View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Institutional Knowledge at Singapore Management University

Singapore Management University

Institutional Knowledge at Singapore Management University

Research Collection School Of Information Systems

School of Information Systems

9-2014

Online probabilistic learning for fuzzy inference system

Richard Jayadi OENTARYO
Singapore Management University, roentaryo@smu.edu.sg

Meng Joo ER
Nanyang Technological University

San LINN
Nanyang Technological University

Xiang LI
Singapore Institute of Manufacturing Technology

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research

Part of the Databases and Information Systems Commons, and the Numerical Analysis and Scientific Computing Commons

Citation

OENTARYO, Richard Jayadi; ER, Meng Joo; LINN, San; and LI, Xiang. Online probabilistic learning for fuzzy inference system. (2014). *Expert Systems with Applications*. 41, (11), 5082-5096. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/3250

This Journal Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email library@smu.edu.sg.

Author's personal copy

Expert Systems with Applications 41 (2014) 5082–5096

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Online probabilistic learning for fuzzy inference system

Richard J. Oentaryo [a,*], Meng Joo Er [b], San Linn [b], Xiang Li [c]

[a] School of Information Systems, Singapore Management University, 80 Stamford Road, Singapore 178902, Singapore
[b] School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798, Singapore
[c] Manufacturing Execution and Control Group, Singapore Institute of Manufacturing Technology, Nanyang Drive, Singapore 638075, Singapore

A R T I C L E I N F O

Keywords:
Adaptive resonance theory
Bayes' rule
Kalman filter
Neuro-fuzzy system
Online learning

A B S T R A C T

Online learning is a key methodology for expert systems to gracefully cope with dynamic environments. In the context of neuro-fuzzy systems, research efforts have been directed toward developing online learning methods that can update both system structure and parameters on the fly. However, the current online learning approaches often rely on heuristic methods that lack a formal statistical basis and exhibit limited scalability in the face of large data stream. In light of these issues, we develop a new Sequential Probabilistic Learning for Adaptive Fuzzy Inference System (SPLAFIS) that synergizes the Bayesian Adaptive Resonance Theory (BART) and Rule-Wise Decoupled Extended Kalman Filter (RDEKF) to generate the rule base structure and refine its parameters, respectively. The marriage of the BART and RDEKF methods, both of which are built upon the maximum a posteriori (MAP) principle rooted in the Bayes' rule, offers a comprehensive probabilistic treatment and an efficient way for online structural and parameter learning suitable for large, dynamic data stream. To manage the model complexity without sacrificing its predictive accuracy, SPLAFIS also includes a simple procedure to prune inconsequential rules that have little contribution over time. The predictive accuracy, structural simplicity, and scalability of the proposed model have been exemplified in empirical studies using chaotic time series, stock index, and large nonlinear regression datasets.

© 2014 Elsevier Ltd. All rights reserved.

1. Introduction

Soft computing technologies have over the years witnessed great successes in real-world applications (Zadeh, 1994; Lin & Lee, 1996). Two main constituents of soft computing technologies closely related to expert systems research are *fuzzy inference system* (FIS) and *neural network* (NN) (Lin & Lee, 1996). Chiefly, a FIS comprises a set of IF-THEN fuzzy rules that can model complex input–output mappings in the domain of interest, and realize approximate reasoning to deal with uncertainty/imprecision in decision-making. FIS has proven its viability through numerous practical applications ranging from nuclear power plants safety (Ruan & Benítez-Read, 2005) to home appliances such as washing machines (Driankov, Hellendoorn, & Reinfrank, 1996). The design of a FIS, however, traditionally involves extensive human intervention in crafting the rules, which remain static after their initial set-up. By contrast, NN is a trainable dynamical system that can learn, recall, and generalize from data automatically. As such, the NN technology has found its applications in a variety of real-world tasks, such as text and speech recognition, image processing, medical engineering (Haykin, 2009). However, the structure of an NN is generally opaque to the users and difficult to understand.

Given their complementary nature, a promising approach would be to marry the NN and FIS technologies in a way that combines their merits while simultaneously overcoming their individual shortcomings. This gives birth to a powerful hybrid modeling approach called *neuro-fuzzy system* (NFS) (Lin & Lee, 1996; Lughofer, 2011), which is currently under active investigations in expert system research. Specifically, NFS employs NN-based adaptive learning mechanisms to automatically induce, from numerical data, decision logic for inference in the form of intuitive IF-THEN fuzzy rules. As such, it exhibits the learning ability, parallelism, and robustness of NN as well as the human-like logical and approximate reasoning traits of FIS. The NFS approach is more appealing than conventional black-box methods (including NN), which lack the ability to explain the salient knowledge structures in data. NFS also helps eliminate the extensive efforts in manually designing rule-based expert systems, whereby rules are usually fixed and not able to adapt to changes in the task domain.

1.1. From offline to online learning

The learning methods for NFS can be broadly categorized into two main types: *offline* (*batch*) *learning* and *online learning* (Lin & Lee, 1996; Rong, Sundararajan, Huang, & Saratchandran, 2006).

* Corresponding author. Tel.: +65 68085227.
E-mail addresses: roentaryo@smu.edu.sg (R.J. Oentaryo), emjer@ntu.edu.sg (M.J. Er), sanlinn@ntu.edu.sg (S. Linn), xli@SIMTech.a-star.edu.sg (X. Li).

http://dx.doi.org/10.1016/j.eswa.2014.01.034
0957-4174/© 2014 Elsevier Ltd. All rights reserved.

Author's personal copy

Offline learning assumes that all data points are available before learning commences and can be accessed repeatedly, whereas online learning assumes data arrive one at a time from a (possibly infinite) stream. Early works on NFS focused largely on offline learning methods. A classic example is the Adaptive-Network-based Fuzzy Inference System (ANFIS) (Jang, 1993), which has been used in many applications (e.g., Wahab, Quek, Tan, & Takeda, 2009; Yajun et al., 2010). While the original ANFIS included a parameter learning mechanism, its rule base structure (i.e., the node and link constructs) is fixed and needs to be handcoded. In this light, various methods for automated structural learning have been developed (Kim & Kasabov, 1999; Lin, Lin, & Shen, 2001; Cheu, Quek, & Ng, 2012), which typically utilize fuzzy clustering to form the rules and/or initialize their antecedent (premise) parameters prior to an offline parameter tuning process.

In many real-world applications, however, not all training points are available a priori, and often they arrive as a continuous long stream. Offline learning will, in this case, incur high computational cost; as a new data point comes in, the system needs to be retrained all over again, resulting in long training and large memory requirement to retain all previous points. As such, offline learning methods are not suitable for large, nonstationary data, where regime shift exists and a fast response is needed. By contrast, online learning provides a natural means to continuously adapt to new incoming points without a need for retraining. Hence, online learning methods have substantial practical and scalability benefits over offline learning in real-time, dynamic enviroments.

In our research endeavor, we take into account several desiderata of an online learning NFS model. In particular, we adopt the four key characteristics of online learning as advocated in Huang, Saratchandran, and Sundararajan (2005) and Rong et al. (2006):

1. All training observations are *sequentially* (i.e., one-by-one) presented to the system
2. At any time, only *one* observation is seen and learned
3. An observation is *discarded* immediately after learning on that observation is completed
4. The system has no *prior* knowledge on how many total observations will be presented

A similar concept of online learning in data stream mining was recently given in Gama (2010).

1.2. Semi-online vs. online learning

Various approaches dubbed "online learning NFS" have been proposed, but many of them only partially satisfy the above four features and so are not strictly online (Rong et al., 2006)–we hereafter call such approaches *semi-online learning*. For example, the Dynamic Evolving Neural-Fuzzy Inference System (DENFIS) (Kasabov & Song, 2002) normalizes data prior to training, which requires the upper and lower bounds of the data (and implicitly all points) to be known beforehand. Other dynamic NFS methods such as the Self-Organizing Fuzzy Neural Network (SOFNN) (Leng, McGinnity, & Prasad, 2005), Dynamic Fuzzy Neural Network (DFNN) (Wu & Er, 2000) and its variants (Wu, Er, & Gao, 2001; Wang, Er, & Meng, 2009) support inference in real-time. However, to prune or simplify the rule base structure, SOFNN uses the optimal brain surgeon method (Leng et al., 2005), and DFNN the error reduction ratio (Wu & Er, 2000), all of which are not truly online as they need to revisit all data points seen so far.

Likewise, the Flexible Fuzzy Inference System (FLEXFIS) (Lughofer, 2008) and its extended model (Lughofer, 2011) employ an incremental vector quantization method to generate the rules. However, the initial rule base in this approach is generated using some pre-collected data points, and thus it is not fully online either. The more recent Dynamic Parsimonious Fuzzy Neural Network (DPFNN) (Pratama et al., 2013) utilizes the extended self-organizing map and localized least square algorithms to update the rule antecedent and consequent parameters respectively. Nevertheless, the latter algorithm requires several recent data points to be stored in a sliding window, which makes it a semi-online system as well.

To address these limitations, several fully online NFS methods have been developed. The Self-cOnstructing Neural Fuzzy Inference Network (SONFIN) (Juang & Lin, 1998), for instance, generates fuzzy rules on the fly for every point presented. However, SONFIN never removes the rules once created, regardless of whether they are still relevant. Another seminal work is the Evolving Takagi–Sugeno (eTS) (Angelov & Filev, 2004), which creates new rules from high potential data when the existing rules are not sufficiently representative. Similar to SONFIN, however, eTS has no rule pruning procedure, leading to a growing number of inconsequential rules over time. In this light, the Simplified eTS (simpl_eTS) (Angelov & Filev, 2005) and Evolving Extended Takagi–Sugeno (exTS) (Angelov & Zhou, 2006) were developed, whereby rules with low support are disabled (though not explicitly removed from physical memory). An improved eTS was recently presented in Angelov (2010, 2012), which applies the concept of rule age, utility, and support for rule pruning.

An alternative method to online learning NFS is the Sequential Adaptive Fuzzy Inference System (SAFIS) (Rong et al., 2006), which features dynamic rule growing and pruning mechanisms developed in Huang et al. (2005). SAFIS dynamically creates and deletes rules based on significance criteria that are directly linked to the output accuracy, and the rule base parameters are updated using the Extended Kalman Filter (EKF). More recently, inspired by the neurobiological principle of meta-plasticity in associative learning, a Self-reorganizing Fuzzy Associative Machine (SeroFAM) was developed (Tan & Quek, 2010). SeroFAM features fully online learning with a sliding threshold-based self-correcting mechanism that can modulate the learning intensity based on pre- and post-synaptic activities of the rule nodes. Tung, Quek, and Guan (2011) presented a Self-adaptive Fuzzy Inference Network (SaFIN), which uses a new one-pass clustering technique that can recruit new clusters (rules) in each input–output dimension when the existing clusters are insufficient to represent a new incoming data point.

The latest developments in (fully) online learning NFS include the works presented in Lughofer and Buchtala (2013), Lin, Chang, Pal, and Lin (2013), Pratama, Anavatti, Angelov, and Lughofer (2014) and Lin, Chang, and Lin (2014). In Lughofer and Buchtala (2013), a seminal evolving NFS method was developed to address the problem of multi-class classification in an online manner. In this approach, multi-class classification is achieved by building a regression NFS model for each class pair. Pratama et al. (2014) presented a Parsimonious Network-based Fuzzy Inference System (PANFIS) that features a new projection method to provide interpretation for the hyper-ellipsoidal rules as well as an enhanced recursive least square method for pruning inconsequential rules. The online learning methodology was also recently investigated under the framework of type-2 fuzzy system (Lin et al., 2013, 2014), which generalizes conventional (type-1) fuzzy system for better handling of uncertainty. Such approach improves the system robustness in noisy environments, but also results in a more complex and less comprehensible model.

1.3. Research motivation and proposed method

While the above-mentioned full online NFS methods exhibit good modeling accuracy and support continuous adaptation to data stream, there remain several rooms for improvement:

(1) The learning mechanisms of the existing online NFS methods are generally heuristic and lack a formal theoretical

Author's personal copy

foundation. For example, when determining which rule is the most suitable to represent a given data point, the current methods typically utilize distance- or density-based metrics (e.g., Angelov & Filev, 2004; Rong et al., 2006; Tan & Quek, 2010) that solely consider the *likelihood* of the rules matching that data point. Such approach does not take into account the *prior* estimate of the rule importance ("mass"), resulting in a lack of sound probabilistic/statistical interpretation.

(2) The present online NFS methods typically employ either stochastic gradient descent (SGD) (e.g., Juang & Lin, 1998; Tung et al., 2011; Lin et al., 2014) or recursive least square (RLS)-style algorithms (e.g., Angelov & Filev, 2004; Rong et al., 2006; Pratama et al., 2014) for online updating of the model parameters. The SGD method is simple and efficient, but not ideal for online learning due to its sensitivity to noise/outliers and the tendency to easily forget previously learned patterns. On the other hand, the RLS method (aka Kalman filtering) is more robust and has good statistical properties, but often requires computing the full correlation between *all* pairs of model parameters, which is expensive and not scalable. There is thus a need to balance between system robustness and efficiency.

(3) The ever-increasing complexity of the recently developed online learning methods leads not only to high variability in the outcomes of different realizations of an NFS model on different datasets, but also a less practical system that is difficult to implement. Simplicity is much wanted. Meanwhile, empirical studies on the existing online NFS algorithms have so far been conducted on relatively small datasets (typically having $< 10,000$ data points), and their scalability against long data stream has not been comprehensively tested.

In light of the above-mentioned issues, our research goal is to develop a new type of online learning mechanism that conforms to established statistical (probabilistic) principles as well as exhibits an effective tradeoff between simplicity, robustness and scalability. To this end, adapting from the four features of online learning described in Section 1.1 (cf. Huang et al., 2005; Rong et al., 2006), we consider several additional requirements in building our own online NFS framework:

(1) Both the structural and parameter learning mechanisms in the NFS should be statistically tractable and adhere to sound probabilistic principles, in particular to the Bayes rule (Duda, Hart, & Stork, 2001)
(2) The NFS should exhibit parsimonious rule base structure, achieving satisfactory modeling accuracy with concise set of rules and/or model parameters
(3) The NFS is efficient and can gracefully scale up to long data stream
(4) The NFS can adapt to time-varying characteristics in dynamic environments without catastrophically forgetting the knowledge structure previously learned
(5) The NFS can dynamically generate its parameters and structures from scratch with no prior knowledge on the optimal number of rules or parameters.

To realize all these traits, we develop and present in this paper a new *Sequential Probabilistic Learning for Adaptive Fuzzy Inference System* (SPLAFIS). Deviating from the contemporary online NFS methods, our proposed model features a unique learning mechanism that is built upon the sound concepts of *posterior*, *likelihood*, and *prior* probability in Bayesian statistics (Duda et al., 2001). In particular, we cast the structural and parameter learning in SPLAFIS as a *maximum a posteriori* (MAP) problem, where the posterior of the model structure and parameters depends on the data likelihood as well as prior probability estimates. To our best knowledge, this constitutes the first approach providing a comprehensive probabilistic treatment to online learning in fuzzy inference systems.

We summarize the main contributions of our SPLAFIS framework as follows:

(1) We develop a modified *Bayesian Adaptive Resonance Theory* (BART) algorithm (Vigdor & Lerner, 2007) to generate the rule base structure of the SPLAFIS model. The key strength of the BART method is that it facilitates a robust online structural learning method that takes into account for both the density of a given rule (i.e., prior) and the similarity of the rule with an incoming data (i.e., likelihood). Our modified BART approach is not only statistically tractable, but is also able to dynamically evolve the rule base structure from scratch, with no prior knowledge on the rule base structure. The account for both prior and likelihood in the structural learning process also generally leads to a sparse rule base structure with small number of rules.
(2) We devise a scalable and robust parameter learning mechanism termed the *Rule-Wise Decoupled Extended Kalman Filter* (RDEKF), which computes new rule parameter estimates at current time step (i.e., posterior) based on the parameter fitting to the current data (i.e., likelihood) and the parameter estimates from previous time step (i.e., prior). Exploiting the fact that the rules crafted by BART are generally sparse, the RDEKF algorithm performs a localized update of the rule parameters that concentrates on the pairwise correlation of parameters *within* a particular rule. This is far more efficient than the global EKF method (that computes the pairwise parameter correlation *across* all rules), but still maintains the robustness of the global method.
(3) We present a simple procedure to prune inconsequential rules that have little contribution to the overall performance over time. The rule contribution is computed based on a simple criterion that approximates the cumulative posterior of the rules over time. This pruning strategy nicely complements the BART and RDEKF methods in ensuring a good balance between the predictive accuracy and structural simplicity of the rule base constructed.

The synergy of these three mechanisms results in a powerful and statistically-principled online learning NFS model. As such, the SPLAFIS model has many practical implications in real-world applications: it can accurately model the task domains using a compact, intuitive rule base structure, it can gracefully scale up against large nonstationary data stream, and the model setting can be easily adjusted based on the available computational budget. These capacities are evident in our empirical studies on several complex tasks and have been compared with the state-of-the-art learning methods.

The remainder of this paper is organized as follows. In Section 2, we introduce the BART and Kalman filter methods. Section 3 describes the SPLAFIS model architecture, followed by its detailed learning procedure in Section 4. Section 5 presents several empirical studies for evaluating the efficacy of the proposed approach. Finally, Section 6 concludes this paper.

2. Preliminaries

2.1. Bayesian adaptive resonance theory

The BART algorithm (Vigdor & Lerner, 2007) is an extension of the fuzzy ART (FART) (Carpenter, Grossberg, Markuzon, Reynolds, & Rosen, 1992) algorithm that features a fast unsupervised learning for online discovery of clusters from continuous data stream.

Author's personal copy

Armed with the ART learning (Grossberg, 1976), the FART algorithm is able to identify clusters (also called *categories* in the ART framework) by itself without knowing a priori the possible number and type of clusters. The clusters crafted by FART are also sufficiently stable to preserve important past knowledge, but remain adaptable enough to grasp novel information.

One issue with the FART method is the inadequacy of its data representation, i.e., its category selection and learning steps are based on fuzzy ‘min’ and ‘max’ operators, giving hyper-rectangular clusters that are not suitable for real-world data in practice (often with normal distribution). Also, for high-dimensional data, the hyper-rectangular category would cover high volume with very few or no point supporting it (e.g., at the hyper-rectangle corners). Moreover, novel or noisy points may lead to an overproduction of highly-overlapping clusters, aka *category proliferation* (Carpenter et al., 1992).

To mitigate these issues, BART utilizes Gaussian representation in place of the hyper-rectangular category in FART. In BART, a Gaussian category is defined by its *mean vector*, *covariance matrix*, and *prior probability*, reflecting the category’s central of mass, shape of distribution, and dominance with respect to other categories, respectively (Vigdor & Lerner, 2007). This is more intuitive than the weight vector of a FART category (vaguely characterized by its two extreme corners Carpenter et al., 1992), and results in fewer/sparser categories. Moreover, in contrast to FART that uses fuzzy set operators, the category selection in BART is based on the statistically-established Bayes’ rule; it considers not only the distance of a category to a point, but also its dominance to other categories in terms of prior probability (Vigdor & Lerner, 2007). As a result of statistical learning, a category in BART can grow or shrink, whereas the categories in FART only grow over time. In this work, we exploit the capabilities of the BART algorithm in our SPLAFIS model, with several modifications for improvement as later described in Section 4.1.

2.2. Kalman filter

In essence, Kalman filter (Kalman, 1960) formulates sequential learning as an iterative *prediction-correction* process. In the prediction step, time update takes place where one-step ahead prediction of observation is computed. In the correction step, measurement update is taken where correction to the estimate of the current state is calculated. Kalman filter can be viewed as a whitening filter, and is optimal in that it is an unbiased, minimum variance estimator. Moreover, it has a nice probabilistic interpretation derived from Bayesian framework, i.e., it reduces to a maximum a posteriori (MAP) solution (Ho & Lee, 1964). By utilizing the state-space concept, Kalman filter elegantly overcomes the stationarity assumption and gives an exact solution for linear Gaussian prediction/filtering tasks.

In practice, however, the use of the original Kalman filter is limited by the ubiquitous nonlinearity of the physical world. Generally, a nonlinear filtering task consists of finding the conditional probability distribution/density of the state given the observations up to current time. A solution to nonlinear filtering task is by linearization about estimate of the current mean and covariance, a method widely known as *Extended Kalman Filter* (EKF) (Jazwinski, 1970). The EKF algorithm considers, at each cycle, linearization of the nonlinear dynamics around the last consecutive predicted and filtered estimates of the state. The prediction and correction functions need to be differentiable. Accordingly, a matrix of partial derivatives (i.e., Jacobian) can be computed and evaluated with current predicted states at each time step. The matrix can then be used in the Kalman filter equations. This process linearizes the non-linear function around the current estimate.

Our interest is to exploit the unique properties of the EKF so as to update our system’s parameters in a computationally feasible manner without compromising its tracking abilities. The answer can be found in a decoupled form of the EKF, in which the computational complexity is made to suit the requirements of a particular application and of the available computing resources (Puskorius & Feldkamp, 1991). In particular, we exploit the fact that the BART algorithm generally produces sparse rules (clusters) with small overlap, and thus the parameters of one rule are fairly uncorrelated with those of other rules. This leads us to develop the Rule-wise Decoupled EKF (RDEKF), whereby we treat the parameters of a rule as one block and perform parameter refinement on each block locally. This decoupling imposes significant reduction in memory and time requirements, with similar solution quality to that of the (global) EKF. Details on our RDEKF procedure are given in Section 4.2.

3. SPLAFIS architecture

3.1. System structure

The SPLAFIS model, as illustrated in Fig. 1, consists of a five-layer, multi–input–multi–output (MIMO) network structure. Nodes in the input layer, termed input variable nodes IV_i, capture the ith input features of interest x_i. The antecedent layer comprises rule antecedent nodes $A_{i,k}$, each representing a fuzzy linguistic label. Each node R_k in the rule layer represents a fuzzy **IF-THEN** associative rule. Nodes $C_{k,m}$ in the consequent layer correspond to the rule consequent function. Lastly, each output variable node OV_m in the output layer denotes the mth output feature y_m. The total numbers of inputs, outputs, and rules are I, M and K respectively.

Essentially, the aforementioned structure realizes the first-order Takagi–Sugeno–Kang (TSK) fuzzy inference system (Takagi & Sugeno, 1985), which comprises fuzzy rules in the form of (1):

$$\begin{aligned}&\textbf{IF } x_1 \text{ is } A_{1,k} \wedge \ldots x_i \text{ is } A_{i,k} \wedge \ldots x_I \text{ is } A_{I,k}\\ &\textbf{THEN } y_1 = C_{k,1} \wedge \ldots y_m = C_{k,m} \wedge \ldots y_M = C_{k,M}\end{aligned} \tag{1}$$

Here, the antecedent label $A_{i,k}$ of rule R_k is defined using Gaussian membership function as per (2), while the consequent function $C_{k,m}$ is computed using linear Eq. (3):

$$\mu_{A_{i,k}} = \exp\left(-\frac{(x_i - c_{i,k})^2}{2\sigma_{i,k}^2}\right) \tag{2}$$

$$C_{k,m} = \sum_{i=0}^{I} w_{i,k,m} x_i \tag{3}$$

where $c_{i,k}$ and $\sigma_{i,k}$ are the center and width of the Gaussian membership function, respectively, $w_{i,k,m}$ is the consequent weight parameter, and $x_0 = 1$.

3.2. Inference scheme

The decision-making process of the proposed SPLAFIS model involves the following inference scheme, in which the system outputs y_m are computed based on given inputs x_i. First, each input layer node IV_i simply captures x_i and directly propagate it to the next (antecedent) layer. Then, the antecedent layer computes the Gaussian membership degree $\mu_{A_{i,k}}$ of the rule antecedents based on (2). The firing strength of each rule R_k in the rule layer is subsequently computed using the product (fuzzy) T-norm of $\mu_{A_{i,k}}$, as per (4):

$$\mu_{R_k} = \prod_{i=1}^{I} \mu_{A_{i,k}} = \exp\left(-\frac{1}{2}\sum_{i=1}^{I}\frac{(x_i - c_{i,k})^2}{\sigma_{i,k}^2}\right) \tag{4}$$

Next, the consequent outputs $C_{k,m}$ are calculated in the consequent layer via (3), and the normalized firing strength of each rule R_k is calculated using (5):

Author's personal copy

$$\lambda_{R_k} = \frac{\mu_{R_k}}{\sum_{l=1}^{K}\mu_{R_l}} \tag{5}$$

Based on (3) and (5), the overall system outputs y_m are finally inferred using (6):

$$y_m = \sum_{k=1}^{K}\lambda_{R_k}C_{k,m} = \sum_{k=1}^{K}\lambda_{R_k}\left(\sum_{i=0}^{I}w_{i,k,m}x_i\right) \tag{6}$$

4. SPLAFIS learning procedure

The online learning of the SPLAFIS model consists of three phases, namely: *rule construction*, *parameter adjustment*, and *rule pruning*, all carried out for every single data point. SPLAFIS starts with an empty rule base and continually updates its structure and parameters as a new data point arrives. Algorithm 1 outlines the proposed learning procedure, and Sections 4.1, 4.2, 4.3 further elaborate the three phases. A complexity analysis of the procedure is also given in Section 4.4.

Algorithm 1. SPLAFIS Learning Procedure

```
Define: Input-target pair
  $(\vec{x},\vec{t}) = ([x_1 \dots x_i \dots x_I]^T, [t_1 \dots t_m \dots t_M]^T)$, vigilance
  parameter $\rho \in (0,1]$, width scale $\alpha \in (0,1]$, and pruning
  threshold $\beta \in [0,1]$
/* Phase 1: Rule Construction Using BART */
Compute $p(R_k|\vec{x})$ and $\widehat{V}_k$ of all rules $R_k$ via (7) and (10)
$\widehat{V}_{max} \leftarrow \rho\sum_{k=1}^{K}\widehat{V}_k$
$K_{tmp} \leftarrow K$
for $j = 1$ to $K$ do
  $k_p \leftarrow \arg\max_k(p(R_k|\vec{x}))$
  if $\widehat{V}_{k_p} \leqslant \widehat{V}_{max}$ then
    Perform learning on rule $R_{k_p}$ using $\vec{x}$ via (16)–(18)
    $j \leftarrow j + K$ /* break the for loop */
  else
    $p(R_{k_p}|\vec{x}) \leftarrow 0$ /* remove from competition */
  end if
end for
if $p(R_{k_p}|\vec{x}) = 0$ then
  Create a new rule $R_{K+1}$ based on $\vec{x}$ and $\alpha$ via (13)–(15)
  $K \leftarrow K + 1$
end if
/* Phase 2: Parameter Adjustment Using RDEKF */
if $K_{tmp} = K$ then
  Compute the memberships $\mu_{R_k}$ of all rules $R_k$ using (4)
  $k_w \leftarrow \arg\max_k(p(R_k|\vec{x}))$
  Do RDEKF procedure on rule $R_{k_w}$ using $\vec{t}$ via (20)–(22)
else
  Initialize the parameters of the new rule $R_K$ via (33) and (34)
  Reset the covariance matrices of rules $R_1$-$R_{K-1}$ via (35)
end if
/* Phase 3: Rule Pruning */
Compute influences $\psi_k$ of all rules $R_k$ using (36)
$k_p \leftarrow \arg\min_k(\psi_k)$
if $\psi_{k_p} < \beta$ then
  Prune rule $R_{k_p}$ /* prune the least influential rule */
  Delete the covariance matrix of $R_{k_p}$
  $K \leftarrow K - 1$
end if
```

4.1. Rule construction

In this phase, a modified BART algorithm is used to construct rules (categories) on the fly, which comprises three main steps, namely *category choice*, *vigilance test*, and *category learning*. We shall describe each step hereafter.

(1) *Category choice:* In this step, all existing rules compete to represent the current data point. The posterior probability of rule R_k given input point $\vec{x} = [x_1, \dots, x_i, \dots, x_I]^T$ is computed using the Bayes' rule (Duda et al., 2001), as defined in (7):

$$p(R_k|\vec{x}) = \frac{p(\vec{x}|R_k)p(R_k)}{\sum_{l=1}^{K}p(\vec{x}|R_l)p(R_l)} \tag{7}$$

where K is the total number of rules, $p(R_k)$ is the estimated prior probability of rule R_k, and $p(\vec{x}|R_k)$ is the estimated likelihood of R_k with respect to $\vec{x}$. The prior probability and likelihood are defined in (8) and (9) respectively:

$$p(R_k) = \frac{N_k}{\sum_{l=1}^{K}N_l} \tag{8}$$

$$p(\vec{x}|R_k) = \frac{1}{(2\pi)^{I/2}\widehat{V}_k^{1/2}}\exp\left(-\frac{1}{2}\sum_{i=1}^{I}\frac{(x_i - c_{i,k})^2}{\sigma_{i,k}^2}\right) \tag{9}$$

where N_k is the number of times R_k has won the competition, and $\widehat{V}_k$ is the hypervolume of data space covered by R_k, estimated via the product of sides as defined in Vigdor and Lerner (2007) and (10):

$$\widehat{V}_k = \prod_{i=1}^{I}\sigma_{i,k}^2 \tag{10}$$

For each competition step, the winning rule R_{k_p} is defined as one with the *maximum a posteriori* (MAP) as per (11):

$$k_p = \arg\max_k(p(R_k|\vec{x})) \tag{11}$$

That is, the winning rule R_{k_p} is either more populated (i.e., higher $p(R_k)$) than the other rules, or more likely to be true (i.e., higher $p(\vec{x}|R_k)$ as it is the closest to $\vec{x}$), or both. The MAP criterion is expected to select a winning rule more accurately than using likelihood or prior probability alone. For example, it may prefer a rule with higher prior probability over another rule, though the normalized distance (i.e., the term $\sum_{i=1}^{I}\frac{(x_i-c_{i,k})^2}{\sigma_{i,k}^2}$ in (9)) between the former to $\vec{x}$ is larger.

(2) *Vigilance test:* The goal of this test is to ensure that the chosen rule R_{k_p} is limited in size. That is, the test restricts the hypervolume (coverage) $\widehat{V}_{k_p}$ of the chosen rule to the maximal hypervolume $\widehat{V}_{max}$ allowed for a rule, as per (12):

$$\widehat{V}_{k_p} \leqslant \widehat{V}_{max} = \rho\sum_{k=1}^{K}\widehat{V}_k \tag{12}$$

where $\rho \in (0,1]$ is the vigilance parameter. In practice, we typically choose ρ to be $\leqslant 0.1$. When R_{k_p} meets (12), category learning is performed (as described shortly). Otherwise, the rule is removed from the competition for the current point $\vec{x}$ (e.g., by resetting its posterior probability $p(R_k|\vec{x}) = 0$), and a search for another rule with high posterior probability that complies with (12) is conducted. If all rules fail the vigilance test, then a new rule R_{K+1} is created and its center vector $\vec{c}_{K+1} = [c_{1,K+1}, \dots, c_{i,K+1}, \dots, c_{I,K+1}]^T$, width vector $\vec{\sigma}_{K+1} = [\sigma_{1,K+1}, \dots, \sigma_{i,K+1}, \dots, \sigma_{I,K+1}]^T$, and winning count N_{K+1} are initialized using (13)–(15), respectively:

$$\vec{c}_{K+1} = \vec{x} \tag{13}$$

Author's personal copy

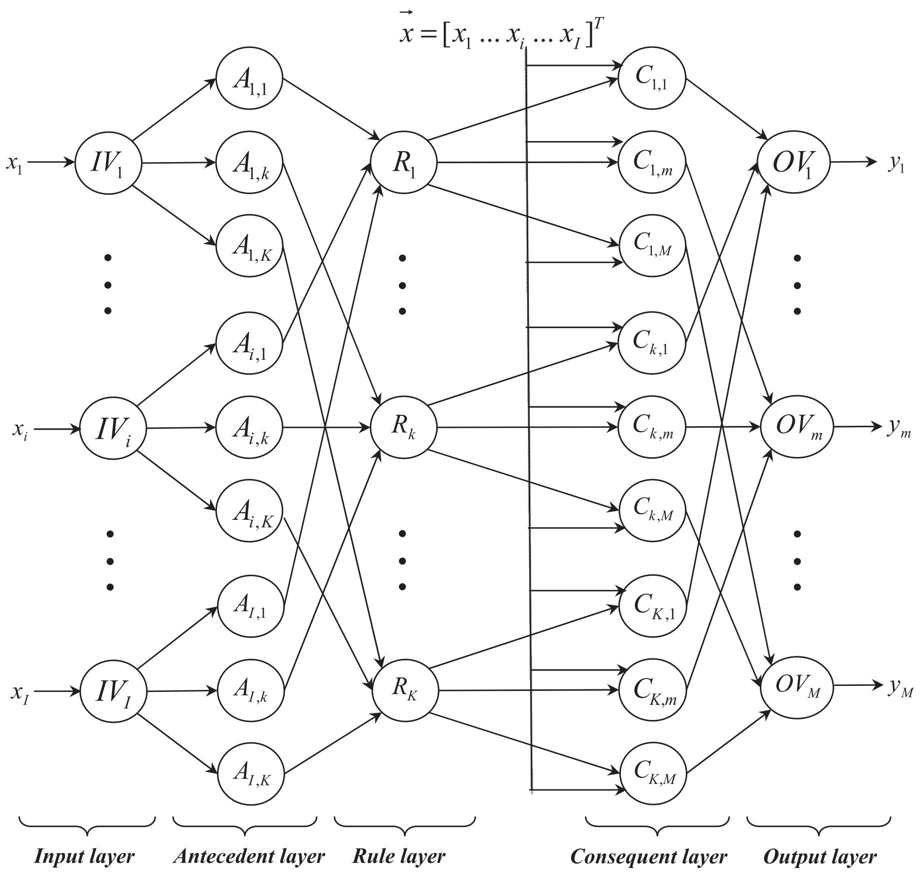


Fig. 1. Structure of the proposed SPLAFIS model.

$$\vec{\sigma}_{K+1} = \alpha \times \min_{k=1}^{K} \|\vec{x} - \vec{c}_k\| \tag{14}$$

$$N_{K+1} = 1 \tag{15}$$

where $\|.\|$ is the Euclidean norm and $\alpha \in (0,1]$ is the user-defined width scale parameter.

(3) *Category learning:* When a chosen rule R_{k_p} passes the vigilance test, the parameters $\vec{c}_{k_p}, \vec{\sigma}_{k_p}, N_{k_p}$ of that rule are updated using (16)–(18) respectively:

$$\vec{c}^{new}_{k_p} = \vec{c}^{old}_{k_p} + \frac{\vec{x} - \vec{c}^{old}_{k_p}}{N^{old}_{k_p} + 1} \tag{16}$$

$$\left(\vec{\sigma}^{new}_{k_p}\right)^2 = \left(\vec{\sigma}^{old}_{k_p}\right)^2 + \frac{\left(\vec{x} - \vec{c}^{new}_{k_p}\right)^2 - \left(\vec{\sigma}^{old}_{k_p}\right)^2}{N^{old}_{k_p} + 1} \tag{17}$$

$$N^{new}_{k_p} = N^{old}_{k_p} + 1 \tag{18}$$

The update formulae in (16) and (17) are essentially expanded from sequential maximum likelihood estimation for a single Gaussian to the multidimensional case (Vigdor & Lerner, 2007).

The above modified BART algorithm differs from the original BART in several ways:

(1) In the original BART, each category is defined by a full covariance matrix (in addition to the mean vector). From a fuzzy rule viewpoint, though, such representation is not interpretable; a fuzzy rule is usually viewed as conjunction of input feature-wise fuzzy membership functions. To improve the semantics, our modified BART uses a diagonal covariance matrix instead. This simplifies the likelihood calculation and reduces the computational cost, though it may lose flexibility in the cluster representation. Regardless, our focus is the prediction performance at the output rather than clustering quality at the inputs. The later RDEKF procedure (cf. Section 4.2) will be used to tweak the cluster parameters.

(2) To restrict the size of a category during learning, the original BART uses an absolute maximum hypervolume parameter, which is positive and unbounded. While intuitive, finding a suitable configuration for such unbounded parameter is difficult in practice. In our modified BART approach, we instead define the maximum hypervolume as a fraction $\rho \in (0,1]$ of the sum of the hypervolumes of all categories, which is easier to configure.

(3) When a new rule is created, the original BART initializes the rule's width (variance) vector using a small value, resulting in a small hyper-spherical category that has low coverage. To improve the initialization, our modified BART method sets the initial widths of a new category based on the (Euclidean) distance of the current point to the nearest category, similar to Rong et al. (2006).

These modifications are reflected in the formulae (9), (12) and (14), respectively.

4.2. Parameter adjustment

This phase consists of two alternative scenarios. The first scenario is when no new rule is created in the rule construction phase

Author's personal copy

(i.e., when $K_{tmp} = K$ in Algorithm 1) and involves updating the parameters of the winning rule that passes the vigilance test. The second case is about how to initialize the consequent parameters of the new rule, which is created when all the other (existing) rules fail the vigilance test. The two scenarios are detailed hereafter:

(1) *Winning rule update:* In this scenario, the parameters of the winning rule (selected by (11)) are adjusted via the RDEKF procedure, an adaptation of Puskorius and Feldkamp (1991). Let $\varphi(n) = [\vec{h}(1), \ldots, \vec{h}(n)]$ be a set of observations up to time step n, whereby each observation $\vec{h}(n)$ comprises input vector $\vec{x}(n)$ and target vector $\vec{t}(n)$. The RDEKF algorithm aims at maximizing the posterior $p(\varphi(n)|\theta(n))$, where $\theta(n)$ is the set of rule base parameters at time n. Using Bayes' rule, this can be written as (19):

$$p(\theta(n)|\varphi(n)) = \frac{p(\vec{h}|\theta(n))p(\theta(n)|\varphi(n-1))}{p(\vec{h}(n)|\varphi(n-1))} \tag{19}$$

Treating the parameters of each rule as one block and assuming that $p(\vec{h}|\theta(n))$ and $p(\theta(n)|\varphi(n-1))$ follow a Gaussian distribution, we arrive at the RDEKF updating procedure as per (20)–(22):

$$\mathbf{G}_{k_w}(n) = \mathbf{P}_{k_w}(t-1)\mathbf{H}_{k_w}(n)\left[\mathbf{R} + \mathbf{H}^T_{k_w}(n)\mathbf{P}_{k_w}(t-1)\mathbf{H}_{k_w}(n)\right]^{-1} \tag{20}$$

$$\mathbf{P}_{k_w}(n) = \left[\mathbf{I}_{Z\times Z} - \mathbf{G}_{k_w}(n)\mathbf{H}^T_{k_w}(n)\right]\mathbf{P}_{k_w}(t-1) \tag{21}$$

$$\vec{\theta}_{k_w}(n) = \vec{\theta}_{k_w}(t-1) + \mathbf{G}_{k_w}(n)[\vec{t} - \vec{y}] \tag{22}$$

where $\mathbf{G}_{k_w}(n)$, $\mathbf{P}_{k_w}(n)$, $\mathbf{H}_{k_w}(n)$ and $\vec{\theta}_{k_w}(n)$ are the Kalman gain, covariance matrix, Jacobian matrix, and parameter vector of rule R_{k_w} at time n, respectively. $\mathbf{R}$ is the observation noise variance, $\mathbf{I}_{Z\times Z}$ is identity matrix such that $Z = 2I + (I+1)M$ is the length of $\vec{\theta}_{k_w}(n)$, and $\vec{t} = ([t_1, \ldots, t_m, \ldots, t_M]^T$ and $\vec{y} = ([y_1, \ldots, y_m, \ldots, y_M]^T$ are the target and system output vectors, respectively. For simplicity and to avoid an extra free parameter, we set $\mathbf{R}$ as identity matrix (i.e., $\mathbf{R} = \mathbf{I}_{M\times M}$).
The parameter vector $\vec{\theta}_{k_w}(n)$ is decomposed into $\vec{\theta}_{k_w}(n) = \left[\vec{w}^T_{k_w}, \vec{c}^T_{k_w}, \vec{\sigma}^T_{k_w}\right]^T$, which respectively correspond to (23)–(25):

$$\vec{w}_{k_w} = [w_{0,k_w,1} \ldots w_{i,k_w,1} \ldots w_{I,k_w,1}, w_{0,k_w,m} \ldots w_{i,k_w,m} \ldots w_{I,k_w,m}, w_{0,k_w,M} \ldots w_{i,k_w,M} \ldots w_{I,k_w,M}]^T \tag{23}$$

$$\vec{c}_{k_w} = [c_{1,k_w} \ldots c_{i,k_w} \ldots c_{I,k_w}]^T \tag{24}$$

$$\vec{\sigma}_{k_w} = [\sigma_{1,k_w} \ldots \sigma_{i,k_w} \ldots \sigma_{I,k_w}]^T \tag{25}$$

Meanwhile, the Jacobian matrix $\mathbf{H}_{k_w}(n)$ is given by (26):

$$\mathbf{H}_{k_w}(n) = \begin{bmatrix} \frac{\partial C_{k_w,1}}{\partial \vec{w}_{k_w,1}} & \ldots & 0 & \ldots & 0 \\ 0 & \ldots & \frac{\partial C_{k_w,m}}{\partial \vec{w}_{k_w,m}} & \ldots & 0 \\ 0 & \ldots & 0 & \ldots & \frac{\partial C_{k_w,M}}{\partial \vec{w}_{k_w,M}} \\ \frac{\partial C_{k_w,1}}{\partial \vec{c}_{k_w}} & \ldots & \frac{\partial C_{k_w,m}}{\partial \vec{c}_{k_w}} & \ldots & \frac{\partial C_{k_w,M}}{\partial \vec{c}_{k_w}} \\ \frac{\partial C_{k_w,1}}{\partial \vec{\sigma}_{k_w}} & \ldots & \frac{\partial C_{k_w,m}}{\partial \vec{\sigma}_{k_w}} & \ldots & \frac{\partial C_{k_w,M}}{\partial \vec{\sigma}_{k_w}} \end{bmatrix} \tag{26}$$

where the gradient vectors are defined in (27)–(29):

$$\frac{\partial y_m}{\partial \vec{w}_{k_w,m}} = \left[\frac{\partial y_m}{\partial w_{0,k_w,m}} \ldots \frac{\partial y_m}{\partial w_{i,k_w,m}} \ldots \frac{\partial y_m}{\partial w_{I,k_w,m}}\right]^T \tag{27}$$

$$\frac{\partial y_m}{\partial \vec{c}_{k_w}} = \left[\frac{\partial y_m}{\partial c_{1,k_w}} \ldots \frac{\partial y_m}{\partial c_{i,k_w}} \ldots \frac{\partial y_m}{\partial c_{I,k_w}}\right]^T \tag{28}$$

$$\frac{\partial y_m}{\partial \vec{\sigma}_{k_w}} = \left[\frac{\partial y_m}{\partial \sigma_{1,k_w}} \ldots \frac{\partial y_m}{\partial \sigma_{i,k_w}} \ldots \frac{\partial y_m}{\partial \sigma_{I,k_w}}\right]^T \tag{29}$$

and each element in the vectors is computed using (30)–(32):

$$\frac{\partial y_m}{\partial w_{i,k_w,m}} = \lambda_{R_{k_w}} x_i \tag{30}$$

$$\frac{\partial y_m}{\partial c_{i,k_w}} = \lambda_{R_{k_w}} (C_{k_w,m} - y_m) \frac{(x_i - c_{i,k_w})}{\sigma^2_{i,k_w}} \tag{31}$$

$$\frac{\partial y_m}{\partial \sigma_{i,k_w}} = \lambda_{R_{k_w}} (C_{k_w,m} - y_m) \frac{(x_i - c_{i,k_w})^2}{\sigma^3_{i,k_w}} \tag{32}$$

The full derivations for (30)–(32) can be found in Appendix A. Notably, the RDEKF procedure is able to produce a solution quality approaching that of the global (original) EKF, but with much less time and memory requirements. Whereas the global EKF tracks the correlations between every pair of rule parameters, the RDEKF procedure updates the parameters of each rule locally. Indeed, in practice the correlations between parameters of different rules tend to be very small (i.e., ≈ 0), thanks to the small overlap among the rules produced by the BART clustering. Fig. 2(a) and (b) illustrate the difference between the EKF and RDEKF procedures. The former requires to store and update a full covariance matrix of size $Z^2 \times K^2$, while the latter maintains only K (sub) matrices of size $Z \times Z$ each (see the grey blocks), which is much smaller.

(2) *New rule initialization:* When a new rule is created, its consequent parameters are set as the weighted average of the consequent parameters of the existing rules, where the weights are normalized rule strengths as per (5). The consequent parameters of a new rule R_{K+1} are initialized via (33):

$$w_{i,K+1,m} = \sum_{k=1}^{K} \lambda_{R_k} w_{i,k,m} \tag{33}$$

Here the key idea is to use the existing rule base parameters as an estimate for the initial consequent parameters of the new rule; this introduces smoother and more stable change to the (next) RDEKF update than initializing with the target vector (Angelov & Filev, 2004). Meanwhile, the parameters of the other rules remain the same and are simply inherited from the previous time step.

On the other hand, the covariance matrix of the new rule R_{K+1} is initialized using (34):

$$\mathbf{P}_{K+1}(n) = \mathbf{I}_{Z\times Z} \tag{34}$$

At the same time, the covariance matrices of all the other rules R_k $(k \in 1, \ldots, K)$ are reset via (35):

$$\mathbf{P}_k(n) = \left(\frac{K^2+1}{K^2}\right)\mathbf{P}_k(t-1) \tag{35}$$

In this manner, the covariance matrix belonging to the new rule R_{K+1} is initialized as usual (here using identity matrix), and the covariance matrices of the remaining rules R_k are updated by multiplication of $\left(\frac{K^2+1}{K^2}\right)$. The latter correction is done to reflect the contribution that the new rule would have if it existed from the beginning. The mathematical rationale for this is further explained in Appendix B.

Author's personal copy

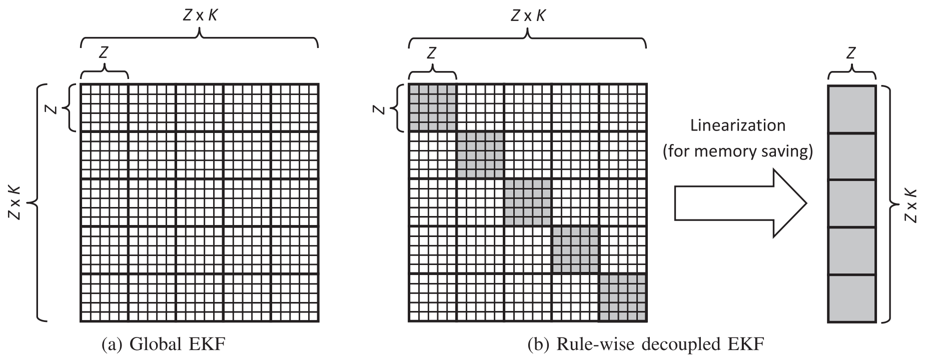


Fig. 2. Error covariance matrices in global and decoupled EKF.

4.3. Rule pruning

A simple and intuitive procedure for rule pruning is carried out in this phase. The key idea is to remove a rule that has the least contribution since it was first created. The contribution φ_k of each rule R_k at time instance t is estimated using (36):

$$\varphi_k = \frac{\sum_{t'=t_k}^{t} \mu_{R_k}(t')}{t - t_k} \in [0,1] \tag{36}$$

where the numerator is the cumulative firing strength of rule R_k, as per (4), and t_k is the time instance at which R_k was created. The numerator can be viewed as approximation to the cumulative posterior probability $p(R_k|\vec{x})$, which measures the relevance of rule R_k over time with respect to the input features $\vec{x}$. Here the cumulative strength is updated recursively, with initial condition $\mu_{R_k}(t_k) = 1$. Next, the least influential rule R_{k_q} is identified using (37):

$$k_q = \arg\min_k(\varphi_k) \tag{37}$$

and it is pruned if $\varphi_{k_q} < \beta$, where $\beta \in [0,1]$ is a user-defined pruning threshold. When pruning takes place, the covariance matrix $\mathbf{P}_{k_q}$ of the pruned rule R_{k_q} is deleted accordingly. This imposes a minimal effect on the overall modeling accuracy while keeping the size of the rule base bounded. Over time, this also aids the SPLAFIS model in adapt to the changes in the task domain better. Typically, β can be set to be a small value (e.g., $\beta \leqslant 0.01$).

4.4. Complexity analysis

Using the notations in the previous sections, Table 1 provides a summary of the (worst) time complexity of the above three learning phases for a single data point. It can be seen that the computational load of the SPLAFIS model lies in the rule construction and

Table 1
Time complexity of the SPLAFIS learning phases.

Learning phase	Time complexity	Description
Rule construction	$O(K^2 + I \times K)$	Find a winning rule and adjust its parameters only if it passes the vigilance test. If no rule meets the test, a new rule is created.
Parameter adjustment	$O(I^2 \times M^3)$	Carry out RDEKF procedure to fine-tune the parameters of the best-fit rule, or otherwise initialize those of the newly created rule.
Rule pruning	$O(K + I \times M)$	Prune the least influential rule if the ratio of its cumulative firing strength and life span is below a certain specified level

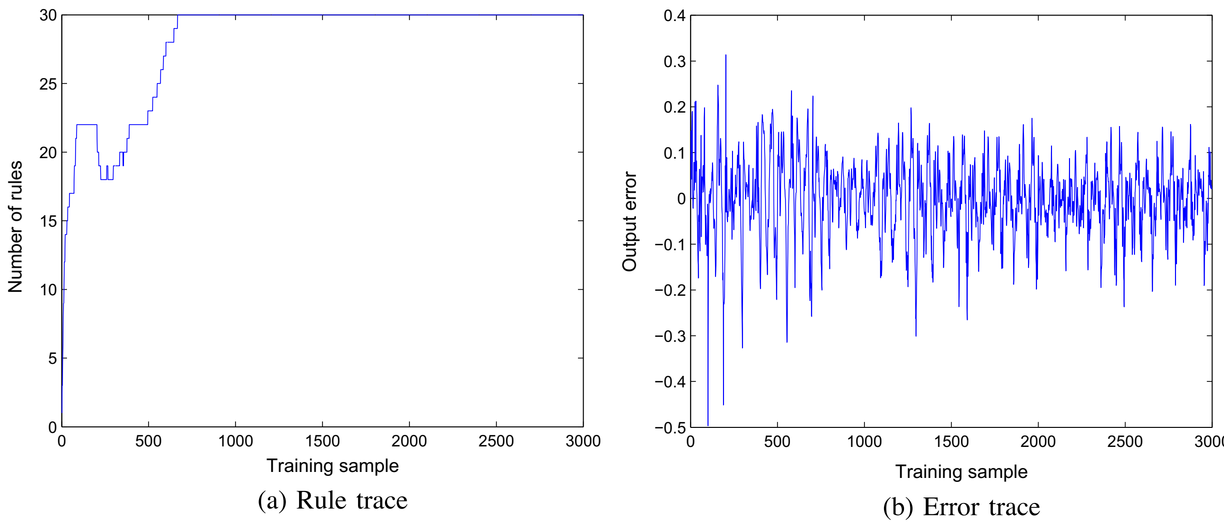


Fig. 3. Training traces for Mackey–Glass data.

Author's personal copy

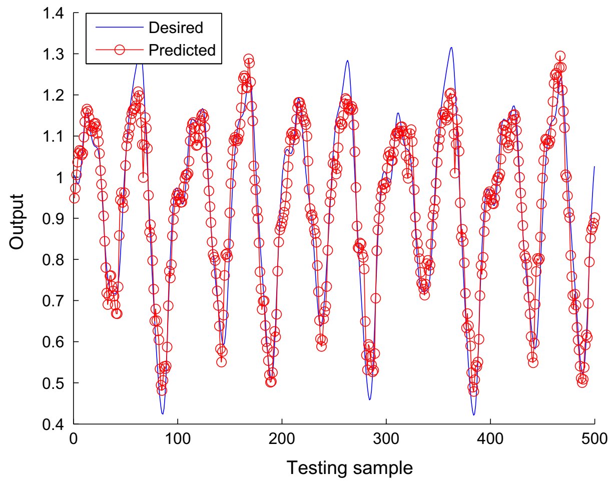


Fig. 4. Actual and predicted outputs for Mackey–Glass data.

Table 2
Benchmark results on Mackey–Glass data.

Method	Type	Learning mode	No. of rules/nodes	Test NDEI
EFuNN	TSK-1	Semi-online	193	0.401
DENFIS	TSK-1	Semi-online	58	0.276
eTS	TSK-1	Online	99	0.356
SAFIS	TSK-0	Online	21	0.380
SeroFAM	Mamdani	Online	52	0.345
SPLAFIS	TSK-1	Online	30	0.279

TSK-0/1: zero/first-order Takagi–Sugeno–Kang fuzzy system. Mamdani: Mamdani fuzzy system.

parameter adjustment steps, especially when the number of inputs I and number of rules K are large. For a training data stream comprising N points, the total time complexity of the learning process is given by (38):

$$O\left(\left[K^2 + I \times K + I^2\right] \times N\right) \tag{38}$$

Such complexity is fairly low nonetheless, as in most cases K is much smaller than N (i.e., $K \ll N$). This may be attributed to the vigilance test (12) and pruning procedure (36), which together restrict the size of the rule base. Also, since the SPLAFIS learning procedure assumes no prior domain knowledge and that each training point is fed into the system only once (i.e. no data revisit), its speed performance compares favorably to that of other conventional NFS approaches (e.g., Jang, 1993; Wu & Er, 2000; Kasabov, 2001; Kasabov & Song, 2002; Wang et al., 2009). This benefit is evident in our simulation studies (see e.g., Section 5.3).

On the other hand, the overall space complexity of the learning procedure is given by (39):

$$O\left(K \times I^2\right) \tag{39}$$

which can be largely attributed to the size of the covariance matrix $\mathbf{P}_k(n)$, as given by (21) and (34). This requirement is reasonable nevertheless, recalling that $K \ll N$. It is also an order-of-magnitude lower than that of the global (original) EKF algorithm (Kadirkamanathan & Niranjan, 1993), which requires storing the full covariance matrix that induces a total complexity of $O\left(K^2 \times I^2\right)$.

5. Experimental studies

To evaluate the efficacy of the SPLAFIS model, we performed a series of experiments on both static and dynamic tasks. All experiments were done in MATLAB© R2010b, running on an Intel Core i5 machine with 4 GB RAM. In each case study, the vigilance parameter ρ, width scale α, and pruning threshold β were chosen empirically from $[0.001, 0.01]$, $[0.1, 0.9]$, and $[0.001, 0.01]$, respectively. Detailed results and analysis are presented in Section 5.1, 5.2 and 5.3.

5.1. Standard benchmark: chaotic Mackey–Glass time series

As our preliminary study to verify the efficacy of the proposed SPLAFIS model in online modeling of volatile data, we experimented with the chaotic Mackey–Glass time series (Mackey & Glass, 1977) data. The data has been used to benchmark the approximation abilities of various neural and fuzzy neural techniques (Platt, 1991; Kasabov & Song, 2002; Angelov & Filev, 2004; Rong et al., 2006). The time series was generated using a differential delay equation in (40):

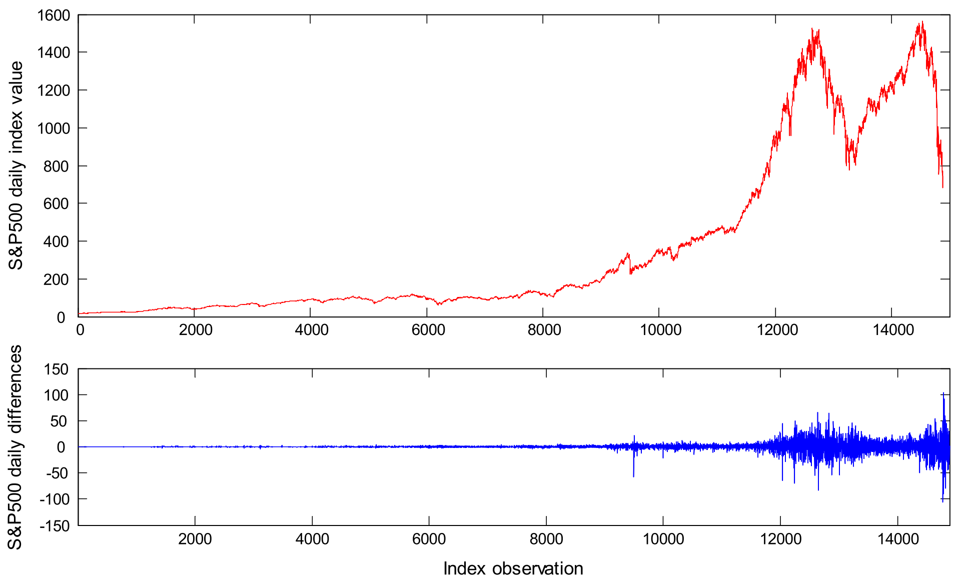


Fig. 5. S&P-500 daily index from 3 January 1950 to 12 March 2009.

Author's personal copy

Table 3
Benchmark results on S&P-500 data.

Method	Type	Learning mode	No. of rules	Test NDEI
EFuNN	TSK-1	Semi-online	114.3	0.1544
DENFIS	TSK-1	Semi-online	6.0	0.0200
ANFIS	TSK-1	Offline	32.0	0.0154
eTS	TSK-1	Online	14.0	0.0155
SeroFAM	Mamdani	Online	29.9	0.0272
SPLAFIS	TSK-1	Online	10.2	0.0153

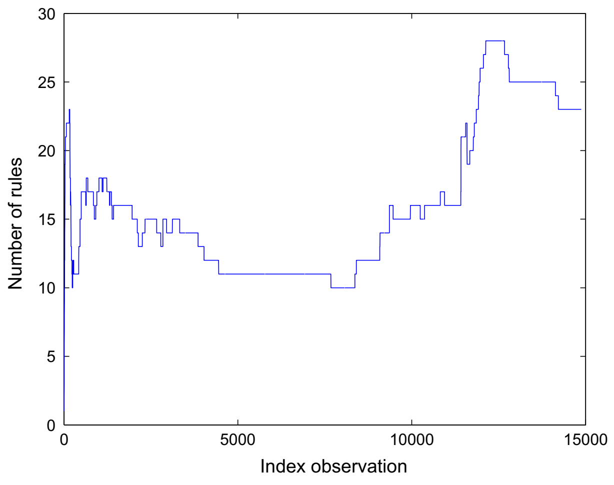
TSK-0/1: zero/first-order Takagi–Sugeno–Kang fuzzy system. Mamdani: Mamdani fuzzy system.

$$\frac{dy(n)}{dt} = \frac{0.2y(t-\tau)}{1+x^{10}(t-\tau)} - 0.1y(n) \tag{40}$$

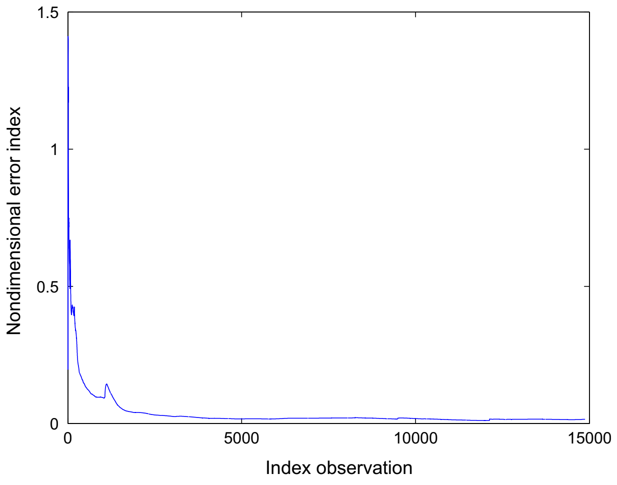
Originally, the equation was used to illustrate the complex dynamics in physiological control systems by way of bifurcations in the dynamics (Mackey & Glass, 1977). This suggests that many physiological disorders (or dynamical diseases) can be characterized by changes in qualitative features of dynamics.

Following the setup in Angelov and Filev (2004), 6000 observations were generated via the 4th-order Runge–Kutta method using a step-size 0.1 and delay coefficient $\tau = 17$. The observations for $201 \leqslant t \leqslant 3200$ and $5001 \leqslant t \leqslant 5500$ were uniformly distributed in the range $[0.4, 1.4]$, and were used as the train and test sets respectively. The goal is to predict a few steps ahead of the current time t. Four input features were used: $[y(n), \; y(t-6), \; y(t-12), y(t-18)]$, and the output to be predicted was $y(t+85)$.

The training traces of the SPLAFIS model are summarized in Fig. 3. Fig. 3(a) depicts the evolution of the rule base, which grows and shrinks at the beginning of training process and stabilizes toward the end. Fig. 3(b) shows the trace of the output error during the course of training. As shown, the output error tends to decrease as more data points are fed in, which demonstrates the ability to learn the data more accurately over time. Fig. 4 compares the testing output forecasted by SPLAFIS and the target (testing) series. As shown, the system manages to approximate the target series well.



(a) Rule trace



(b) Error trace

Fig. 6. Forecasting trace for S&P-500 data.

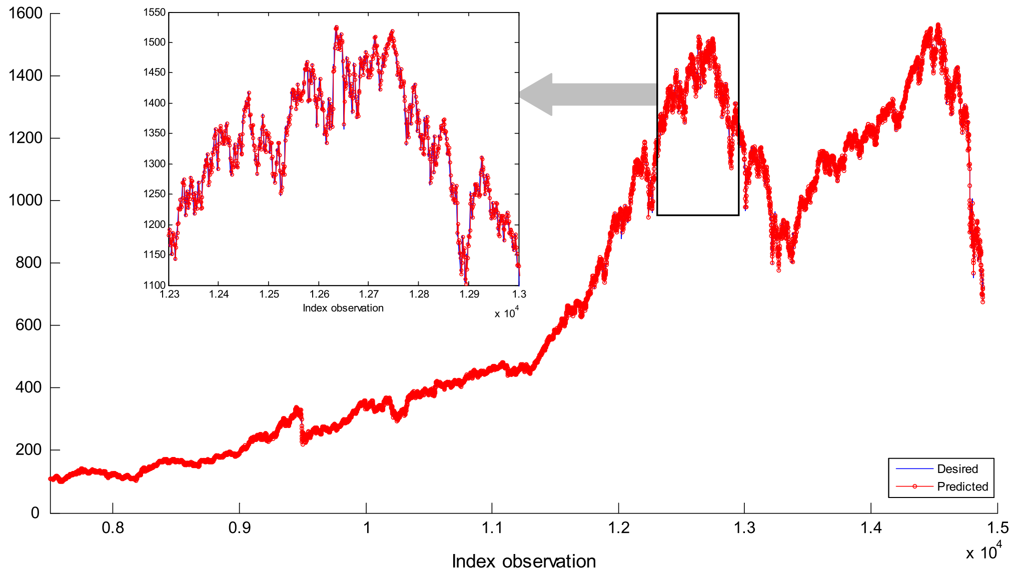


Fig. 7. Actual and forecast outputs for S&P-500 data.

Author's personal copy

The consolidated results of the SPLAFIS model are presented in Table 2 and compared with those *published results* of popular neuro-fuzzy approaches: EFuNN (Kasabov, 2001), DENFIS (Kasabov & Song, 2002), eTS (Angelov & Filev, 2004), and SAFIS (Rong et al., 2006). The generalization performance of the system on the test set is evaluated based on *non-dimensional error index* (NDEI), i.e., the root mean square error on the test set divided by the standard deviation of the target (testing) series (Kasabov & Song, 2002).

From Table 2, we can conclude that SPLAFIS produces competitive performances. It achieves relatively low NDEI $= 0.279$, and performs well with moderate 30 rules. Although DENFIS gives comparable NDEI in this study, its rule base size is nearly twice that of SPLAFIS, implying higher storage requirement. Moreover, the DENFIS learning procedure normalizes data before training, which assumes a prior knowledge of the upper and lower bounds of the data (and thus all training points). As such, DENFIS is not a fully online system. On the other hand, SPLAFIS yields slightly more rules than SAFIS. Regardless, the former's predictive accuracy is substantially better than the latter. In sum, the results show that SPLAFIS achieves good balance between model accuracy and complexity, while yielding competitive performance with respect to contemporary NFS methods.

5.2. Time-varying data modeling: stock index forecasting

This study aims at investigating the self-organizing ability of the SPLAFIS model in non-stationary time-series forecasting task based on the Standard and Poor's 500 (S&P-500) market index. Following (Tan & Quek, 2010), the dataset comprises 60 years of daily index values collected from Yahoo! Finance website on the ticker symbol "ĜSPC" from 3 January 1950 to 12 March 2009. In total, there are 14,893 data (stock index) points. Fig. 5 shows the time-varying behavior of the index values with a nonuniform distribution in the range of [16.66, 1565.15]. In this study, we focus on the trajectory shifts of the index values after the 1980s (i.e., the second half of Fig. 5), where there is a noticeable increase in the volatility of daily differences.

We carried out an online simulation of the daily forecast of the S&P-500 index. Four input features were used: $[y(t-4), y(t-3), y(t-2), y(t-1), y(n)]$, where $y(n)$ is the absolute value of S&P-500 index at the tth day, and the output feature to be forecast is $y(t+1)$. The result of the SPLAFIS model for this stepwise task is shown in Table 3, along with the results of NFS methods including EFuNN (Kasabov, 2001), DENFIS (Kasabov & Song, 2002), ANFIS (Jang, 1993), eTS (Angelov & Filev, 2004), and SeroFAM (Tan & Quek, 2010).

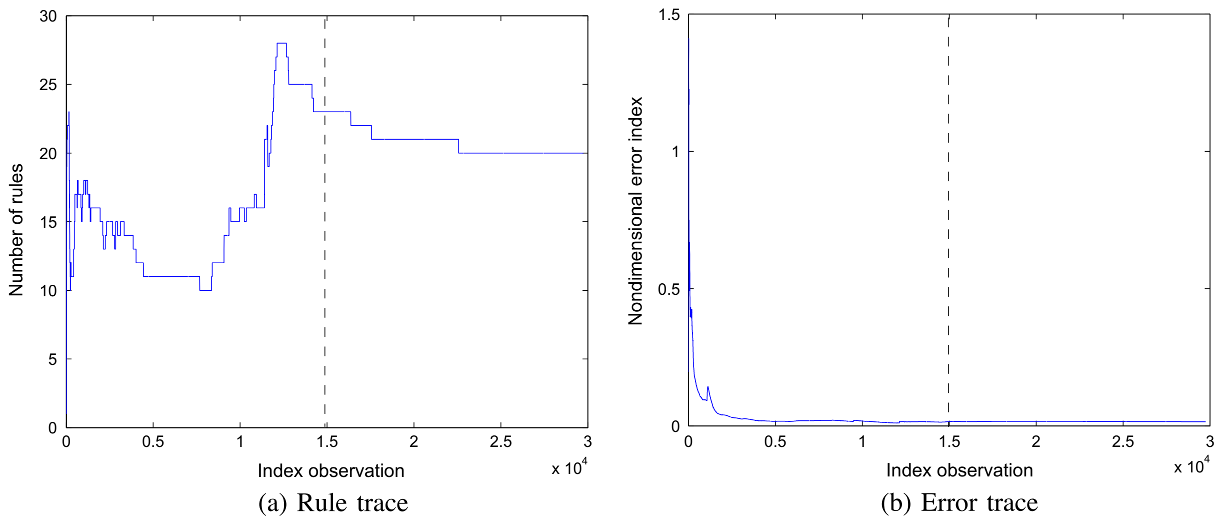


Fig. 8. Forecasting trace for S&P-500 with market reversal.

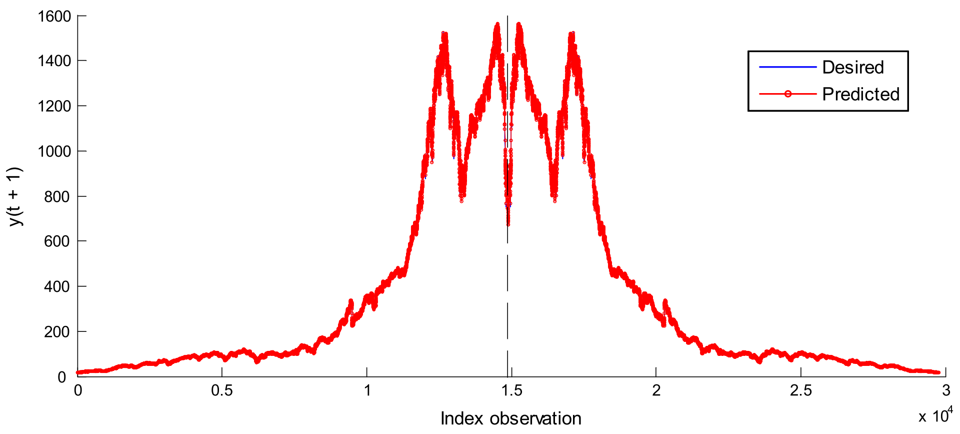


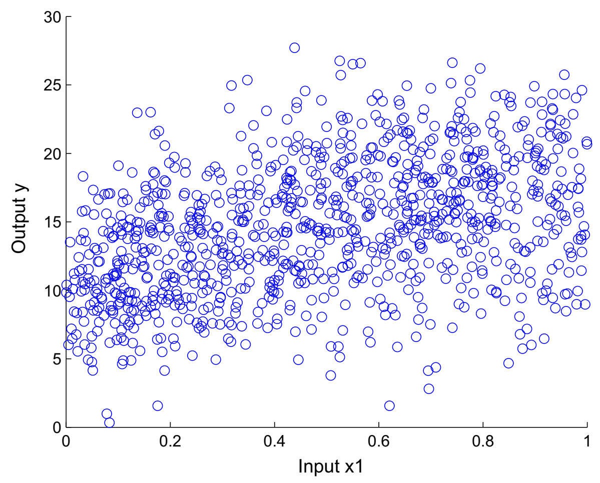
Fig. 9. Desired and predicted outputs for S&P-500 with simulated market reversal.

Author's personal copy

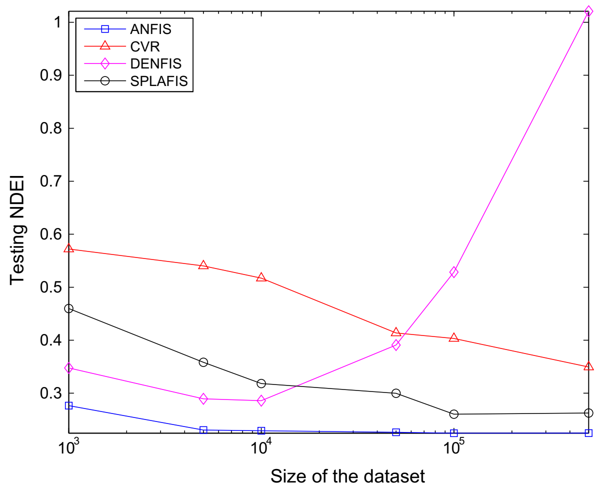
We can see that SPLAFIS gives competitive performance in terms of the test NDEI, comparable to that of ANFIS and eTS. However, SPLAFIS yields fewer rules than the other two. Meanwhile, although DENFIS has fewer rules than SPLAFIS, the former has substantially higher NDEI. It is also important to note that ANFIS is an offline system, while DENFIS and EFuNN are not fully online learning systems. That is, DENFIS requires normalization of data prior to training, while the rule layer in EFuNN only self-organizes the rule nodes (i.e., the membership functions are fixed during evolution). By contrast, SPLAFIS and SeroFAM are able to train and predict recursively without prior knowledge of the complete dataset at any time. Clearly from the results, however, SPLAFIS is superior to SeroFAM in terms of rule base size and forecasting accuracy (i.e., NDEI).

Fig. 6(a) illustrates the structural reorganization processes in the SPLAFIS rule base over 60 years. Major reorganization takes place in the first 4000 days (about 16 years), whereby the system initially tries to settle from a coarse to reasonable rule structure, and in the last 5000 days (about 20 years), whereby high volatility occurs in the S&P index. On the other hand, the rule base structure is stable for the 6000 days in between, indicating low volatility during this period. Fig. 6(b) shows that, despite the high initial error, the daily NDEI value stabilizes over the period of 60 years. Altogether, these results show the ability of the SPLAFIS to closely track the major trajectory shifts in time-varying environments. This can be further justified by comparing the actual and forecast S&P-500 index values in Fig. 7, i.e., the system can nicely follow the changes in the index value regime, including the two peaks (around year 2000 and 2007) and valley in between (around year 2003).

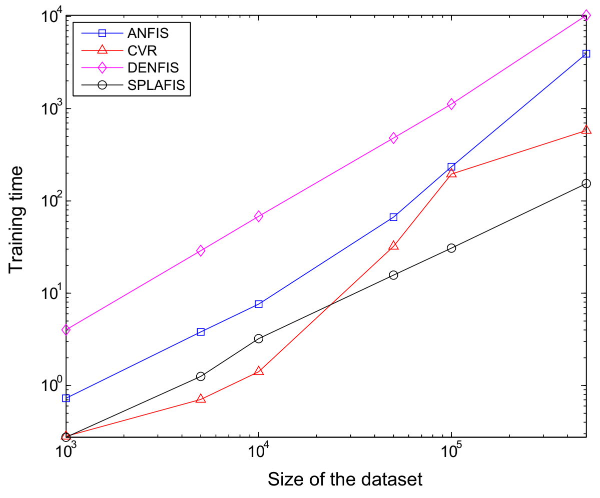
Another study was conducted to see what happens to the generated rule base if the environment returns to its original condition. To this end, additional 60 years of reversed-order S&P index values were appended to the original index series. The same user parameters of the SPLAFIS are used. Figs. 8 and 9 summarize the results. As seen, SPLAFIS continues to follow through the reversal in the trajectory shifts for the following years (till year 2068). Interestingly, Fig. 8(a) shows that the rule trace for the original series is not exactly a mirror-image of that for the reversed series. This can be attributed to the fact that the system already has some existing knowledge by the time the reversed series comes in. Nevertheless, rule reorganization (i.e., pruning) still gradually occurs upon the presentation of the reversed series. Such result demonstrates the system's ability to adapt to the changing environment without radically overwriting the knowledge previously gained. This can be contrasted to the result of SeroFAM (Tan &



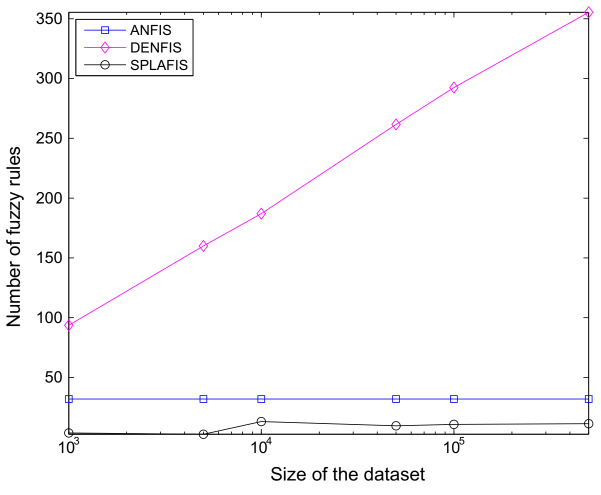
(a) Friedman's data (1000 points)



(b) Test NDEI



(c) Training time



(d) No. of rules

Fig. 10. Experimental results on Friedman's data.

Author's personal copy

Quek, 2010), which revamps its rule base quite frequently, and is more susceptible to catastrophic forgetting and unstable learning.

5.3. Scalability test: large nonlinear regression

The last empirical study reported in this paper involves a large function approximation problem based on Friedman's data generator (Friedman, 1991). The objective is to test the scaling properties of the SPLAFIS model in the face of data having large number of points. In this study, the input features $[x_1, x_2, x_3, x_4, x_5]$ were generated independently, each of which is uniformly distributed within the range of [0, 1]. The target output, on the other hand, is given by (41):

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \sigma(0,1) \qquad (41)$$

where $\sigma(0,1)$ is the noise term normally distributed with mean and variance of 0 and 1, respectively. For performance and scalability evaluations, we generated datasets of varying sizes from $1,000$ to $500,000$ data points. Fig. 10(a) shows a projection of the Friedman's dataset comprising $1,000$ points.

We conducted a 5-fold CV procedure and evaluate the results based on three metrics: test NDEI, training time, and number of rules (all averaged across 5 folds). We benchmark our SPLAFIS model against two scalable NFS methods: ANFIS (Jang, 1993) and DENFIS (Kasabov & Song, 2002) (Other NFS methods are not scalable enough for the task, or their implementations are not available). We also compare our result with that of core vector regression (CVR) (Tsang, Kwok, & Zurada, 2006), a fast solver for support vector regression. For this experiment, the SPLAFIS pruning threshold was fixed as $\beta = 0.01$, and the CVR's approximation parameter set as $\epsilon = 10^{-5}$ (as recommended in Tsang et al., 2006). For ANFIS, its number of rules was set as $M = 2^5 = 32$ so as to keep the rule base size moderate. All other parameters of the four systems are determined based on a validation set containing 20% points (randomly) taken from every dataset.

Fig. 10(b)–(d) present the consolidated benchmark results for varying data sizes, in terms of test NDEI, training time, and rule base size, respectively. Fig. 10(b) shows that SPLAFIS performs quite well in terms of test NDEI, although it does not outperform ANFIS. We attribute the latter to the offline multi-pass learning in ANFIS, as opposed to the one-pass online learning in SPLAFIS. Nevertheless, SPLAFIS exhibits much faster training time and smaller rule base than ANFIS as well as all other methods, as evident from Fig. 10(c)–(d). We also note that DENFIS initially has low NDEI for small data sizes ($\leqslant 10,000$), but its performance deteriorates rapidly for larger number of data points, along with substantial increase of its rule base size and training time. These can be largely attributed to the absence of pruning procedure in the DENFIS method. It is also shown that SPLAFIS outperforms CVR in terms of test NDEI and speed when the data size gets larger. All in all, these results suggest that the proposed SPLAFIS model is scalable and, at the same time, able to balance between predictive accuracy and rule base size.

6. Conclusion

In this paper, we present a new online probabilistic learning method for fuzzy inference systems termed the SPLAFIS model. A unique contribution of the proposed model is that it is the first kind of fuzzy inference system that derives both its structural and parameter learning mechanisms from the statistically-sound principle of maximum a posterior (MAP) based on the Bayes' rule. For its structural learning, SPLAFIS employs a fast online BART algorithm that can automatically evolve the structure of fuzzy rule base from scratch in accordance to the MAP principle. Parameter learning is then carried out by the RDEKF algorithm, which also stems from MAP and offers an efficient localized sequential optimization method. Finally, SPLAFIS also incorporates a simple method to prune inconsequential rules without compromising the predictive accuracy. Experiments on chaotic time series, stock index, and large nonlinear regression datasets have exemplified the efficacy of the SPLAFIS approach. The results show that SPLAFIS is not only efficient, but also exhibits competitive predictive accuracy and compact rule base structure in both static and dynamic tasks.

The proposed SPLAFIS methodology brings about several broader impacts on the advances in expert systems as well as their applications. First, it presents a new type of online learning approach that bridges the gap between two major camps in expert system research: fuzzy rule-based system and probabilistic learning. That is, the development of SPLAFIS offers insights to the possibility of marrying the two camps in a synergistic way, giving birth to an effective, statistically-sound online learning framework. Second, as our experimental results have demonstrated, our SPLAFIS approach possesses the efficiency and scalability traits necessary to cope with the ever-increasing amount of data generated by many real-world applications today. This would pave a way to a new generation of expert system technologies that can gracefully analyze continuous data streams for knowledge extraction, adapt system management rapidly based on changes of the environment, and make numerous real-time decisions about priorities of things to examine and/or execute.

Although the proposed SPLAFIS framework offers a powerful online learning methodology, there remain several rooms for improvement. For instance, the current SPLAFIS learning algorithm does not explicitly take into account the interpretability of the rule base constructed. Several interesting directions for improving rule interpretability in online NFS have been proposed in Oentaryo, Pasqier, and Quek (2011), Angelov (2012) and Lughofer (2013), but these methods are still heuristic in general, and we would like to develop a more theoretically-sound probability-based technique for rule reduction and interpretability enhancement. Also, the current Kalman filter-based parameter learning of the SPLAFIS model implicitly assumes that the rule base parameters follow a normal (Gaussian) distribution, which may not necessarily hold for all tasks. A plausible generalization to the RDEKF method in SPLAFIS is the particle filtering algorithm (Doucet & Johansen, 2011). Particle filter uses a set of particles to estimate the posterior density in an online manner, and makes no restrictive assumption about the shape of the density function (i.e., it is applicable to non-Gaussian parameters). Last but not least, the current SPLAFIS learning algorithm is not designed to handle high-dimensional data with large (e.g., thousands) input features. Our future endeavor would consist of embedding a new probabilistic online feature selection algorithm into the SPLAFIS model.

Acknowledgment

This research is supported by the A∗STAR Science and Engineering Research Council (SERC), Singapore-Poland Programme. We would like to thank the Singapore Institute of Manufacturing Technology (SIMTech) for providing the computational resources. We also thank all reviewers for their constructive feedbacks that resulted in improvements of this paper.

Appendix A. Derivation of RDEKF Adjustment

This Appendix describes the mathematical derivation of the formulae in (27)–(29), associated with the RDEKF procedure in Section 4.2. First, based on (6), the gradient with respect to the consequent weights $w_{i,k_w,m}$ can be computed via (42):

Author's personal copy

$$\frac{\partial y_m}{\partial w_{i,k_w,m}} = \frac{\partial \sum_{k=1}^{K} \lambda_{R_k} \left(\sum_{i=0}^{I} w_{i,k,m} x_i \right)}{\partial w_{i,k_w,m}} = \frac{\partial \sum_{k=1}^{K} \sum_{i=0}^{I} \lambda_{R_k} w_{i,k,m} x_i}{\partial w_{i,k_w,m}} = \lambda_{R_{k_w}} x_i \tag{42}$$

On the other hand, the derivative for the center of the (Gaussian) membership function can be computed using the standard *chain rule*. This is given by (43), based on the inference procedure defined in (4)–(6):

$$\frac{\partial y_m}{\partial c_{i,k_w}} = \frac{\partial \sum_{k=1}^{K} \lambda_{R_k} C_{k,m}}{\partial c_{i,k_w}} = \frac{\partial \frac{\sum_{k=1}^{K} \mu_{R_k} C_{k,m}}{\sum_{k=1}^{K} \mu_{R_k}}}{\partial c_{i,k_w}} = \frac{\left[\left(\sum_{k=1}^{K} \mu_{R_k} \right) C_{k_w,m} - \left(\sum_{k=1}^{K} \mu_{R_k} C_{k,m} \right) \right] \frac{\partial \mu_{R_k}}{\partial c_{i,k_w}}}{\left(\sum_{k=1}^{K} \mu_{R_k} \right)^2} = \left[\frac{C_{k_w,m} - y_m}{\sum_{k=1}^{K} \mu_{R_k}} \right] \frac{\partial \mu_{R_{k_w}}}{\partial c_{i,k_w}} \tag{43}$$

where $\frac{\partial \mu_{R_{k_w}}}{\partial c_{i,k_w}}$ can be resolved as in (44):

$$\frac{\partial \mu_{R_{k_w}}}{\partial c_{i,k_w}} = \frac{\partial \exp\left(-\frac{1}{2} \sum_{i'=1}^{I} \frac{(x_{i'} - c_{i',k_w})^2}{\sigma_{i',k_w}^2} \right)}{\partial c_{i,k_w}} = \left[\prod_{i'=1, i' \neq i}^{I} \exp\left(-\frac{1}{2} \frac{(x_{i'} - c_{i',k_w})^2}{\sigma_{i',k_w}^2} \right) \right] \times \exp\left(-\frac{1}{2} \frac{(x_i - c_{i,k_w})^2}{\sigma_{i,k_w}^2} \right) \left(\frac{x_i - c_{i,k_w}}{\sigma_{i,k_w}^2} \right) = \mu_{R_{k_w}} \left(\frac{x_i - c_{i,k_w}}{\sigma_{i,k_w}^2} \right) \tag{44}$$

Substituting (44) into (43), the formula (45) is finally obtained:

$$\frac{\partial \mu_{R_{k_w}}}{\partial c_{i,k_w}} = \frac{\mu_{R_{k_w}}}{\sum_{k=1}^{K} \mu_{R_k}} (C_{k_w,m} - y_m) \left(\frac{x_i - c_{i,k_w}}{\sigma_{i,k_w}^2} \right) = \lambda_{R_{k_w}} (C_{k_w,m} - y_m) \left(\frac{x_i - c_{i,k_w}}{\sigma_{i,k_w}^2} \right) \tag{45}$$

In a similar manner to (43), the gradient of the Gaussian width can be resolved as in (46):

$$\frac{\partial y_m}{\partial \sigma_{i,k_w}} = \frac{\partial \sum_{k=1}^{K} \lambda_{R_k} C_{k,m}}{\partial \sigma_{i,k_w}} = \left[\frac{C_{k_w,m} - y_m}{\sum_{k=1}^{K} \mu_{R_k}} \right] \frac{\partial \mu_{R_{k_w}}}{\partial \sigma_{i,k_w}} \tag{46}$$

The term $\frac{\partial \mu_{R_{k_w}}}{\partial \sigma_{i,k_w}}$ can subsequently be resolved as per (47):

$$\frac{\partial \mu_{R_{k_w}}}{\partial \sigma_{i,k_w}} = \left[\prod_{i'=1, i' \neq i}^{I} \exp\left(-\frac{1}{2} \frac{(x_{i'} - c_{i',k_w})^2}{\sigma_{i',k_w}^2} \right) \right] \times \frac{\partial \exp\left(-\frac{1}{2} \frac{(x_i - c_{i,k_w})^2}{\sigma_{i,k_w}^2} \right)}{\partial \sigma_{i,k_w}} = \left[\prod_{i'=1, i' \neq i}^{I} \exp\left(-\frac{1}{2} \frac{(x_{i'} - c_{i',k_w})^2}{\sigma_{i',k_w}^2} \right) \right] \times \exp\left(-\frac{1}{2} \frac{(x_i - c_{i,k_w})^2}{\sigma_{i,k_w}^2} \right) \left(\frac{(x_i - c_{i,k_w})^2 4\sigma_{k_w}}{(2\sigma_{i,k_w}^2)^2} \right) = \mu_{R_{k_w}} \left(\frac{(x_i - c_{i,k_w})^2}{\sigma_{i,k_w}^3} \right) \tag{47}$$

Substituting (47), the overall gradient can be written as (48):

$$\frac{\partial \mu_{R_{k_w}}}{\partial \sigma_{i,k_w}} = \frac{\mu_{R_{k_w}}}{\sum_{k=1}^{K} \mu_{R_k}} (C_{k_w,m} - y_m) \left(\frac{(x_i - c_{i,k_w})^2}{\sigma_{i,k_w}^3} \right) = \lambda_{R_{k_w}} (C_{k_w,m} - y_m) \left(\frac{(x_i - c_{i,k_w})^2}{\sigma_{i,k_w}^3} \right) \tag{48}$$

The Eqs. (42), (45) and (48) complete the proofs for the gradient formulae given in (30)–(32), respectively.

Appendix B. Update of covariance matrix

Based on (30)–(32), it can be seen that the derivatives share a common factor $\lambda_{R_{k_w}}$. In turn, we can extract $\lambda_{R_{k_w}}$ out and write the Jacobian matrix $\mathbf{H}_{k_w}(n)$ given by (26) as in (49):

$$\mathbf{H}_{k_w}(n) = \lambda_{R_{k_w}} \mathbf{X}_{k_w(n)} \tag{49}$$

where $\mathbf{X}_{k_w}$ contains the residual elements (i.e., $\mathbf{H}_{k_w}(n)$ divided by $\lambda_{R_{k_w}}$). Meanwhile, by substituting (20), the update formula (21) for the RDEKF covariance matrix (assuming a multi-input–single-output system and $\mathbf{R} = 1$) can be expressed as (50):

$$\mathbf{P}_{k_w}(n) = \left[\mathbf{I}_{Z \times Z} - \mathbf{G}_{k_w}(n) \mathbf{H}_{k_w}^T(n) \right] \mathbf{P}_{k_w}(t-1) = \mathbf{P}_{k_w}(t-1) - \mathbf{G}_{k_w}(n) \mathbf{H}_{k_w}^T(n) \mathbf{P}_{k_w}(t-1) = \mathbf{P}_{k_w}(t-1) - \frac{\mathbf{P}_{k_w}(t-1) \mathbf{H}_{k_w}(n) \mathbf{H}_{k_w}^T(n) \mathbf{P}_{k_w}(t-1)}{1 + \mathbf{H}_{k_w}^T(n) \mathbf{P}_{k_w}(t-1) \mathbf{H}_{k_w}(n)} \tag{50}$$

Accordingly, substituting (49) into (50) results in (51):

$$\mathbf{P}_{k_w}(n) = \mathbf{P}_{k_w}(t-1) \tag{51}$$

$$-\frac{\lambda_{R_{k_w}}^2 \mathbf{P}_{k_w}(t-1) \mathbf{X}_{k_w}(n) \mathbf{X}_{k_w}^T(n) \mathbf{P}_{k_w}(t-1)}{1 + \lambda_{R_{k_w}}^2 \mathbf{X}_{k_w}^T(n) \mathbf{P}_{k_w}(t-1) \mathbf{X}_{k_w}(n)} \tag{52}$$

Then, using the definition in (5), (51) translates to (53):

$$\mathbf{P}_{k_w}(n) = \mathbf{P}_{k_w}(t-1) - \frac{\mu_{R_{k_w}}^2 \mathbf{P}_{k_w}(t-1) \mathbf{X}_{k_w}(n) \mathbf{X}_{k_w}^T(n) \mathbf{P}_{k_w}(t-1)}{\left(\sum_{l=1}^{K} \mu_{R_l} \right)^2 + \mu_{R_{k_w}}^2 \mathbf{X}_{k_w}^T(n) \mathbf{P}_{k_w}(t-1) \mathbf{X}_{k_w}(n)} \tag{53}$$

or we can express the history until time t in an explicit way using (54):

$$\mathbf{P}_{k_w}(n) = \mathbf{I}_{Z \times Z} - \sum_{u=1}^{t} \frac{A_u}{F + B_u} \tag{54}$$

where $A_u = \mu_{R_{k_w}}^2 \mathbf{P}_{k_w}(u-1) \mathbf{X}_{k_w}(u) \mathbf{X}_{k_w}^T(u) \mathbf{P}_{k_w}(u-1)$, $B_u = \mu_{R_{k_w}}^2 \mathbf{X}_{k_w}^T(u) \mathbf{P}_{k_w}(u-1) \mathbf{X}_{k_w}(u)$, and $F = \left(\sum_{l=1}^{K} \mu_{R_l} \right)^2$.

Now supposing a rule added at time t has been added from the beginning, the covariance matrix at time t should be in the form of (55):

$$\widetilde{\mathbf{P}}_{k_w}(n) = \mathbf{I}_{Z \times Z} - \sum_{u=1}^{t} \frac{A_u}{\left(\sum_{l=1}^{K} \mu_{R_l} + \mu_{R_{K+1}} \right)^2 + B_u} = \mathbf{I}_{Z \times Z} - \sum_{u=1}^{t} \frac{A_u}{F + \delta F_1 + \delta F_2 + B_u} \tag{55}$$

where $\delta F_1 = \mu_{R_{K+1}}^2$ and $\delta F_2 = 2\mu_{R_{K+1}} \sum_{l=1}^{K} \mu_{R_l}$. In sum, adding a rule at time t leads to a corruption of the covariance matrix, i.e., increase of the denominator of the part subtracted from $\mathbf{P}_0 = \mathbf{I}_{Z \times Z}$. Note that $\delta F_1 \ll 1$ and $\delta F_2 \ll 1$ in practice, since both are quadratic forms of fuzzy membership values. By contrast, B_u may be large because it is a quadratic form of the input data multiplied by the covariance matrix. Also note that $F > F_1$ as it is a sum of K positive membership

Author's personal copy

values, whereas F_1 stems simply from a single membership value. Moreover, $F > F_2$ holds if $\mu_{R_{K+1}} > \frac{1}{2}\sum_{l=1}^{K}\mu_{R_l}$. As such, the influence of the addends will be more significant only when *all* values of the input x_i for *all* time steps tend to zero or the covariance matrix tends to zero. Practical tests using several functions shows the corruption of the covariance matrix due to the addition of a new rule is marginal.

In SPLAFIS, this small influence is estimated as follows: Based on (54) and (55), the corrupted covariance matrix $\widetilde{\mathbf{P}}_{k_w}(n)$ is a function of the original matrix $\mathbf{P}_{k_w}(n)$, as per (56):

$$\widetilde{\mathbf{P}}_{k_w}(n) = f(\mathbf{P}_{k_w}(n)) \tag{56}$$

A reasonable approximation of the $f(.)$ would be the inverse squared mean given in (57), as the effect of the corruption will decrease when the number of rules K increases. This is essentially a squared dependency given by (57):

$$\widetilde{\mathbf{P}}_{k_w}(n) = \left(\frac{K^2+1}{K^2}\right)\mathbf{P}_{k_w}(n) \tag{57}$$

which corresponds to the reset mechanism described in (35).

References

Angelov, P. P. (2010). Evolving Takagi–Sugeno fuzzy systems from streaming data, eTS+. In P. Angelov, D. Filev, & N. Kasabov (Eds.), *Evolving intelligent systems: methodology and applications* (pp. 21–50). John Wiley & Sons.

Angelov, P. (2012). *Autonomous learning systems: From data to knowledge in real time*. John Willey and Sons.

Angelov, P., & Filev, D. (2005). Simpl_eTS: A simplified method for learning evolving Takagi–Sugeno fuzzy models. In *Proceedings of the IEEE international conference on fuzzy systems* (pp. 1068–1073).

Angelov, P., & Zhou, X. W. (2006). Evolving fuzzy systems from data streams in real-time. In *Proceedings of the IEEE international symposium on evolving fuzzy systems* (pp. 29–35).

Angelov, P., & Filev, D. (2004). An approach to online identification of Takagi–Sugeno fuzzy models. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, 34*(1), 484–498.

Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J., & Rosen, D. (1992). Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks, 3*(5), 698–713.

Cheu, E. Y., Quek, C., & Ng, S. K. (2012). ARPOP: An appetitive reward-based pseudo-outer-product neural fuzzy inference system inspired from the operant conditioning of feeding behavior in aplysia. *IEEE Transactions on Neural Networks and Learning Systems, 23*(2), 317–329.

Doucet, A., & Johansen, A. M. (2011). A tutorial on particle filtering and smoothing. In D. Crisan & B. Rozovsky (Eds.), *The oxford handbook of nonlinear filtering*. Oxford University Press.

Driankov, D., Hellendoorn, H., & Reinfrank, M. (1996). *An introduction to fuzzy control*. Heidelberg, Germany: Springer Verlag.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification* (2nd ed.). New York: Wiley.

Friedman, J. (1991). Multivariate adaptive regression splines. *Annals of Statistics, 19*(1), 1–141.

Gama, J. (2010). *Knowledge discovery from data streams*. Boca Raton, Florida: Chapman & Hall/CRC.

Grossberg, S. (1976). Adaptive pattern recognition and universal encoding II: Feedback, expectation, olfaction, and illusions. *Biological Cybernetics, 23*, 187–202.

Haykin, S. (2009). *Neural networks and learning machines* (3rd ed.). Prentice Hall.

Ho, Y., & Lee, R. (1964). A bayesian approach to problems in stochastic estimation and control. *IEEE Transactions on Automatic Control, 9*(4), 333–339.

Huang, G. B., Saratchandran, P., & Sundararajan, N. (2005). A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. *IEEE Transactions on Neural Networks, 16*(1), 57–67.

Jang, J.-S. R. (1993). ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man and Cybernetics, 23*(3), 665–685.

Jazwinski, A. H. (1970). *Stochastic processes and filtering theory*. New York: Academic Press.

Juang, C. F., & Lin, C. T. (1998). An on-line self-constructing neural fuzzy inference network and its applications. *IEEE Transactions on Fuzzy Systems, 6*(1), 12–32.

Kadirkamanathan, V., & Niranjan, M. (1993). A function estimation approach to sequential learning with neural networks. *Neural Computation, 5*(6), 954–975.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME, Journal of Basic Engineering, 82*(Series D), 35–45.

Kasabov, N. (2001). Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning. *IEEE Transactions on Systems, Man and Cybernetics, Part B, 31*(6), 902–918.

Kasabov, N. K., & Song, Q. (2002). DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *IEEE Transactions on Fuzzy Systems, 10*(2), 144–154.

Kim, J., & Kasabov, N. (1999). HyFIS: Adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems. *Neural Networks, 12*(9), 1301–1319.

Leng, G., McGinnity, T. M., & Prasad, G. (2005). An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network. *Fuzzy Sets and Systems, 150*(2), 211–243.

Lin, Y. Y., Chang, J. Y., & Lin, C. T. (2014). A TSK-type-based self-evolving compensatory interval type-2 fuzzy neural network (TSCIT2FNN) and its applications. *IEEE Transactions on Industrial Electronics, 61*, 447–459.

Lin, Y. Y., Chang, J. Y., Pal, N. R., & Lin, C. T. (2013). A mutually recurrent interval type-2 neural fuzzy system (MRIT2NFS) with self-evolving structure and parameters. *IEEE Transactions on Fuzzy Systems, 21*(3), 492–509.

Lin, C. T., & Lee, C. S. G. (1996). *Neural fuzzy systems: A neuro-fuzzy synergism to intelligent systems*. Upper Saddle River, NJ: Prentice Hall.

Lin, F. J., Lin, C. H., & Shen, P. H. (2001). Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive. *IEEE Transactions on Fuzzy Systems, 9*, 751–759.

Lughofer, E. (2008). FLEXFIS: A robust incremental learning approach for evolving Takagi–Sugeno fuzzy models. *IEEE Transactions on Fuzzy Systems, 16*(6), 1393–1410.

Lughofer, E. (2011). *Evolving fuzzy systems: Methodologies. Advanced concepts and applications*. Berlin Heidelberg: Springer Verlag.

Lughofer, E. (2013). On-line assurance of interpretability criteria in evolving fuzzy systems: Achievements, new concepts and open issues. *Information Sciences, 251*, 22–46.

Lughofer, E., & Buchtala, O. (2013). Reliable all-pairs evolving fuzzy classifiers. *IEEE Transactions on Fuzzy Systems, 21*(4), 625–641.

Mackey, M. C., & Glass, L. (1977). Oscillation and chaos in physiological control systems. *Science, 197*(4300), 287–289.

Oentaryo, R. J., Pasqier, M., & Quek, C. (2011). RFCMAC: A novel reduced localized neuro-fuzzy system approach to knowledge extraction. *Expert Systems with Applications, 38*(10), 12066–12084.

Platt, J. (1991). A resource-allocating network for function interpolation. *Neural Computation, 3*(2), 213–225.

Pratama, M., Anavatti, S. G., Angelov, P. P., & Lughofer, E. (2014). PANFIS: A novel incremental learning machine. *IEEE Transactions on Neural Networks and Learning Systems, 25*(1), 55–68.

Pratama, M., Er, M. J., Li, X., Oentaryo, R. J., Lughofer, E., & Arifin, I. (2013). Data-driven modeling based on dynamic parsimonious fuzzy neural network. *Neurocomputing, 110*, 18–28.

Puskorius, G. V., & Feldkamp, L. A. (1991). Decoupled extended Kalman filter training of feedforward layered networks. In *Proceedings of the international joint conference on neural networks* (Vol. 1, pp. 771–777), Seattle.

Rong, H., Sundararajan, N., Huang, G., & Saratchandran, P. (2006). Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction. *Fuzzy Sets and Systems, 157*(9), 1260–1275.

Ruan, D., & Benítez-Read, J. S. (2005). Fuzzy control for nuclear reactor operation: Strengths, weaknesses, opportunities and threats. *Journal of Intelligent and Fuzzy Systems, 16*(4), 289–295.

Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics, 15*(1), 116–132.

Tan, J., & Quek, C. (2010). A BCM theory of meta-plasticity for online self-reorganizing fuzzy-associative learning. *IEEE Transactions on Neural Networks, 21*(6), 985–1003.

Tsang, I. W. H., Kwok, J. T. Y., & Zurada, J. M. (2006). Generalized core vector machines. *IEEE Transactions on Neural Networks, 17*(5), 1126–1140.

Tung, S. W., Quek, C., & Guan, C. (2011). SaFIN: A self-adaptive fuzzy inference network. *IEEE Transactions on Neural Networks, 22*(12), 1928–1940.

Vigdor, B., & Lerner, B. (2007). The Bayesian ARTMAP. *IEEE Transactions on Neural Networks, 18*(6), 1628–1644.

Wahab, A., Quek, C., Tan, C. K., & Takeda, K. (2009). Driving profile modeling and recognition based on soft computing approach. *IEEE Transactions on Neural Networks, 20*(4), 563–582.

Wang, N., Er, M. J., & Meng, X.-Y. (2009). A fast and accurate online self-organizing scheme for parsimonious fuzzy neural networks. *Neurocomputing, 72*(16–18), 3818–3829.

Wu, S., & Er, M. J. (2000). Dynamic fuzzy neural networks: A novel approach to function approximation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, 30*(2), 358–364.

Wu, S., Er, M. J., & Gao, Y. (2001). A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks. *IEEE Transactions on Fuzzy Systems, 9*(4), 578–594.

Yajun, Z., Tianyou, C., Hong, W., Jun, F., Liyan, Z., & Yonggang, W. (2010). An adaptive generalized predictive control method for nonlinear systems based on ANFIS and multiple models. *IEEE Transactions on Fuzzy Systems, 18*(6), 1070–1082.

Zadeh (1994). Soft computing and fuzzy logic. *IEEE Software, 11*(6), 48–56.

View publication stats