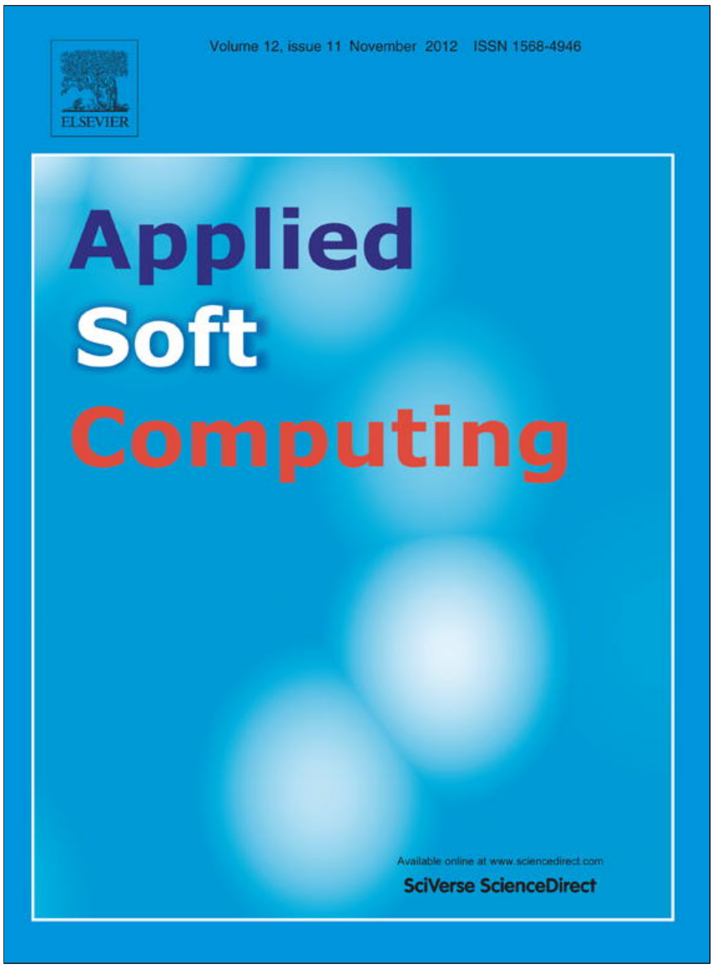
Volume 12, issue 11 November 2012 ISSN 1568-4946
ELSEVIER
Applied Soft Computing
Available online at www.sciencedirect.com
SciVerse ScienceDirect

Applied Soft Computing 12 (2012) 3603–3614

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

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

A meta-cognitive sequential learning algorithm for neuro-fuzzy inference system

K. Subramanian, S. Suresh*

School of Computer Engineering, Nanyang Technological University, Singapore

ARTICLE INFO

Article history:
Received 13 October 2011
Received in revised form 30 May 2012
Accepted 19 June 2012
Available online 5 July 2012

Keywords:
Meta-cognition
Self-regulatory learning
Neuro-fuzzy inference systems
Time series prediction
Non-linear system identification
Classification

ABSTRACT

In this paper, we present a meta-cognitive sequential learning algorithm for a neuro-fuzzy inference system, referred to as, 'Meta-Cognitive Neuro-Fuzzy Inference System' (McFIS). McFIS has two components, *viz.*, a cognitive component and a meta-cognitive component. The cognitive component employed is a Takagi–Sugeno–Kang type-0 neuro-fuzzy inference system. A self-regulatory learning mechanism that controls the learning process of the cognitive component, by deciding *what-to-learn*, *when-to-learn* and *how-to-learn* from sequential training data, forms the meta-cognitive component. McFIS realizes the above decision by employing *sample deletion*, *sample reserve* and *sample learning* strategy, respectively. The meta-cognitive component use the instantaneous error of the sample and spherical potential of the rule antecedents to select the best training strategy for the current sample. Also, in sample learning strategy, when a new rule is added the rule consequent is assigned such that the localization property of Gaussian rule is fully exploited. The performance of McFIS is evaluated on four regression and eight classification problems. The performance comparison shows the superior generalization performance of McFIS compared to existing algorithms.

1. Introduction

A fuzzy inference system and artificial neural networks are both universal estimators with the ability to estimate any non-linear function to a prescribed degree of accuracy [1]. The functional relationship between a radial basis function and fuzzy inference system [2] has been exploited by many researchers to provide neuro-fuzzy inference systems (NFIS) [1,3–8], with the intention of merging the advantages of fuzzy logic and neural networks. Various works available in literature have demonstrated the ability of NFIS for system modeling and identification [1,3–9]. In [10,11] NFISs have been employed in medical domain, whereas [12,13] aim to solve electrical load forecasting problems. Various other applications including industrial control [14,15] and surveillance [16] have been reported in literature. One can opt either batch learning mode [2,6,17,18] or sequential learning mode [3,5,7,19] to train the network. In batch learning mode, the network parameters are tuned by presenting all the training samples repeatedly (epochs). For most of the practical time-series forecasting and system identification problems, the complete training data is not available *apriori* or the non-linear relationship between input/output changes with respect to time. For these problems, training neuro-fuzzy network in batch mode is not suitable. Therefore, researchers are focusing on developing algorithms in which the structure and parameters adapts temporally and such an inference system is called adaptive NFIS.

An adaptive sequential NFIS training algorithm starts with no rules and builds up the required number of rules to approximate the non-linear function accurately by presenting the training samples one-by-one and only once. One of the first research work on adaptive neuro-fuzzy inference systems is evolving fuzzy neural networks by Kasabov [19] based on the concept of resource allocation network [20]. Dynamic evolving fuzzy inference system [17] is another evolving fuzzy neural network which uses an evolving clustering method to add rules. The output of this network is calculated based on the *m*-most significant rules chosen by offline clustering from a fuzzy rule set and is therefore not a truly sequential algorithm. In Growing and Pruning Fuzzy Neural Network (GP-FNN) [8], the sensitivity of a rule with respect to a current sample is used as a measure of influence for rule growing/pruning. Here, a computationally intensive Fourier decomposition of the variance of the output over the past samples is used for the growing/pruning criteria and a slow gradient descent method is used for updating the structure thereby increasing the training time of the system.

Dynamic fuzzy neural network, which implements a Takagi–Sugeno–Kang (TSK) based extended Radial Basis Function (RBF) neural network has been proposed in [1,21]. Although, the algorithm employs a hierarchical on-line self-organized learning scheme where the width of the RBF unit is dynamically adjusted, the learning algorithm requires the complete training samples *apriori* and primarily used for off-line learning. A self-constructing neuro-fuzzy inference network which uses a clustering scheme

* Corresponding author. Tel.: +65 6790 6185.
E-mail address: ssundaram@ntu.edu.sg (S. Suresh).

1568-4946/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2012.06.012

with an input data alignment scheme and a projection based correlation measure to evolve the system architecture has been proposed in [9]. In absence of pruning technique, the number of rules in this network grow with cluster, resulting in a large network with redundant rules. This problem of rule pruning has been addressed in [7,22]. In [22], a Self-Organizing Fuzzy Neural Network (SOFNN) which employs an optimal brain surgeon based rule pruning strategy, whereas, in [7], a Self-Organizing Fuzzy Modified Least Squared (SOFMLS) algorithm which employs density based rule pruning strategy have been used.

An evolving Takagi–Sugeno model (eTS) which recursively updates the structure of the network has been proposed in [3]. eTS employs the concept of potential, which is based on distance of current sample from all other training samples, to update the structure, and is computationally intensive. Also, eTS does not incorporate any rule pruning strategies and result in memory wastage and high computation cost. The above mentioned drawbacks in eTS are circumvented in Simpl_eTS [4] by employing scatter as a measure of density. The algorithm minimizes the number of rules based on the population of each rule (number of training samples associated with each rule). Many variations of eTS algorithms can be found in literature [23–25]. In [25] a fast evolving neuro-fuzzy model has been proposed. Here, a recursive Gatha-Geva clustering is employed to take advantage of elliptical clusters as membership functions. An inverse singular value decomposition is employed for reducing the computational cost involved in elliptical clusters.

In [26], an Online/Sequential Fuzzy Extreme Learning Machine (OS-fuzzy-ELM) using recursive least square method has been proposed. In OS-fuzzy-ELM, the parameters of the fuzzy membership functions are selected randomly and the consequent parameters of the fuzzy rules are calculated analytically using the recursive least squares method. Since the number of rules has to be fixed *apriori*, this algorithm is not truly adaptive in nature.

Sequential Adaptive Fuzzy Inference System (SAFIS), which uses the concept of influence of a rule in a statistical sense for growing/pruning the rules, has been proposed [5]. In SAFIS, the parameters of the nearest rule are updated using an extended Kalman filter. But, SAFIS does not consider the influence of the current sample while assigning a new rule consequent resulting in sub-optimal efficiency.

All the above mentioned algorithms only address the issue of *how-to-learn* a given sample efficiently. These algorithms train the network employing all the samples assuming the data to be uniformly distributed. Such scenarios are seldom encountered by practical problems there by leading to over-training. Thus there is a need to develop a neuro-fuzzy inference system which is able to self-regulate its learning by deciding *what-to-learn*, *when-to-learn* and *how-to-learn* from training samples. Recent studies in human educational psychology [27,28] and available works in machine learning literature [29–32] indicate that meta-cognition help realize self-regulation by enabling the human learner to assess ones current knowledge, identify when a new knowledge is required and in addition, provide the human learner with strategies to acquire new knowledge.

The interactions between cognitive processes (learning from experience) and the meta-cognitive processes (derivative knowledge stored in the semantic memory) are clearly explained in the area of human learning psychology [28,33]. It has been theoretically proven in the literature of neuro-fuzzy inference systems that, such approaches achieve better performance [34].

Although several models of meta-cognition are available in literature [27], one of the simplest model is Nelson and Narens model [28]. Hence in this paper, we develop a meta-cognitive neuro-fuzzy inference system based on this model.

According to their model of meta-cognition, cognitive processes could be split into at least two specifically interrelated components referred to as cognitive component and meta-cognitive component as shown in Fig. 1(a). There are two main relations ‘control’ and ‘monitoring’ which are defined in terms of direction of flow of signal between the meta-cognitive component and cognitive component. The signal flowing from meta-cognitive component to cognitive component is ‘control’, which is used to modify the cognitive component by performing one these actions: (1) initiate an action, (2) continue an action and (3) terminate an action. Since the control signal in itself does not yield any information about the state of the cognitive component, a logically independent ‘monitoring’ signal which informs the meta-cognitive component about the cognitive component is employed. The meta-cognitive component changes its knowledge about the cognitive component depending on this signal.

In this paper, we propose a meta-cognitive sequential learning algorithm for a NFIS. During the learning process, as shown in Fig. 1(b), the self-regulatory learning mechanism (meta-cognitive component) *monitors* the knowledge in the NFIS (cognitive component) and *controls* the same by changing the state of the cognitive component or the cognitive component itself. McFIS achieves this by deciding on: (a) *what-to-learn*, (b) *how-to-learn* and (c) *when-to-learn*. These three components are realized through *sample deletion strategy*, *sample learning strategy* and *sample reserve strategy*:

- (a) *Sample deletion strategy*: Samples with similar information content are deleted without learning.
- (b) *Sample learning strategy*: Depending on the information content in the current sample with respect to existing knowledge, the meta-cognitive component initiates rule growing, parameter update and rule pruning actions. In rule growing, new rule will be added to McFIS such that the newly added Gaussian rule exploits the localization property completely. Extended decoupled Kalman filter is used to update the parameters of nearest rule and rules which is not contributing to McFIS output significantly will be pruned.
- (c) *Sample reserve strategy*: Samples with higher information content is learnt first, and those with lower information content are pushed to the rear of the data-stream to be learnt at a later stage in the training process. By virtue of the self-regulatory thresholds, these samples will later be used in the learning process.

These three actions addresses the *what-to-learn*, *how-to-learn* and *when-to-learn* components of meta-cognition, respectively and help McFIS achieve the best human learning strategy reported in literature.

The performance of McFIS is evaluated on two non-linear system identification problems discussed in [4,5,9], Mackey-Glass time series prediction problem [35] and Box–Jenkins gas furnace problem to estimate the output CO_2 concentration from input gas flow [36]. In addition, the performance of McFIS is also compared on classification problems from UCI Machine Learning Repository [37]. The performance of McFIS for approximation problems is compared with well-known algorithms in neuro-fuzzy framework like sequential adaptive fuzzy inference system [5], OS-fuzzy-ELM [26], evolving Takagi–Sugeno [3] and its variant (simpl_eTS) [4] and recently developed algorithms like, self organizing fuzzy modified least square network [7] and fast evolving neuro-fuzzy model [25]. The performance for classification problem is compared with Meta-cognitive Neural Network (McNN) [31], which is a specifically designed classifier based on meta-cognitive principles, in addition to SAFIS and OS-Fuzzy-ELM. The results show the advantage of meta-cognitive neuro-fuzzy inference system over other neuro-fuzzy inference systems.

This paper is organized as follows: In the next Section (2), we will describe the neuro-fuzzy inference system and its sequential self-regulatory learning algorithm. Section 3 presents the

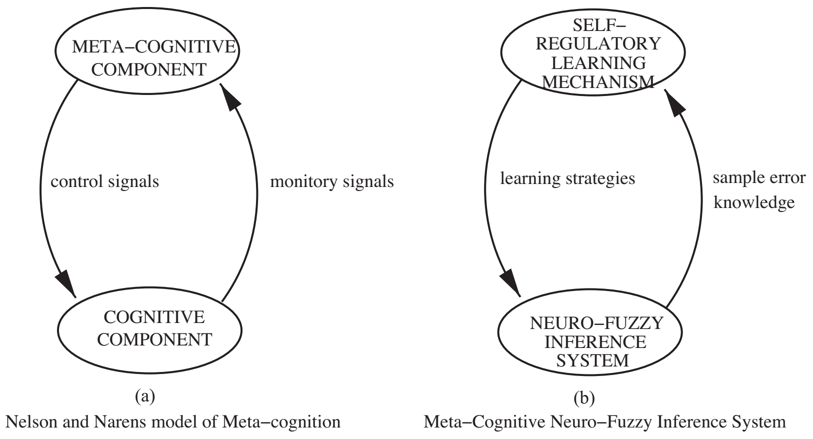
META–COGNITIVE COMPONENT
control signals
monitory signals
COGNITIVE COMPONENT
(a)
Nelson and Narens model of Meta–cognition
SELF–REGULATORY LEARNING MECHANISM
learning strategies
sample error knowledge
NEURO–FUZZY INFERENCE SYSTEM
(b)
Meta–Cognitive Neuro–Fuzzy Inference System

Fig. 1. Model of meta-cognition adapted from [28] and its analogy to McFIS.

performance evaluation of McFIS on benchmark non-linear system identification, time series prediction problem and Box–Jenkins CO_2 concentration prediction problem. Validity and verification of concepts introduced in McFIS is discussed in Section 4. The performance comparison on classification problems is presented in 5. This paper is concluded in Section 6 by summarizing the key features of McFIS.

2. Meta-cognitive neuro-fuzzy inference system

Meta-cognitive neuro-fuzzy inference system (McFIS) consists of two components, *viz.*, a cognitive component and a meta-cognitive component. The cognitive component is a TSK type-0 neuro-fuzzy inference system, while the meta-cognitive component is a self-regulatory learning mechanism that controls the learning ability of the neuro-fuzzy inference system. The learning mechanism is an adaptive sequential learning algorithm that starts with no rule and builds the necessary number of rules based on the information contained in the training samples. In this section, we describe the neuro-fuzzy inference system followed by its self-regulatory learning mechanism.

Let us assume that we have a stream of training data represented as input–output pair given by $\{(\mathbf{x}(1), \mathbf{y}(1)), \ldots, (\mathbf{x}(t), \mathbf{y}(t)), \ldots\}$, where $\mathbf{x}(t) = [x_1(t), \ldots, x_m(t)]^T \in m\Re$ is the input to the network and $\mathbf{y}(t) = [y_1(t), \ldots, y_n(t)]^T \in n\Re$ is its corresponding target. For large range multiple-input–multiple-output dynamic system, the $\mathbf{x}(t)$ could be expressed as $\mathbf{x}(t) = [\mathbf{y}(t-1), \ldots, \mathbf{y}(t-\nu-1); \mathbf{v}(t-1), \ldots, \mathbf{v}(t-\chi-1)]^T$, where, $\mathbf{v}$ is the input to the dynamic system and ν and χ are the maximum lags of the output and input, respectively. The above relationship could be expressed as:

$$\mathbf{y}(t) = \mathbf{f}[\mathbf{x}(t)] = \mathbf{f}[\mathbf{y}(t-1), \ldots, \mathbf{y}(t-\nu-1); \mathbf{v}(t-1), \ldots, \mathbf{v}(t-\chi-1)] \quad (1)$$

where $\mathbf{f}[.]$ is the functional relationship that maps the inputs to their respective targets ($\mathbf{x} \to \mathbf{y}$). The aim of McFIS is to approximate $\mathbf{f}[.]$ such that, the predicted output

$$\hat{\mathbf{y}}(t) = \hat{\mathbf{f}}[\mathbf{x}(t), \mathbf{w}(t)] \quad (2)$$

is as close as possible to the desired target $\mathbf{y}(t)$. It must be noted that $\mathbf{w}(t)$ is the parameter vector of McFIS. The difference between the actual and the predicted output (error $\mathbf{e}(t) = [e_1(t), \ldots, e_n(t)]^T$) is defined as,

$$e_j(t) = y_j(t) - \hat{y}_j(t) \quad j = 1, 2, \ldots n \quad (3)$$

The absolute prediction error is given by

$$E(t) = \|\mathbf{e}(t)\| \quad (4)$$

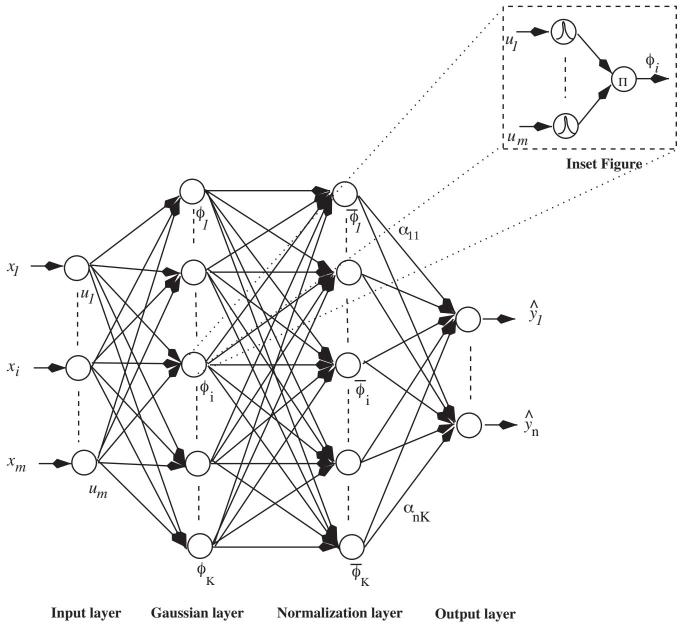

Fig. 2. Architecture of neuro-fuzzy inference system. The inset figure describes the internal structure of the *i*th Gaussian node.

2.1. Cognitive component: the neuro-fuzzy inference system

In McFIS, the cognitive component is a neuro-fuzzy inference system with Gaussian rules. It is a four layer network that realizes zero order Takagi–Sugeno–Kang type fuzzy inference system [38]. The architecture of the neuro-fuzzy inference system is given in Fig. 2. It consists of an input layer with m nodes, a Gaussian layer with K nodes which forms the rule antecedent and the aggregation layer of an NFIS, a normalization layer with K nodes and an output layer with n nodes. The weight vector connecting normalization layer and output layer forms the rule consequent parameters of an NFIS. A detailed description of each layer is given below:

Input layer: The input layer is a linear layer with m nodes. It transmits the input features directly to the Gaussian layer. The output of the ith input node is

$$u_i = x_i(t) \quad i = 1, 2, \ldots, m \quad (5)$$

Gaussian layer: The nodes in the Gaussian layer employ the Gaussian activation function to compute the membership value of each input node. The nodes in this layer forms the antecedent and does rule aggregation operations of a fuzzy inference system.

Similar to other neuro-fuzzy systems, McFIS exploits the functional equivalence between a radial basis function and a fuzzy inference system. Without loss of generality let us assume that K rules are grown from $t-1$ samples. The overall membership values of the input features are computed by each of the K rules of Gaussian layer. The firing strength of kth rule is given by

$$\phi_k(\mathbf{u}) = \prod_{i=1}^{m} \exp\left(\frac{-(u_i-\mu_{ki})^2}{2(\sigma_k)^2}\right), \quad k=1,2,\ldots,K \tag{6}$$

where μ_{ki} is the center of the kth Gaussian node for the ith input feature and σ_k is the width of the kth node. The above equation could be rewritten as,

$$\phi_k(\mathbf{u}) = \exp\left(\frac{-\|\mathbf{u}-\boldsymbol{\mu}_k\|^2}{2(\sigma_k)^2}\right), \quad k=1,2,\ldots,K \tag{7}$$

Normalization layer: This layer provides average firing potential for each rule. The output of the kth rule is

$$\overline{\phi}_k = \frac{\phi_k}{\sum_{l=1}^{K}\phi_l}, \quad k=1,2,\ldots,K \tag{8}$$

Output layer: The output layer is a linear layer with n nodes. The predicted output is given by

$$\hat{y}_j(t) = \sum_{k=1}^{K}\alpha_{jk}\overline{\phi}_k \quad j=1,2,\ldots,n \tag{9}$$

where α_{jk} is the weight connecting kth normalized node and the jth output node.

For a given input $\mathbf{x}(t)$, McFIS calculates the predicted output $\hat{\mathbf{y}}(t)$ by processing the input through all the four layers.

2.2. Meta-cognitive component: the self-regulatory learning mechanism

A self-regulatory learning mechanism forms the meta-cognitive component of McFIS. The self-regulatory learning mechanism controls the learning ability of the cognitive component by deciding *what-to-learn*, *when-to-learn* and *how-to-learn*, as shown in Fig. 3. As a result, when a sample is presented to the neuro-fuzzy inference system, the meta-cognitive component of McFIS controls the sequential learning process by invoking one of the three strategies *viz.*, sample delete, sample learning and sample reserve. These three self-regulatory strategies of meta-cognition helps McFIS to perform approximation tasks efficiently. We now describe, how these strategies are implemented in McFIS.

- (a) *Sample deletion strategy*: If the predicted error for a particular sample is below a threshold, then the knowledge in the current sample is similar to one already present in the network. Hence, it is deleted without being used in the learning process. The *sample deletion* criterion is given by

  $$\text{IF } E(t) \le E_d, \quad \text{THEN } \textbf{Delete} \text{ the } \textit{sample} \tag{10}$$

  where E_d is the delete threshold which is chosen according to the desired prediction accuracy. The sample deletion strategy prevents the learning of similar samples and thereby avoids over-training and reduces the computational effort.
- (b) *Sample learning strategy*: This strategy works by either 'growing' a new rule, 'updating' the parameters of the rules or pruning of insignificant rules from the network.

  Rule growing criterion: When a sample contains significant novel knowledge, a new rule has to be added to the network to capture this knowledge (novelty). Existing algorithms in neuro-fuzzy framework use error based criteria or influence of a rule as a measure of novelty. In addition to this error-based criterion, McFIS also uses the spherical potential that is widely used in kernel methods [39,40] as a measure of novelty.

  A brief description of spherical potential: The novelty is measured based on the projection of an input feature $\mathbf{x}(t)$ on to a hyper-dimensional feature space $\mathbb{S}$. Since a Gaussian function is employed for projection, the hyper-dimensional feature space will be spherical in nature [40]. In McFIS, the center ($\boldsymbol{\mu}$) and the width (σ) of the Gaussian rule antecedents describe the hyper-dimensional feature space $\mathbb{S}$. Let the origin of K-dimensional space be $\phi_0 = \frac{1}{K}\sum_{k=1}^{K}\phi(\boldsymbol{\mu}_k)$. The spherical potential (ψ) of any sample ($\mathbf{x}(t)$) in the feature space is expressed as the squared distance from the hyper-dimensional mapping $\mathbb{S}$ centered at origin ϕ_0 [40] and is given by

  $$\psi = \|\phi(\mathbf{x}(t)) - \phi_0\|^2 \tag{11}$$

  From [40] the above equation can be expanded as

  $$\psi = \phi(\mathbf{x}(t),\mathbf{x}(t)) - \frac{2}{K}\sum_{j=1}^{K}\phi(\mathbf{x}(t),\boldsymbol{\mu}_j) + \frac{1}{K^2}\sum_{i,j=1}^{K}\phi(\boldsymbol{\mu}_i,\boldsymbol{\mu}_j) \tag{12}$$

  From Eq. (12), it could be seen that for Gaussian function, the first term $\phi(\mathbf{x}(t),\mathbf{x}(t))$ and the last term $(\frac{1}{K^2}\sum_{i,j=1}K\phi(\boldsymbol{\mu}_i,\boldsymbol{\mu}_j))$ are constants and could be discarded as the potential is a measure of novelty. The spherical potential (ψ) is defined as the absolute value of quantity in squared distance mapping

  $$\psi = \left|-\frac{2}{K}\sum_{j=1}^{K}\phi\left(\mathbf{x}(t),\boldsymbol{\mu}_j\right)\right| \tag{13}$$

  The spherical potential is a direct measure of the knowledge contained in the sample, *i.e.*, a higher potential (close to two) indicates that the sample is similar to the existing knowledge in the network and a lower potential (closer to $\frac{2}{K}$) indicates that the sample is novel.

  Therefore, a new rule is added in McFIS if:

  $$\text{IF } E(t) > E_a \text{ AND } \psi < E_S \text{ THEN } \textit{add a rule} \tag{14}$$

  where E_S is the novelty threshold and E_a is the self-regulatory adding threshold. When a new rule is added, E_a is self-regulated according to:

  $$E_a := \delta E_a + (1-\delta)E(t) \tag{15}$$

  δ being the slope that controls the contribution of prediction error in self-regulating the thresholds. For all the problems in this paper δ is set close to 1. The self-regulatory adding threshold, E_a, is designed such that it captures the global knowledge first and at a later stage fine tunes the network by virtue of its self-regulatory nature. For the problems in this paper if the prediction error is higher than [0.1, 0.5], a new rule has to be added to capture this knowledge. If the range is kept too high, few new rules will be added to capture the knowledge which will affect generalization, where as too low range will result in large network size. According to the concept of spherical potential, a sample is novel if the spherical potential calculated is near zero. In this paper, we consider any sample with spherical potential less than [0.1, 0.5] to contain significant novelty which deem addition of new rule. These two parameters should be chosen depending on network size and required accuracy.

  When the $K+1$th fuzzy rule is added the antecedent parameters are initialized as

  $$\begin{aligned} \boldsymbol{\mu}_{K+1} &= \mathbf{x}(t) \\ \sigma_{K+1} &= \kappa * \min_{\forall j}\|\mathbf{x}(t)-\boldsymbol{\mu}_j\| \quad j=1,2,\ldots,K \end{aligned} \tag{16}$$

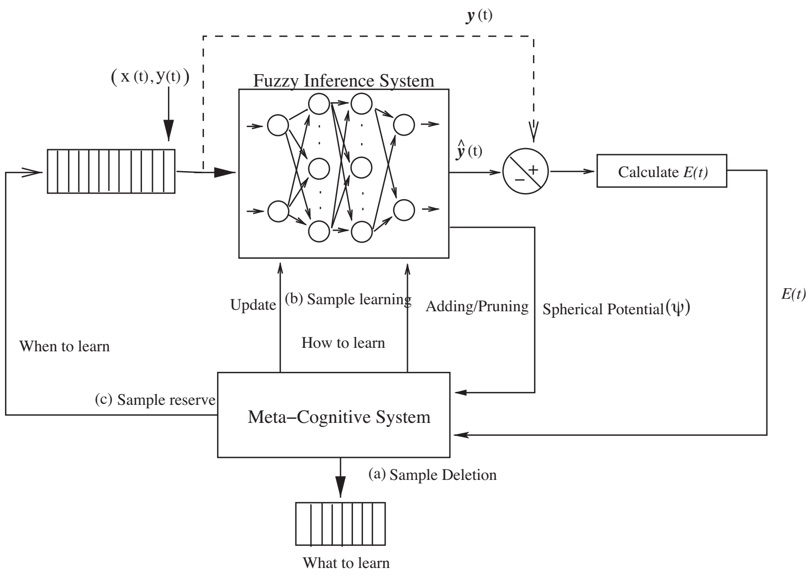

Fig. 3. Schematic diagram highlighting the self-regulating learning mechanism in McFIS.

where κ is the parameter which controls the overlap between the rule antecedents. For the problems considered in this paper, κ is chosen in the range [0.5, 0.9]. Lower value of κ will result in little overlap among rules which will affect the generalization ability of the network. Similarly, too much overlap among rules will also affect the performance adversely.

While initializing the output weight ($\boldsymbol{\alpha}_{K+1}$), the existing neuro-fuzzy algorithms do not fully exploit the localization property (respond to only a local region of input space [41]) of Gaussian (membership) function, efficiently [3–5]. The output weight ($\boldsymbol{\alpha}_{K+1}$) should be initialized such that they fully exploit the localization property of Gaussian (membership) function efficiently. On initialization of output weight, ($\boldsymbol{\alpha}_{K+1}$), the predicted output, $\hat{\mathbf{y}}(t)$, with $K+1$ rules should be equal to the actual output, $\mathbf{y}(t)$. Since McFIS employs a Gaussian function as the membership function, the firing strength of the new rules for the current sample is one ($\phi_{K+1}=1$). The output weight ($\boldsymbol{\alpha}_{K+1}$) is initialized as

$$\alpha_{j,K+1} = y_j(t) - \frac{\sum_{k=1}^{K}\alpha_{j,k}\phi_k}{1+\sum_{l=1}^{K}\phi_l}, \quad j=1,2,\ldots,m \tag{17}$$

Rule parameter update criterion: The parameters of the nearest rule is updated, when the prediction error ($\|\mathbf{e}\|$) is greater than self-regulatory parameter update threshold, E_l. If

$$E(t) \geq E_l, \tag{18}$$

then update the network parameters. The threshold, E_l, is self-regulated based on the prediction error as:

$$E_l := \delta E_l + (1-\delta)E(t) \tag{19}$$

where δ is the slope control parameter. The parameter update threshold is designed such that it captures the global knowledge first then fine tunes the network by virtue of its self-regulatory nature. For all the problems in this paper δ is set close to 1.

The parameters of the nearest rule ($\mathbf{w}_{nr} = [\boldsymbol{\alpha}_{nr}, \boldsymbol{\mu}_{nr}, \sigma_{nr}] \in (m+n+1)\Re$) are updated as

$$\mathbf{w}_{nr} = \mathbf{w}_{nr} + \mathbf{G}E(t) \tag{20}$$

where *nr* represents the rule antecedent which is nearest to the current input $\mathbf{x}(t)$ in a Euclidean sense, $E(t)$ is the error defined in Eq. (4) and $\mathbf{G} \in (m+n+1) \times n\Re$ is the Kalman gain matrix given by

$$\mathbf{G} = \mathbf{P}_{nr}\mathbf{a}[\mathbf{R} + \mathbf{a}^T\mathbf{P}_{nr}\mathbf{a}]^{-1} \tag{21}$$

Here $\mathbf{a} \in (m+n+1) \times n\Re$ is the gradient of the output with respect to the parameters ($\mathbf{w}_{nr}$), $\mathbf{R} = r_0 I_{n\times n}$ is the variance of measurement noise and $\mathbf{P}_{nr} \in (m+n+1)\times(m+n+1)\Re$ is the error covariance matrix of the nearest rule. Note that I is the identity matrix.

The gradient ($\mathbf{a}$) is given by

$$\mathbf{a} = \begin{bmatrix} \dfrac{\partial \hat{y}_1}{\partial \boldsymbol{\alpha}_{1,nr}} & \cdots & \dfrac{\partial \hat{y}_i}{\partial \boldsymbol{\alpha}_{1,nr}} & \cdots & \dfrac{\partial \hat{y}_n}{\partial \boldsymbol{\alpha}_{1,nr}} \\ \vdots & & \vdots & & \vdots \\ \dfrac{\partial \hat{y}_1}{\partial \boldsymbol{\alpha}_{n,nr}} & \cdots & \dfrac{\partial \hat{y}_i}{\partial \boldsymbol{\alpha}_{n,nr}} & \cdots & \dfrac{\partial \hat{y}_n}{\partial \boldsymbol{\alpha}_{n,nr}} \\ \dfrac{\partial \hat{y}_1}{\partial \boldsymbol{\mu}_{1,nr}} & \cdots & \dfrac{\partial \hat{y}_i}{\partial \boldsymbol{\mu}_{1,nr}} & \cdots & \dfrac{\partial \hat{y}_n}{\partial \boldsymbol{\mu}_{1,nr}} \\ \vdots & & \vdots & & \vdots \\ \dfrac{\partial \hat{y}_1}{\partial \boldsymbol{\mu}_{m,nr}} & \cdots & \dfrac{\partial \hat{y}_i}{\partial \boldsymbol{\mu}_{m,nr}} & \cdots & \dfrac{\partial \hat{y}_n}{\partial \boldsymbol{\mu}_{m,nr}} \\ \dfrac{\partial \hat{y}_1}{\partial \sigma_{nr}} & \cdots & \dfrac{\partial \hat{y}_i}{\partial \sigma_{nr}} & \cdots & \dfrac{\partial \hat{y}_n}{\partial \sigma_{nr}} \end{bmatrix} \tag{22}$$

and

$$\frac{\partial \hat{y}_i}{\partial \alpha_{i,nr}} = \overline{\phi}_{nr}, \quad i=1,2,\ldots,n \tag{23}$$

$$\frac{\partial \hat{y}_i}{\partial \mu_{j,nr}} = 2\phi_{nr}\frac{x_j-\mu_{j,nr}}{\sigma_{nr}^2}\frac{\alpha_{i,nr}-\hat{y}_i}{\sum_{l=1}^{K}\phi_l}, \quad j=1,2,\ldots,m \tag{24}$$

$$\frac{\partial \hat{y}_i}{\partial \sigma_{nr}} = 2\phi_{nr}\frac{||\mathbf{x}-\boldsymbol{\mu}_{nr}||^2}{\sigma_{nr}^3}\frac{\alpha_{i,nr}-\hat{y}_i}{\sum_{l=1}^{K}\phi_l} \tag{25}$$

The error covariance matrix ($\mathbf{P}_{nr}$) is updated as

$$\mathbf{P}_{nr} = \left[I_{z\times z} - \mathbf{G}\mathbf{a}^T\right] \mathbf{P}_{nr} + q_0 I_{z\times z} \tag{26}$$

Thus the *sample delete strategy*, *sample learning strategy* and *sample reserve strategy* helps McFIS actualize the meta-cognitive components of *what-to-learn*, *when-to-learn* and *how-to-learn*.

The proposed McFIS is summarized in Algorithm 1.

Algorithm 1. Pseudo-code for McFIS.

```
while samples in data-stream are learnt do
    for each input $\mathbf{x}(t)$ do
        Calculate the network output $\hat{\mathbf{y}}(t)$ (eqn.(9)), prediction error,
        $E(t)$ (eqn.(4)) and significance $\psi$ (eqn.(13)) for a given input $\mathbf{x}(t)$.
        if $E(t) \leq E_d$ then
            Sample is deleted from training sequence without learning.
        else if $E(t) \geq E_a$ AND $\psi < E_S$ then
            A new rule is added according to eqs. (16), (17), (28). The self-regulating
            adding threshold is updated according to eqn. (15).
        else if $\mathbf{E(t)} \geq \mathbf{E}_l$ then
            The parameters of the nearest rule is updated according to eqn. (20). The
            self-adaptive learning threshold is adapted according to eqn. (19).
        else if $\beta_i \leq \leftarrow E_p$ consistently then
            Prune the rule from the network and update the error covariance matrix
            $\mathbf{P}$.
        else
            Push the sample to the rear-end of the sample pool to be learnt later. if a
            sample is not learnt after few processing then
                remove it from the data-stream.
            end
    end
end
```

where $z = m + n + 1$ and the scalar quantity (q_0) determines the allowed step in the direction of gradient vector.

Rule pruning strategy: A rule is deemed to be insignificant and pruned from McFIS if its contribution (β_k) is lower than a pruning threshold, E_p, for N_w consecutive samples. If E_p and N_w are chosen very large, more rules will be pruned from the network and it may lead to system instability. Also, lower value of E_p and N_w will lead to few rules being pruned thereby increasing the network size and affecting the network performance. The choice of E_p and N_w depends on the training data size. The contribution of a rule (β_k) for a rule k is given as

$$\beta_k = \overline{\phi}_k \max_j \left| e_j \alpha_{kj} \right| \quad j = 1, \ldots, m \tag{27}$$

where ψ (Eq. (13)) is the spherical potential calculated, ϕ_k (Eq. (7)) is the firing strength of the kth rule and $\boldsymbol{\alpha}_k$ is the output weight of the kth rule.

When a rule is added to the network, the dimensionality of error covariance matrix is updated as

$$\mathbf{P} = \begin{bmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & p_0 I_{z\times z} \end{bmatrix} \tag{28}$$

where p_0 is the measure of estimated uncertainty. Similarly when a rule is pruned from the network, the dimensionality of the error covariance matrix is reduced by removing the respective rows and columns of the matrix $\mathbf{P}$, i.e., remove $(i-1)(m+n+1)+1$ to $i(m+n+1)$ rows/columns of matrix $\mathbf{P}$.

- (c) *Sample reserve strategy*: Unlike *sample deletion strategy* that deletes samples which do not contain any new information, *sample reserve strategy* defers the learning process to a later stage. This is achieved by pushing the current sample to the rear-end of the training data stream. These samples may be used at a later stage in learning process by the virtue of self-regulatory nature of McFIS. The meta-cognitive component of *when-to-learn* is realized in McFIS by the *sample reserve strategy*.

Next, we shall evaluate the performance of the proposed McFIS on standard benchmark problems.

3. Performance evaluation of McFIS for function approximation problems

In the previous section, the principle and working of the sequential learning algorithm for meta-cognitive neuro-fuzzy inference system was discussed in detail. In this section, the effectiveness of McFIS is demonstrated on two non-linear system identification problems discussed in [4,5,9], Mackey-Glass time series prediction problem [35] and Box–Jenkins gas furnace problem to estimate the output CO_2 concentration from input gas flow [36]. The performance of McFIS is compared with well-known algorithms in neuro-fuzzy framework like sequential adaptive fuzzy inference system (SAFIS) [5], evolving Takagi–Sugeno (eTS) [3] and its variant (simpl_eTS) [4], recently developed algorithms like, self organizing fuzzy modified least square network (SOFMLS) [7], online/sequential fuzzy extreme learning machine [26] and fast evolving neuro-fuzzy model (ENFM) [25]. The results show the advantage of meta-cognitive neuro-fuzzy inference system over other neuro-fuzzy inference systems.

3.1. Performance measures

In this study, two measures are employed for evaluating the performance of the algorithm.

- Root mean square error (RMSE) as used in [9,36] is employed as a performance measure. RMSE is defined as,

$$\text{RMSE} = \sqrt{\frac{E^2(t)}{N}}$$

where N is the total number of samples, $E(t)$ is the prediction error for the tth sample as defined in Eq. (4). It should be noted that, RMSE is an accuracy based performance measure.

- The percentage of samples used (PS) given by

$$PS = \frac{\text{number of samples used for training}}{\text{total number of samples available for training}} * 100\%$$

is used as an additional measure for comparison.

3.2. Non-linear system identification problem 1

First we shall consider non-linear system identification problem as described by Juang and Ling [9]. The system to be identified is given by

$$y(t+1) = \frac{y(t)}{1+y^2(t)} + u^3(t) \tag{29}$$

where $u(t)$ is the training input and $y(t)$ is the corresponding output. The input is generated using sinusoidal function given by, $u(t) = sin(2\pi t/100)$. The network is trained with 50,000 samples and tested with 200 samples. The output, $y(t)$ follows uniform sample distribution in the range [− 1.5, 1.5].

For this problem, the control parameters are chosen as follows: meta-cognitive parameters: {delete threshold, $E_d = 0.002$, add threshold, $E_a = 0.25$, parameter update threshold, $E_l = 0.12$} novelty threshold, $E_S = 0.5$, pruning parameters: {pruning threshold, $E_p = 0.5$, pruning window, $N_w = 90$}, EKF parameters: {$p_0 = 1.05$, $q_0 = 0.007$, $r_0 = 1.5$}, learning slope control parameter, $\delta = 0.006$, rule antecedent overlap constant, $\kappa = 0.9$. The logarithmic prediction error of the system trained using these parameters is given in Fig. 4 for every 100th training sample. It can observed from Fig. 4 that the most of the samples do not contribute to the learning process. The meta-cognitive components of *what-to-learn*, *when-to-learn* and *how-to-learn* has helped McFIS learn the underlying function well. We shall now consider how the different strategies present in meta-cognition aids accurate approximation using McFIS.

- (a) *Sample deletion strategy*: When the prediction error of a sample is less than the delete threshold, it is deleted without being used in the learning process. We shall exemplify the working of this strategy with a figure. Fig. 5 gives a snap-shot of prediction error for samples in the range 2100–2200 along with the delete threshold (E_d). Those samples with error lower than E_d (0.002) are deleted without participating in the learning process. In Fig. 5, the prediction error for sample at instant 2120 is lower than the expected error given by the delete threshold and is thus removed without participating in the learning process. By deleting samples with negligible novel information, the sample deletion strategy helps the network avoid over-training and save computational effort.
- (b) *Sample learning strategy*: The sample learning strategy works by either growing or pruning a rule in the network or updating the antecedent and consequent parameters of the nearest rule in the network based on the information present in the sample. *Spherical potential* along with self-regulating error based criteria are employed to determine the novelty of a sample. Let us study the effect of these measures in this problem by considering a snap-shot of first 100 samples. The spherical potential for these samples and the novelty threshold are given in Fig. 6.a, whereas, Fig. 6.b gives the prediction error, self-regulatory adding and parameter update threshold for these 100 samples.

  A sample contains significant novelty if the spherical potential calculated is less than the novelty threshold. In McFIS, a new rule is added to the network if the spherical potential calculated is lesser than the novelty threshold (0.5) and error is greater than the self-regulatory adding threshold. It could be noticed from the figures that even though the sample is novel, new rule is not added to the network if the error based criterion is not satisfied. Let us consider sample at instant 11. Since both the spherical potential calculated is lower than the novelty threshold and the instantaneous error is higher than the adding threshold, a new rule is added to the network. In McFIS, in addition to spherical potential, error based criterion should also be satisfied for rule addition. One could notice from Fig. 6.b that upon adding a new rule, the self-regulatory adding threshold adapts its value based on the prediction error.

  If the prediction error of a sample is greater than the self-regulatory parameter update threshold, the sample is used for updating the parameters of the network. In the Fig. 6.a the samples in the range 35–45 does not contain significant novel information to be considered for rule growing and are thus used to update the parameters of the nearest rule. The self-regulatory parameter update threshold adapts its value based on the prediction error until it reaches the delete threshold.
- (c) *Sample reserve strategy*: The samples which do not satisfy any of the learning criteria described above are reserved by the network to be considered for learning later. These samples, by the virtue of the self-regulatory nature of the adding and parameter update thresholds, may be used in the learning process at a later time. In Fig. 6, the samples marked with a diamond are reserved to be used later in the training process. It could be noticed that these samples do not satisfy either the adding or the parameter update criterion at the instant they were presented. These samples will be pushed to the rear of the data-stream to be learnt later. It should be noted that these samples may be used in the learning process at a later stage to fine tune the network.

We shall now perform a quantitative analysis of McFIS with existing algorithms. The result of performance analysis is given in Table 1. For this problem, McFIS is compared with well-known SAFIS [5], eTS [3], OS-fuzzy-ELM [26] and simpl_eTS [4] algorithms. Table 1 gives the number of rules employed, training and testing RMSE and the percentage of samples used by each of the algorithms. From the table, we can observe that the McFIS outperforms other algorithms considered. The McFIS attains better results than the next best performing algorithm SAFIS by utilizing merely 5 rules and 25% of the samples. McFIS is able to attain error an order of magnitude lower than other algorithms. The meta-cognitive component of *what-to-learn* has helped McFIS remove the rest 75% of the samples, whereas the *how-to-learn* and *when-to-learn* component together with the self-regulatory learning thresholds have helped McFIS approximate the underlying function accurately.

3.3. System identification problem 2

Let us consider another non-linear system identification problem which has been studied extensively [4,5,7], given by:

$$y(t) = \frac{y(t-1) * y(t-2) * (y(t-1) - 0.5)}{1 + y^2(t-1) + y^2(t-2)} + u(t-1) \tag{30}$$

where $u(t)$ and $y(t)$ are the input and output at tth instant.

As in the previous studies [5,7], the input to the system is given by $u(t) = sin(2\pi t/25)$ and the equilibrium state of the unforced system is (0,0). For training, 5000 data samples are generated and the trained model is tested using 200 data samples. The performance of the McFIS is compared with SAFIS [5], eTS [3], simpl_eTS [4] and SOFMLS [7]. For this study, the network parameters are chosen as: {$E_d = 0.002$, $E_a = 0.35$, $E_l = 0.08$}, $E_S = 0.15$, {$E_p = 0.07$, $N_w = 14$}, {$p_0 = 1.9$, $q_0 = 0.03$, $q_0 = 0.03$, $r_0 = 1.5$}, $\delta = 0.03$, $\kappa = 0.56$. The performance comparison of McFIS using the above parameters is given in Table 2. Table 2 gives the number of rules, training and testing RMSE and percentage of samples used by each of the algorithms.

From Table 2 it could be seen that McFIS using the same number of rules as next best performing algorithm SOFMLS, attains a

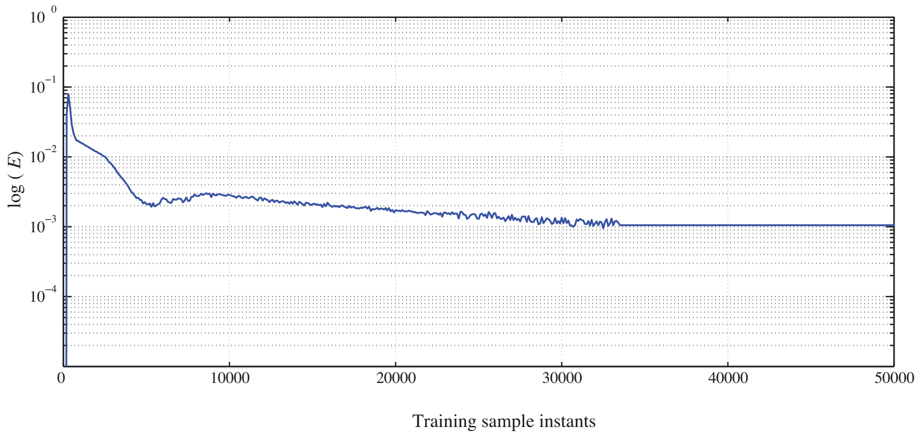

Fig. 4. Logarithmic prediction error for non-linear system identification problem 1.

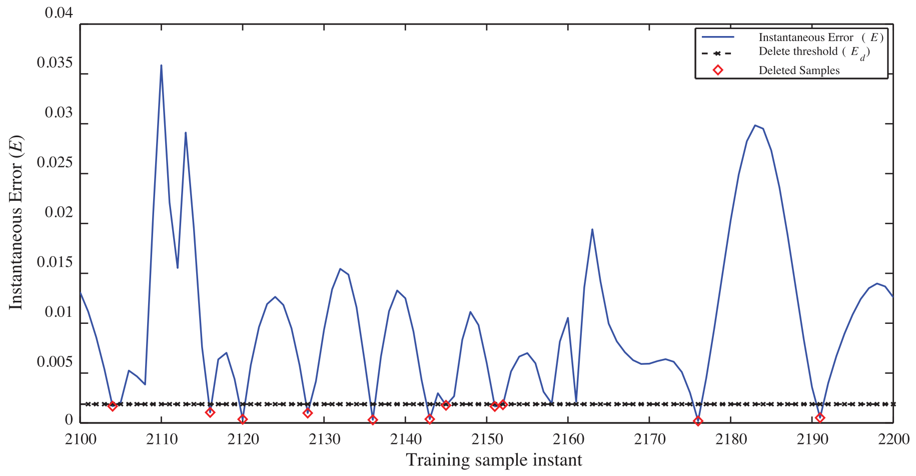

Fig. 5. Exemplification of sample deletion strategy for non-linear system identification problem 1.

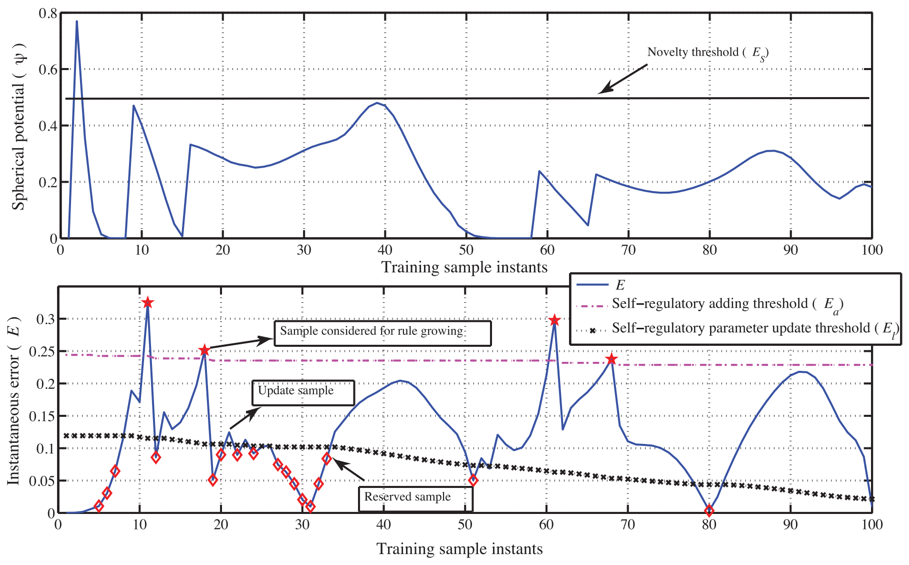

Fig. 6. Spherical potential and prediction error with self-regulatory thresholds for system identification problem 1.

network which produces significantly lower error. McFIS attains this performance by utilizing merely 50% of the training data. The spherical potential as a measure of novelty, is able to select samples which contains new knowledge and the complete exploitation of the Gaussian rule has helped the network approximate the samples well. In addition, the learning strategies has helped McFIS in choosing what samples to be learnt, when and how to learn those samples.

Table 1
Performance comparison for non-linear system identification problem 1.

Algorithm	# rules	CPU time (s)	Train RMSE	Test RMSE	PS
SAFIS [5]	17	140	0.0539	0.0221	100%
eTS [3]	49	–	0.0292	0.0212	100%
OS-fuzzy-ELM [26]	30	–	0.129	0.049	100%
simpl_eTS [4]	22	–	0.0528	0.0297	100%
McFIS	**5**	**20**	**0.0023**	**0.0023**	**25%**

Table 2
Performance comparison for non-linear system identification problem 2.

Algorithm	# rules	CPU time (s)	Training RMSE	Testing RMSE	PS
SAFIS [5]	17	4.2	0.0539	0.0221	100%
eTS [3]	49	–	0.0292	0.0212	100%
simpl_eTS [4]	22	–	0.0528	0.0225	100%
SOFMLS [7]	5	–	0.0341	0.0201	100%
McFIS	**5**	**2.3**	**0.002**	**0.004**	**51%**

Table 3
Performance comparison for Mackey-Glass series.

Algorithm	# rules	CPU time (s)	Test NDEI
eTS [3]	99	0.37	0.356
Simpl_eTS [4]	21	–	0.376
SAFIS [5]	21	–	0.38
ENFM [25]	8	–	0.06
McFIS	**6**	**0.33**	**0.03**

Table 4
Performance comparison for Box–Jenkins furnace problem.

Algorithm	# rules	CPU time (s)	Test RMSE
SAFIS [5]	5	0.26	0.071
eTS [3]	5	–	0.049
simpl_eTS [4]	3	–	0.0485
SOFNN [9]	4	–	0.48
SOFMLS [7]	5	–	0.047
McFIS	**4**	**0.23**	**0.036**

3.4. *Mackey-Glass time series problem*

One of the classical benchmark problem in literature is the Mackey-Glass time series prediction [35,42]. This chaotic time series data is generated from the differential equation:

$$\frac{dx}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{31}$$

where $\tau = 17$ and $x(0) = 1.2$. Aim of this study is to predict $x(t+\hat{h})$ from $[x(t), x(t-\Delta t), \ldots, x(t-(n-1)\Delta t)]$. For the sake of this study, we consider the following conditions: $n = 4$, $\Delta t = 6$, $\hat{h} = 85$ and the numerical solution to the differential equation (31) is found using the forth order Runge–Kutta method.

For comparing the performance of Mackey-Glass series data, we employ a normalized root mean squared error called non-destructive error index (NDEI) [4,5]. NDEI is defined as RMSE divided by the standard deviation of the target data:

$$\text{NDEI} = \frac{\text{RMSE}}{S.D.}$$

where, $S.D.$ is the standard deviation of the target data.

Table 3 gives the performance comparison of Mackey-Glass time series prediction with well-known algorithms like eTS [3],simpl_eTS [4], SAFIS [5] and ENFM [25]. Table 3 gives the number of rules employed and testing NDEI for each algorithm employed. To generate this result the following parameter setting was employed: $\{E_d = 0.0004, E_a = 0.4, E_l = 0.006\}, E_S = 0.4, \{E_p = 0.02, N_w = 37\}, \{p_0 = 1.4, q_0 = 0.003, r_0 = 1.1\}, \delta = 0.004, \kappa = 0.6$. It could be seen that McFIS attains the best performance with the least number of rules. It should be noted that McFIS uses about 98% of the samples to learn the underlying function in Mackey-Glass time series prediction problem. From the results, we can see that the proposed McFIS approximates the decision surface more efficiently than existing results in the literature.

3.5. *Box–Jenkins gas furnace problem*

The system identification of Box–Jenkins gas furnace [36] is a well-known problem. The aim of the problem is to predict the output CO_2 concentration from input gas flow rate in a furnace. The data set consists of 290 data samples of which, 200 samples are used for training and rest 90 samples are used for testing. For McFIS, a series-parallel model, $\hat{y}(t) = f(y(t-1), u(t-4))$ is used to model the system.

The performance of McFIS is compared with other algorithms in neuro-fuzzy framework like, eTS [3], simpl_eTS [4], SOFNN [22] and SOFMLS [7]. The following thresholds were utilized to generate the result in this paper: $\{E_d = 0.0003, E_a = 0.44, E_l = 0.01\}, E_S = 0.3, \{E_p = 0.07, N_w = 78\}, \{p_0 = 1.2, q_0 = 0.001, r_0 = 1.2\}, \delta = 0.004, \kappa = 0.8$. It could be observed from Table 4 that, McFIS approximates the functions better than other algorithms using fewer rules. Although SOFNN uses the same number of rules as McFIS, McFIS achieves significantly better performance. For Box–Jenkins Gas Furnace Problem, McFIS utilizes about 98% of the samples to learn the underlying data distribution.

4. Discussion

In this paper two significant contributions are made: (a) exploitation of localization property of Gaussian rule during rule addition and (b) meta-cognitive learning philosophy for training an NFIS. Authors argue that with the inclusion of these principles, the performance of any algorithm could be improved. In order to buttress these arguments, three separate experiments are performed using SAFIS algorithm.

- *Experiment A*: When a growing criterion is satisfied in SAFIS, the output weight ($\boldsymbol{\alpha}_{K+1}$) is updated using Eq. (17). The above changes will help SAFIS to exploit the localization property of the Gaussian rules effectively. We call this algorithm as 'SAFIS*'.
- *Experiment B*: The 'SAFIS*' is trained using the samples selected by McFIS to approximate the function. The selected training samples

Table 5
Verification of McFIS philosophy.

Data set	Algorithm	No. of rules	Train RMSE	Test RMSE
System identification 1	SAFIS	17	0.0539	0.0221
	SAFIS*	12	0.0327	0.0201
	SAFIS†	11	0.0301	0.0171
	SAFIS‡	11	0.022	0.0191
	McFIS	5	0.0023	0.0023
System identification 2	SAFIS	17	0.0539	0.0221
	SAFIS*	10	0.0152	0.0155
	SAFIS†	8	0.0118	0.0121
	SAFIS‡	8	0.0115	0.0124

Table 6
Description of benchmark data sets.

Type of data set	Problem	Number of features	Number of classes	Number of samples		Imbalance factor	
				Training	Testing	Training	Testing
Multi category	Image segmentation	19	7	210	2100	0	0
	Iris	18	4	424	422	0	0
	Wine	13	3	60	118	0	0.29
	Glass identification	9	6	109	105	0.68	0.77
Binary	Liver disorder	6	2	200	145	0.17	0.14
	PIMA	8	2	400	368	0.22	0.39
	Breast cancer	9	2	300	383	0.26	0.33
	Ionosphere	34	2	100	251	0.28	0.28

are presented one-by-one in the same order in which they are used in the McFIS learning algorithm. We call this algorithm as ‘SAFIS†’.

- *Experiment C*: The ‘SAFIS*’ is trained using the selected samples by McFIS followed by the deleted samples. We call this algorithm as ‘SAFIS‡’.

The three experiments are validated on system identification problem 1 and system identification problem 2. The results are available in Table 5.

From the table it could be noticed that by exploiting the localization property of Gaussian rule, there is a significant improvement in performance of SAFIS* compared to SAFIS, i.e., with fewer rules the network is better able to capture the knowledge in the samples. Earlier we had argued that presenting similar samples to the network lead to over-training and *what-to-learn* strategy has to be included in the learning algorithm. In order to support the claim, ‘Experiment B’ is conducted. The results from Table 5 indicates SAFIS†, employing fewer samples (selected by *what-to-learn* and *when-to-learn* components of McFIS), is able to attain better training and testing performance in comparison to SAFIS and SAFIS*. ‘Experiment C’ which employs samples selected by McFIS followed by deleted samples clearly shows the over-training in both the cases. The difference in the performance of SAFIS-variants and McFIS might be due to the difference in *how-to-learn* strategy in both the algorithms.

5. Performance evaluation of McFIS for classification problems

In this section, the effectiveness of McFIS is demonstrated on a set of benchmark classification problems from UCI Machine Learning Repository [37]. The performance of the proposed algorithm is compared with SAFIS [5], OS-Fuzzy-ELM [26] and McNN [31]. The data sets considered along with their descriptions, including the number of input features, the number of training and testing samples, and the imbalance factor of the data set is given in Table 6. The imbalance factor of a data set as defined in [43] is

$$\text{I.}F. = 1 - \frac{n}{N}\min_{i=1,\cdots,n} N_i \tag{32}$$

where N_i is the number of samples belonging to class l and $N = \sum_{l=1}^{n} N_i$.

5.1. Performance measures

In this paper, we have employed two measures to evaluate the performance of the algorithm on classification problems.

- Overall accuracy (η_o): It is defined as

$$\eta_o = \times\frac{1}{N}\sum_{i=1}^{n} q_{ii} \times 100\% \tag{33}$$

where q_{ii} is the number of samples correctly classified in class i.
- Average accuracy (η_a): It is given by

$$\eta_a = \frac{1}{N}\sum_{i=1}^{n} \frac{q_{ii}}{N_i} \times 100\% \tag{34}$$

5.2. Performance comparison

The number of rules, overall and average testing efficiency for SAFIS, OS-Fuzzy-ELM, McNN and McFIS are reported in Table 7. The table contains results of both the binary and multi category classification data sets from the UCI Machine Learning Repository. From Table 7, we can see that McFIS performs better than SAFIS and OS-Fuzzy-ELM on all the eight data sets.

On the balanced classification problems, the generalization performance of McFIS has improved by 1–5% compared to SAFIS and OS-Fuzzy-ELM, whereas McFIS performs similar to McNN. For these problems, the meta-cognitive components of *what-to-learn*, *when-to-learn* and *how-to-learn* has helped to build an efficient classifier. On the imbalanced data sets (wine, GI, Liver, PIMA, BC and ION), the generalization performance of McFIS is approximately 1–5%

Table 7
Performance comparison for classification problems.

Data sets	SAFIS			OS-Fuzzy-ELM			McNN			McFIS		
	K	Testing		K	Testing		K	Testing		K	Testing	
		η_o	η_a		η_o	η_a		η_o	η_a		η_o	η_a
IS	156	90.38	90.38	30	87.33	87.33	49	93.38	93.38	78	91.33	91.33
Iris	24	92.38	92.38	25	97.14	97.14	5	97.14	97.14	6	97.14	97.14
Wine	35	95.76	95.76	30	94.07	94.07	9	98.49	98.49	12	96.78	96.78
GI	134	73.24	83.32	150	76.19	78.1	73	85.71	87.03	100	82.9	85.7
Liver	133	68.28	68.82	100	66.21	65.68	68	73.79	71.60	66	71.7	73.4
PIMA	179	75.82	71.46	100	68.75	66.36	76	80.16	77.31	133	75.81	75.65
BC	55	97.39	97.84	90	91.64	89.45	9	97.39	97.85	33	97.39	97.85
ION	46	91.63	91.78	50	77.69	71.34	20	95.62	95.60	23	94.2	93.29

more than SAFIS and OS-Fuzzy-ELM with fewer rules. The complete exploitation of Gaussian rule along with spherical potential as measure of novelty has helped McFIS to efficiently induct rules. For some problems, performance of McFIS is slightly lower than McNN. This is because, in McNN, authors have developed a neural network based classifier employing class-specific learning criterion in addition to class-specific hinge loss function and spherical potential. If McFIS employs these class-specific criterions, the performance could be similar to that of McNN.

6. Conclusion

In this paper a meta-cognitive sequential learning algorithm for a neuro-fuzzy inference system, referred to as ‘Meta-Cognitive neuro-Fuzzy Inference System’ (McFIS), is presented. McFIS consists of two components: a cognitive component and a meta-cognitive component. An TSK-0 adaptive neuro-fuzzy inference system is the cognitive component and a self-regulatory sequential learning algorithm is the meta-cognitive component. When a new sample is presented the self-regulatory learning mechanism helps the neuro-fuzzy inference system to choose *what-to-learn*, *when-to-learn* and *how-to-learn* by activating one of the following strategies: *sample deletion strategy*, *sample reserve strategy* or *sample learning strategy*. In McFIS, the rule growing strategy employs spherical potential as a measure of novelty and addition of a new rule exploits the localization property of Gaussian rule effectively. The performance of McFIS is evaluated on four regression and eight classification problems. The performance analysis indicate the superior performance of McFIS.

Acknowledgements

The authors would like to acknowledge the Ministry of Education, Singapore, AcRF Tier I (2011-13) fund for supporting this research. Authors would also wish to thank the reviewers for their valuable suggestions.

References

[1] S. Wu, Er.-M. Joo, Dynamic fuzzy neural networks – a novel approach to function approximation, IEEE Transactions on Systems Man and Cybernetics Part B: Cybernetics 30 (2) (2000) 358–364.
[2] J.-S.-R. Jang, C.-T. Sun, Functional equivalence between radial basis function networks and fuzzy inference systems, IEEE Transactions on Neural Networks 4 (1) (1993) 156–159.
[3] P. Angelov, D. Filev, An approach to online identification of Takagi–Sugeno fuzzy models, IEEE Transactions on Systems Man and Cybernetics Part B: Cybernetics 34 (1) (2004) 484–498.
[4] P. Angelov, D. Filev, SIMPL_eTS: a simplified method for learning evolving Takagi–Sugeno fuzzy models, in: IEEE International Conference on Fuzzy Systems, 2005, pp. 1068–1073.
[5] H.-J. Rong, N. Sundararajan, G.-B. Huang, P. Saratchandran, Sequential adaptive fuzzy inference system (SAFIS) for non-linear system identification and prediction, Fuzzy Sets and Systems 157 (9) (2006) 1260–1275.
[6] G. Leng, M.-T. McGinnity, G. Prasad, Design for self-organizing fuzzy neural networks based on genetic algorithms, IEEE Transactions on Fuzzy Systems 14 (6) (2006) 755–766.
[7] J.de.J. Rubio, SOFMLS: online self-organizing fuzzy modified least-squares network, IEEE Transactions on Fuzzy Systems 17 (6) (2009) 1296–1309.
[8] H. Han, J. Qiao, A self-organizing fuzzy neural network based on a growing and pruning algorithm, IEEE Transactions on Fuzzy Systems 18 (6) (2010) 1129–1143.
[9] C.-F. Juang, C.-T. Lin, An online self-constructing neural fuzzy inference network and its applications, IEEE Transactions on Fuzzy Systems 10 (2) (1998) 144–154.
[10] L. Qin, S.-X. Yang, An adaptive neuro-fuzzy approach to risk factor analysis of *Salmonella typhimurium* infection, Applied Soft Computing 11 (8) (2011) 4875–4881.
[11] A. Yardimci, Soft computing in medicine, Applied Soft Computing 9 (3) (2009) 1029–1043.
[12] N. Amjady, F. Keynia, Application of a new hybrid neuro-evolutionary system for day-ahead price forcasting of electricity markets, Applied Soft Computing 10 (3) (2010) 784–792.
[13] M.-J.-B. Reddy, B.-K. Chandra, D.-K. Mohanta, A DOST based approach for the condition monitoring of 11 kV distribution line insulators, IEEE Transactions on Dielectrics and Electrical Insulation 18 (2) (2011) 588–595.
[14] W. Po-ngaen, R. Choomuang, J. Bhuripanyo, Neuro fuzzy in six dof tele-robotic control, in: IEEE International Conference on Mechatronics and Automation, 2008, pp. 959–964.
[15] E. Kayacan, Y. Oniz, A.-C. Aras, O. Kaynak, R. Abiyev, A servo system control with time varying and nonlinear load conditions using type-2 tsk fuzzy neural system, Applied Soft Computing 11 (8) (2011) 5735–5744.
[16] H.-T. Elshoush, I.-M. Osman, Alert correlation in collaborative intelligent intrusion detection systems: a survey, Applied Soft Computing 11 (7) (2011) 4349–4365.
[17] Q. Song, N. Kasabov, DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction, IEEE Transactions on Fuzzy Systems 10 (2) (2002) 144–154.
[18] J.-S.-R. Jang, ANFIS: adaptive network based fuzzy inference system, IEEE Transactions on Systems Man and Cybernetics 23 (3) (1993) 665–685.
[19] N. Kasabov, Evolving fuzzy neural networks for supervise/unsupervised online knowledge-based learning, IEEE Transactions on Systems Man and Cybernetics Part B: Cybernetics 31 (6) (2001) 902–918.
[20] J.-C. Platt, A resource allocating network for function interpolation, Neural Computation 3 (2) (1991) 213–225.
[21] S. Wu, Er.-M. Joo, Y. Gao, A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks, IEEE Transactions on Fuzzy Systems 9 (4) (2001) 578–594.
[22] G. Leng, T.-M. McGinnity, G. Prasad, An approach for online extraction of fuzzy rules using a self-organizing fuzzy neural network, Fuzzy Sets and Systems 150 (2) (2005) 211–243.
[23] E. Lughofer, P. Angelov, Handling drifts and shifts in online data streams with evolving fuzzy systems, Applied Soft Computing 11 (2) (2011) 2057–2068.
[24] P. Angelov, X. Zhou, Evolving fuzzy systems from data stream in real time, in: International Symposium on Evolving Fuzzy Systems, 2006, pp. 29–35.
[25] H.-B. Soleimani, C. Lucas, B. Araabi, Fast evolving neuro-fuzzy model and its application in online classification and time series prediction, Pattern Analysis and Applications (2011) 1–10.
[26] H.-J. Rong, G.-B. Huang, N. Sundararajan, P. Saratchandran, Online sequential extreme learning machine for function approximation and classification problems, IEEE Transactions on Systems Man and Cybernetics Part B: Cybernetics 39 (4) (2009) 1067–1072.
[27] M.-T. Cox, Meta-cognition in computation: a selected reserch review, Artificial Intelligence 169 (2) (2005) 104–141.
[28] T.-O. Nelson, L. Narens, Metamemory: a theoretical framework and new findings, Psychology of Learning and Motivation 26 (C) (1990) 125–173.
[29] S. Suresh, K. Dong, H.-J. Kim, A sequential learning algorithm for self-adaptive resource allocation network classifier, Neurocomputing 73 (16-18) (2010) 3012–3019.
[30] S. Suresh, R. Savitha, N. Sundararajan, A sequential learning algorithm for complex valued self-regulating resource allocation network -CSRAN, IEEE Transactions on Neural Networks 22 (7) (2011) 1061–1072.

[31] G. Sateesh Babu, S. Suresh, Meta-cognitive neural network for classification problems in a sequential learning framework, Neurocomputing 81 (1) (2012) 86–96.
[32] R. Savitha, S. Suresh, N. Sundararajan, Metacognitive learning in a fully complex-valued radial basis function neural network, Neural Computation 24 (5) (2012) 1297–1328.
[33] J. Flavell, Meta-cognition and cognitive monitoring: a new area of cognitive-developmental inquiry, American Psychologist 34 (10) (1979) 906–911.
[34] C. Castiello, G. Castellano, A. Fanelli, Designing a meta-learner by a neuro-fuzzy approach, in: IEEE Annual Meeting of Fuzzy Informations, 2004, pp. 893–898.
[35] M.-C. Mackey, L. Glass, Oscillation and chaos in physiological control systems, Science 197 (4300) (1977) 287–289.
[36] E.-P. George, G.-M. Jenkins, Time Series Analysis, Forecasting and Control, Holden Day, San Francisco, CA, 1976.
[37] C. Blake, C. Merz, UCI Repository of Machine Learning Databases, Department of Information and Computer Sciences, University of California, Irvine, 1998. http://archive.ics.uci.edu/ml/.
[38] T. Takagi, M. Sugeno, Fuzzy identification of systems and its application to modeling and control, IEEE Transactions on Systems Man and Cybernetics 15 (1) (1985) 116–132.
[39] B. Scholkopf, A.-J. Smola, Learning with Kernels, MIT Press, Cambridge, MA, 2002.
[40] H. Hoffmann, Kernel PCA for novelty detection, Pattern Recognition 40 (3) (2007) 863–874.
[41] J. Moody, C. Darken, Lerning with localized receptive fields, in: Proc. Connectionist Models Summer School, 1998, pp. 133–143.
[42] R.-S. Croder, Prediction the Mackey-Glass time series with cascase correlation learning, in: Proc. Connectionist Models Summer School, Carnegic Mellon University, 1990, pp. 117–123.
[43] S. Suresh, N. Sundararajan, P. Saratchandran, Risk sensitive loss functions for sparse multi-category classification problems, Information Sciences 179 (21) (2008) 2621–2638.

Mr. K. Subramanian received his B.Tech degree in electrical and electronics engineering from Amrita University, India, in 2009, and M.Sc. Degree in computer control and automation from Nanyang Technological University, Singapore, in 2010. Currently, he is pursuing his Ph.D. degree from School of Computer Engineering, Nanyang Technological University. His research interests include neural networks, fuzzy logic based systems, machine learning and pattern recognition.

Dr. S. Suresh received the B.E degree in electrical and electronics engineering from Bharathiyar University in 1999, and M.E (2001) and Ph.D. (2005) degrees in aerospace engineering from Indian Institute of Science, India. He was post-doctoral researcher in school of electrical engineering, Nanyang Technological University from 2005 to 2007. From 2007 to 2008, he was in INRIA-Sophia Antipolis, France as ERCIM research fellow. He was in Korea University for a short period as a visiting faculty in Industrial Engineering. From January 2009 to December 2009, he has been with Indian Institute of Technology-Delhi, where he was working as an Assistant Professor in Department of Electrical Engineering. Currently he is working as Assistant Professor in School of Computer Engineering, Nanyang Technological University, Singapore, since 2010. He was awarded best young faculty for the year 2009 by IIT-D. His research interest includes flight control, unmanned aerial vehicle design, machine learning, applied game theory, optimization, and computer vision.