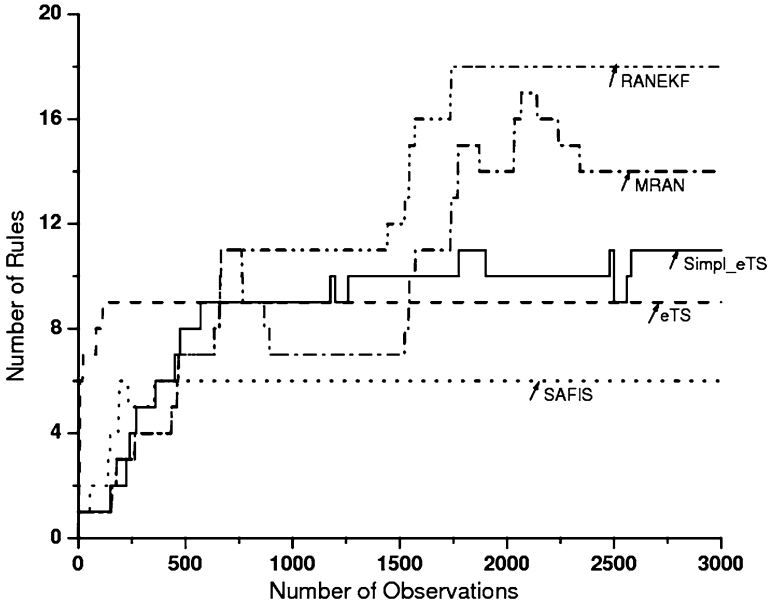

Fig. 10.7 Rule update process between different algorithms for Mackey-Glass time-series prediction

from Table 10.7, all the algorithms produce similar accuracies; however, SAFIS obtains the smallest number of fuzzy rules. The evolution of the fuzzy rules for SAFIS, MRAN, RANEKF, eTS, and Simpl_eTS is shown in Fig. 10.7.

10.4 Summary

In this chapter, a sequential fuzzy inference system called SAFIS is presented to automatically construct a fuzzy inference system using the training data during the learning process. Specifically, SAFIS algorithm implements the structure identification and parameter adjustment for a fuzzy inference system using the ideas from GAP-RBF algorithm. SAFIS algorithm utilizes the influence of a fuzzy rule to add and remove the fuzzy rules during learning. At the same time, the SAFIS

algorithm utilizes the EKF to update the parameters of the nearest rule instead of all the rules without losing the approximation performance. Its performance has been evaluated by some function approximation benchmark problems including two nonlinear system identification problems and the Mackey-Glass time-series prediction problem. The simulation results from these benchmark problems show that, compared with other algorithms, SAFIS produces similar or better testing accuracies with lesser number of rules.

However, for large systems, EKF algorithm used in the parameter update equation increases the computation burden. Also, the calculation of rule influence requires uniform distribution of the input data, and this may degrade the performance. Further studies in these directions are required in the future.

References

1. Angelov, P., Filev, D.: Simpl_eTS: A simplified method for learning evolving Takagi-Sugeno fuzzy models. In: The 14th IEEE International Conference on Fuzzy Systems, pp. 1068–1073 (2005)
2. Angelov, P., Victor, J., Dourado, A., Filev, D.: On-line evolution of Takagi-Sugeno fuzzy models. In: Proceedings of the 2nd IFAC Workshop on Advanced Fuzzy/Neural Control, pp. 67–72. Oulu, Finland (2004)
3. Angelov, P.P., Filev, D.P.: An approach to online identification of Takagi-Sugeno fuzzy models. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics **34**(1), 484–498 (2004)
4. Chiu, S.L.: Selecting input variables for fuzzy models. Journal of Intelligent and Fuzzy Systems **4**, 243–256 (1996)
5. Cho, K.B., Wang, B.H.: Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction. Fuzzy Sets and Systems **83**, 325–339 (1996)
6. Gopal, S., Karthikeyan, B., Kavitha, D.: Partial discharge pattern classification using fuzzy expert system. In: Proceedings of the 2004 IEEE International Conference on Solid Dielectrics, pp. 653–656. Toulouse, France (2004)
7. Gupta, M.M., Rao, D.H.: On the principles of fuzzy neural networks. Fuzzy Sets and Systems **61**(1), 1–18 (1994)
8. Huang, G.B., Saratchandran, P., Sundararajan, N.: An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks. IEEE Transactions on Systems, Man, Cybernetics-Part B: Cybernetics **34**(6), 2284–2292 (2004)
9. Huang, G.B., Saratchandran, P., Sundararajan, N.: A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. IEEE Transactions on Neural Networks **16**(1), 57–67 (2005)
10. Iqdour, R., Zeroual, A.: A rule based fuzzy model for the prediction of daily solar radiation. In: 2004 IEEE International Conference on Industrial Technology (ICIT), pp. 1482–1487. Hammamet, Tunisia (2004)
11. Jang, J.S.R.: ANFIS: Adaptive-network-based fuzzy inference system. IEEE Transactions on Systems, Man, and Cybernetics **23**(3), 665–685 (1993)
12. Jang, J.S.R., Sun, C.T.: Functional equivalence between radial basis function networks and fuzzy inference systems. IEEE Transactions on Neural Networks **4**(1), 156–159 (1993)
13. Juang, C.F., Lin, C.T.: An on-line self-constructing neural fuzzy inference network and its applications. IEEE Transactions on Fuzzy Systems **10**(2), 144–154 (2002)

14. Kadirkamanathan, V., Niranjan, M.: A function estimation approach to sequential learning with neural networks. Neural Computation **5**(6), 954–975 (1993)
15. Kalhor, A., Araabi, B.N., Lucas, C.: An online predictor model as adaptive habitually linear and transiently nonlinear model. Evolving Systems **1**, 29–41 (2010)
16. Kandel, A.: Fuzzy expert systems. Boca Raton, FL CRC Press (1992)
17. Kasaov, N.K., Song, Q.: DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time series prediction. IEEE Transactions on Fuzzy Systems **10**(2), 144–154 (2002)
18. Konjic, T., Miranda, V., Kapetanovic, I.: Prediction of LV substation load curves with fuzzy inference systems. In: Proceedings of the 5th International Conference on Probabilistic Methods Applied to Power Systems, pp. 129–134. Ames, Iowa (2004)
19. Leng, G., McGinnity, T.M., Prasad, G.: An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network. Fuzzy Sets and Systems **150**(2), 211–243 (2005)
20. Mamdani, E.H., Assilian, S.: An experiment in linguistic synthesis with a fuzzy logic controller. International Journal of Man-Machine Studies **7**(1), 1–13 (1975)
21. Mitra, S., Hayashi, Y.: Neuro-fuzzy rule generation: Survey in soft computing framework. IEEE Transactions on Neural Networks **11**(3), 748–768 (2000)
22. Narendra, K.S., Parthasarathy, K.: Identification and control of dynamical systems using neural networks. IEEE Transactions on Neural Networks **1**(1), 4–27 (1990)
23. Olej, V., Krupka, J.: Prediction of gross domestic product development by Takagi-Sugeno fuzzy inference systems. In: Proceedings of the 5th International Conference on Intelligent Systems Design and Applications (ISDA'05), pp. 186–191. Wroclaw, Poland (2005)
24. Pedrycz, W.: Fuzzy Control and Fuzzy Systems. New York: Wiley (1993)
25. Pedrycz, W., Rocha, A.F.: Fuzzy-set based models of neurons and knowledge-based networks. IEEE Transactions on Fuzzy Systems **1**(4), 254–266 (1993)
26. Platt, J.: A resource allocating network for function interpolation. Neural Computation **3**(2), 213–225 (1991)
27. Rubio, J.J.: SOFMLS: Online self-organizing fuzzy modified least-squares network. IEEE Transactions on Fuzzy Systems **17**(6), 1296–1309 (2009)
28. Soleimani, H., Lucas, C., Araabi, B.N.: Recursive Gath-Geva clustering as a basis for evolving neuro fuzzy modeling. Evolving Systems **1**, 59–71 (2010)
29. Tagaki, T., Sugeno, M.: Fuzzy identification of systems and its application to modelling and control. IEEE Transactions on Systems, Man, and Cybernetic **15**(1), 116–132 (1985)
30. Wang, L., Yen, J.: Extracting fuzzy rules for system modeling using a hybrid of genetic algorithm and Kalman Filter. Fuzzy Sets and Systems **101**, 353–362 (1999)
31. Wei, W., Mendel, J.M.: A fuzzy logic method for modulation classification in nonideal environments. IEEE Transactions on Fuzzy Systems **7**(3), 333–344 (1999)
32. Wu, S., Er, M.J.: Dynamic fuzzy neural networks - a novel approach to function approximation. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics **30**(2), 358–364 (2000)
33. Yingwei, L., Sundararajan, N., Saratchandran, P.: A sequential learning scheme for function approximation using minimal radial basis neural networks. Neural Computation **9**(2), 461–478 (1997)