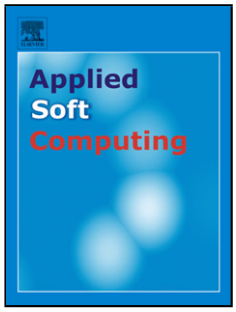
Learning evolving T-S fuzzy systems with both local and global accuracy – a local online optimization approach

DOI:
10.1016/j.asoc.2017.05.046

Document Version
Accepted author manuscript



Title: Learning evolving T-S fuzzy systems with both local and global accuracy – a local online optimization approach

Author: Dongjiao Ge Xiao-Jun Zeng

PII: S1568-4946(17)30311-3
DOI: http://dx.doi.org/doi:10.1016/j.asoc.2017.05.046
Reference: ASOC 4250

To appear in: *Applied Soft Computing*

Received date: 31-1-2017
Revised date: 2-5-2017
Accepted date: 23-5-2017

Please cite this article as: Dongjiao Ge, Xiao-Jun Zeng, Learning evolving T-S fuzzy systems with both local and global accuracy ndash a local online optimization approach, *<![CDATA[Applied Soft Computing Journal]]>* (2017), http://dx.doi.org/10.1016/j.asoc.2017.05.046

Learning evolving T-S fuzzy systems with both local and global accuracy – a local online optimization approach

Dongjiao Ge, Xiao-Jun Zeng*

School of Computer Science, University of Manchester, Manchester M13 9PL, UK

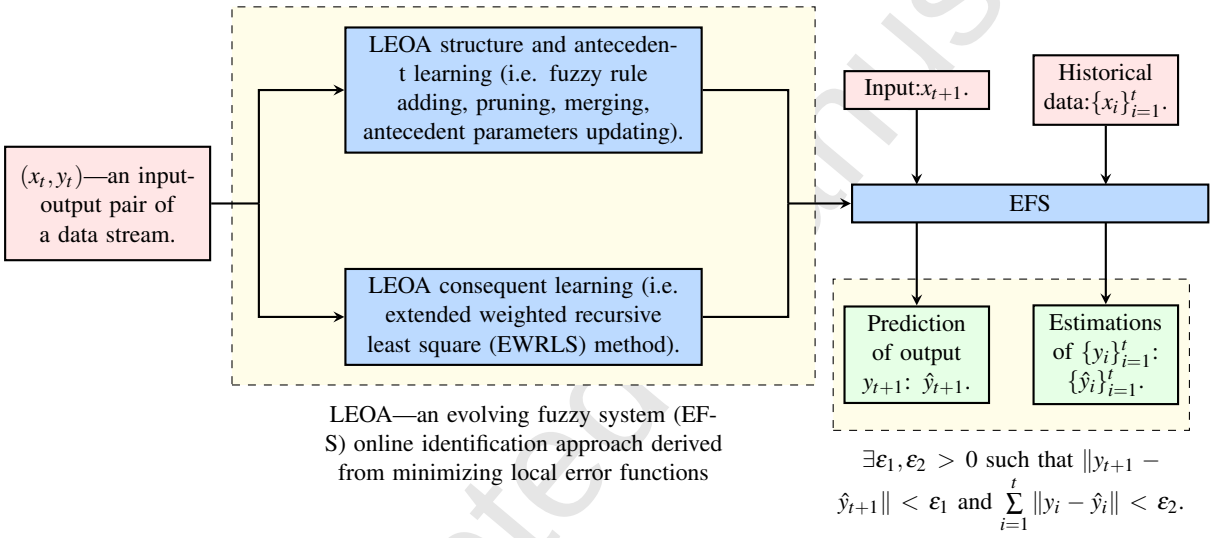
Abstract

Most real data streams are non-linear and non-stationary by nature, which makes it a challenging issue to develop effective learning techniques. With the advantages of updating the system structure and parameters on the fly, evolving fuzzy systems (EFSs) are effective paradigms to address this issue. However, existing methods and algorithms of EFSs are usually: (1) developed based on a heuristic rather than an optimal approach and put main focus on tracking the most recent local model, thus leading to an "unlearning effect" and often poor global accuracy; (2) lack of optimality of the consequent parameters when there is a structure update of the fuzzy system. In order to resolve these issues, this paper proposes a local error optimization approach (LEOA) for identifying evolving T-S fuzzy systems. LEOA has its antecedent learning method derived from minimizing a bunch of local error functions and guarantee the optimality of the consequent parameters by a new extended weighted recursive least square (EWRLS) method. Furthermore, mathematical proofs and calculations are provided to verify the optimality and ε-completeness property of LEOA. Numerical examples based on several benchmark and real-world data sets are tested, and the results demonstrate that LEOA not only makes preferable local prediction accuracy compared with existing state-of-the-art methods but also reserves the global accuracy of the identified models.

Keywords: evolving fuzzy system, weighted recursive least square, local error

*Corresponding Author: Xiao-Jun Zeng

Email addresses: dongjiao.ge@manchester.ac.uk (Dongjiao Ge), x.zeng@manchester.ac.uk (Xiao-Jun Zeng)



LEOA—an evolving fuzzy system (EFS) online identification approach derived from minimizing local error functions

$\exists \varepsilon_1, \varepsilon_2 > 0$ such that $\|y_{t+1} - \hat{y}_{t+1}\| < \varepsilon_1$ and $\sum_{i=1}^{t} \|y_i - \hat{y}_i\| < \varepsilon_2$.

function

1. Introduction

The analysis and prediction of data streams is required in many real-world application areas such as finance [1], [2], energy [3], [4], medicine [5], [6]. As well summarized in [7], a data stream usually has four major characteristics: (1) data comes as unlimited streams that continuously flow with high-speed; (2) the underlying distributions may be non-stationary and evolving over time; (3) data should never being regarded to be independent and identically distributed; (4) data is often time situated and specially. Due to these reasons, data stream learning methods should have the ability to change their parameters or structures to capture the change of the data stream, and give better performance in dynamic and evolving environments. As well highlighted in [8], evolving models are one of the main branches that can effectively deal with these issues, because their parameters and structures can be adapted over time following the changes in the given learning environments. Evolving fuzzy systems (eFSs) are one of the most promising evolving models. Most eFSs work as one-pass learning methods to process data on the fly, learn from data incrementally to extract useful and new knowledge from data, and evolve the system structure to track behavior and structure changes of systems being learned. With these merits, the research in eFSs has attracted a lot of attentions in the recent years and some important progress have been achieved in both online prediction [9–14] and online clustering [15–17]. Generally speaking, the two main and common issues that most eFSs related researches put effort to solve are: (1) how structure being evolved; (2) how parameters being updated.

In order to better address the above two issues and improve the existing approaches in eFSs, in this paper, we propose a local online optimization approach for identifying evolving Takagi-Sugeno systems, known as local error optimization approach (LEOA). LEOA contains the strategies for determining the number of fuzzy rules (i.e. fuzzy rule adding, pruning and merging), and the updating methods for both antecedent and consequent parameters (i.e. cluster centers, radiuses, and coefficients of linear consequent part) from the data stream. From an optimization point of view, the ultimate purpose

for this paper is developing an algorithm which can minimize a series of local error functions considering any changes brought by the change of parameters and structure of a fuzzy system. Consequently, we derive four modules of LEOA. (1) Rule base is expended relying on the activation degree of new data in every existing clusters; (2) rule merging process is triggered when two clusters have highly similar centers and radiuses with similarity degree judged by an activation degree based similarity measure; (3) redundant fuzzy rules are removed according to the cumulative activation degree and age; (4) consequent parameters are updated through an extended weighted recursive least square (EWRLS) algorithm. Furthermore, the optimality of each of these modules is verified by mathematical calculations and proofs.

The rest parts of this paper are organized as follows. Extensive literature review, research gaps between LEOA and existing works as well as novelty of LEOA are presented in section 2. Section 3 gives a brief description of the T-S fuzzy system which needs to be identified. The detailed description of structure learning strategies (i.e. fuzzy rule adding, pruning and merging strategies), and parameters (i.e. antecedent and consequent parameters) updating techniques includes EWRLS are deliberated in section 4. In section 5, the detailed steps and flowchart of the proposed algorithm are presented. Section 6 and 7 analyze the complexity and sensitivity of LEOA. Numerical results of LEOA across four well known benchmark examples are presented in section 8. In the last section, section 9, the conclusions are given.

2. Related works

Initial works for online algorithms to identify eFSs appear around 2000. For example, evolving fuzzy neural networks (EFuNNs) proposed by [18] are a branch of robust online learning algorithms based on the local learning strategy to learn from the data swiftly. [19] develops a non-iterative approach to identify the evolving fuzzy rule based system, which uses an incremental unsupervised learning method to estimate the parameters and allows the existing fuzzy rules to be replaced by new fuzzy rules based on the ranking of the fuzzy rules from informative potential. [9] proposes a dynamic evolving neural-fuzzy inference system known as DENFIS. It has both off-line

and online modes, in which the online version updates the model structure by evolving cluster method (ECM), and estimates the model consequent parameters by recursive least square (RLS) method. Other pioneer works including [20–22] etc. Based on these early works of eFSs identification methods, [10] proposes an online learning method for evolving T-S system (eTS) by combining supervised and unsupervised learning, which enables the rule base and parameters of T-S fuzzy system to be updated from adding new rules and modifying existing rules. Self-organizing fuzzy neural network (SOFNN) proposed by [23] is another competitive algorithm whose fuzzy rule adding criterion is built based on geometric growing criterion. [24] develops a self-evolving neural network with interval type-2 fuzzy sets in antecedent parts for T-S fuzzy system. This methodology is characterized by its online clustering method, which generates fuzzy rules simply by using the firing strength directly and develops a fuzzy set reduction approach to avoid the high overlapping between fuzzy sets. [11] proposes a robust online adaptive method called SOFMLS to identify Mamdani fuzzy systems by adding new fuzzy rules using a distance based approach and pruning fuzzy rules according to the density of the clusters. Similar researches could also be found in [12, 13, 25–33].

Furthermore, most of the latest researches are mainly focusing on how to design criterions to make unsupervised online clustering process of data streams using the inputs data, or noticing to achieve the prediction accuracy from two major aspects: online feature selection and consequent updating. For instance, in ePL+ [34], a subtractive clustering method is applied to complement the participatory learning approach in clustering the inputs data. A gradual forgetting approach and the Hebbian learning mechanism are applied in improving the accuracy for online clustering process being proposed by GSETSK [14]. Statistical contribution depended fuzzy rule online pruning method and online clustering method are proposed by DPFNN[35] and PANFIS [36], respectively. As an extended work of PANFIS, GENEFIS [37] applies the generalized recursive least square (GRLS) [38] algorithm in local learning of the consequent parameters, and achieves accurate prediction results with an economical rule base. Instead of investigating rule generation method, Gen-Smart-EFS [39] puts effort on studying online feature weighting strategies to improve eFSs in transparency and computation load. By the aim of improving the prediction accuracy, [40] consid-

ers non-linearity degree and uncertainties in both model outputs and parameters, and puts forward a sample selection method for data stream regression problems to make decisions on whether a sample can be used in structure and parameters updating. Besides, recent online clustering methods and eFSs identification methods are also being studied on type-2 fuzzy system [41, 42]. More similar eFSs based researches can be found from [39, 43–48]. Furthermore, comprehensive and excellent overviews and comparison about eFS methods can be found in [49], [50].

Although these previous works are known as satisfactory online learning algorithms for the identification of evolving fuzzy systems, they still have two major technical flaws shown as follows:

1) Summarizing these previous state-of-the-art eFSs identification algorithms, it can be seen that one of the most widely used consequent learning method is a local optimum method known as WRLS (e.g. [14, 25, 34, 39]) rather than the global version (i.e. recursive least square (RLS)). The reason behind this is that WRLS minimizes a bunch of local error functions, hence enables more flexible structure evolving. It can be seen from these existing approaches, many different kinds of generalized version of WRLS are plugged in learning consequent parameters. [36] uses an extended recursive least square (ERLS) approach to minimize local weighted error functions. This ERLS method is proved to have small finite system error. In [40], WRLS is used to update the consequent parameters, but the samples that used in this updating procedure is selected to avoid under fitting in highly noisy region through taking the uncertainty in model outputs into consideration. Besides, [37] applies a fuzzily weighted generalized recursive least square (FWGRLS) method that extends the generalized recursive least square (GRLS) [38] to a local learning version and adds a weight decay term in the error function. This decay term conveys the prior distribution of the local model, so as to make sure the convergency and enhance the generalization ability. Furthermore, this FWRLS is also generalized to learn the consequent parameters of type-2 fuzzy systems in [42]. However, using both RLS and WRLS directly to learn the consequent parameters are approximation ways, because there is a

need to assume the rule base is unchanged for ensuring they are accurate method [10]. If the structure of the rule base is unchanged (i.e. there is no rule adding, deleting or merging) or the antecedent parameters are unchanged (i.e. the centers and radiuses of membership functions remain the same), then such a parameter updating algorithm is optimal. However, as the structure or antecedent parameters of the rule base are keeping changing during the learning of an eFS, such a direct application of WRLS algorithm is hard to make sure the optimality of consequent parameter updating. Some existing results, such as [12] and [36] firstly addressed the issue that any structural change of the antecedent part will affect the optimality and convergency of the consequent part, and a sub-optimal solution was proposed. However, there is no existing work which gives the explicit calculation of the optimal consequent parameters and assure their optimality.

2) Most existing eFSs and their online learning algorithms focus on adapting the structure and parameters of a fuzzy system by fitting more recent data. These structure evolving strategies are often designed from a heuristic manner similar to online clustering approaches which are usually one-pass and unsupervised (e.g. [51, 52]). This is certainly correct if we are learning an evolving system. Furthermore, this ignorance of the connection between consequent and antecedent learning of most eFSs online identification methods may lead to an "unlearning effect". To be more specific, in many applications, the streaming data arrived in each given period of time are often local data (that is, with a given time period, all data arrived are located at a sub-region of the input space). In such cases, the obtained fuzzy system by these existing learning algorithms end up as a good local model which fits well the latest arrival data accurately but a poor global model which fits the historical data very inaccurate. As the accuracy measurement for the existing eFSs is usually the one-step ahead or testing error, therefore, such a global accuracy issue has been largely ignored in many of the existing approaches. To illustrate this point in more details, a simulation example is given in Section 8.1. An existing method which effectively deals with this "unlearning effect" is proposed by [53], which developed a consequent learning

method using adaptive forgetting factors updated by the recent contribution of a cluster to enable a fuzzy system keeps its global accuracy. In some sense, this is an indirect approach to address this issue by using different adaptive forgetting factors for different rules. Further, [54] considers to develop connections between antecedent and recursive consequent learning by the aim of ensuring optimality of weighted recursive least square (WRLS) method, but the exact calculation and proofs still have not been given. Hence, this ignored influence of rule base evolving strategies to the optimality of recursive consequent learning method needs to be further investigated.

As an algorithm proposed to address these above weaknesses, LEOA has the following novelties.

1. LEOA starts from an optimization point of view. Both of antecedent and consequent learning methods are derived serving for realizing this goal through minimizing a bunch of local error functions. In detail, this paper builds criterions for triggering the structure evolution of the rule base and the corresponding recursive consequent learning method (i.e. an extended recursive least square algorithm using ERLS for short) at the same time, which establishes a natural connection between consequent and antecedent learning. This LEOA approach ensures the optimality of the consequent parameters. Corresponding mathematical calculations and proofs further verify the optimality of LEOA and its ε-completeness [55] property.

2. Although at the first look it may seem very difficult to achieve both the good track of the non-stationary system change and the high global accuracy, such a double goals in fact are realizable by locally optimization learning. The main idea is that, when a new data point arrives, we only update the local model around this data in order to track the system behavior changes around this data point but keep the other parts of the obtained model largely unchanged. The former will ensure the non-stationary behavior being tracked and the latter will maintain the global accuracy. Specifically, these two goals are achieved by LEOA through remembering more historical knowledge of the data stream by certain characters

in the new EWRLS, and the connections being built between antecedent and consequent learning when achieving the same optimality goal. This enables the eFS to keep the useful historical behavior of the data stream while tracking the more recent data. Therefore, LEOA reaches the target to get high local and global accuracy.

3. Problem statement

The T-S fuzzy system considered in this paper is given as follows. Firstly the form of the i-th fuzzy rule is multi-input-single-output (MISO) type given as (1):

$$\begin{aligned} i: \quad & If\ x_1\ is\ \Gamma_1^i\ and\ x_2\ is\ \Gamma_2^i\ and\ \ldots\ and\ x_n\ is\ \Gamma_n^i, \\ & then\ y^i = \psi_0^i + \sum_{j=1}^{n} \psi_j^i x_j, \end{aligned} \tag{1}$$

where $i = 1, 2, \ldots, R$, R is the number of fuzzy rules, x_j is the j-th input, y^i is the output of the i-th fuzzy rule, ψ_j^i, $j = 0, 2, \ldots, n$, are consequent parameters, n is the dimension of the inputs.

In LEOA, Gaussian membership functions are chosen with the center of fuzzy set Γ_j^i is $c_{i,j}$ and the radius is $\sigma_{i,j}$. The membership function of Γ_j^i is $\mu_{i,j}(x_j)$ shown in (2),

$$\mu_{i,j}(x_j) = \exp^{-\frac{(x_j - c_{i,j})^2}{2(\sigma_{i,j})^2}}. \tag{2}$$

Furthermore, the form of the firing strength $\gamma^i(x)$ of each fuzzy rule and the model output $\hat{y}$ is displayed in (3) and (4), respectively.

$$\gamma^i(x) = \prod_{j=1}^{n} \mu_{i,j}(x_j), \tag{3}$$

$$\hat{y} = \sum_{i=1}^{R} \theta^i(x) y^i, \tag{4}$$

in which

$$\theta^i(x) = \gamma^i(x) / \sum_{j=1}^{R} \gamma^j(x) \tag{5}$$

is the normalized firing degree of the i-th fuzzy rule.[1]

The problem needs to solve is identifying the number of fuzzy rules R, antecedent parameters $c_i = (c_{i,1}, c_{i,2}, \ldots, c_{i,n})$, $\sigma_i = (\sigma_{i,1}, \sigma_{i,2}, \ldots, \sigma_{i,n})$ and consequent parameters $\psi^i = (\psi_0^i, \psi_1^i, \psi_2^i, \ldots, \psi_n^i)$, where $i = 1, 2, \ldots, R$, of the T-S fuzzy system in order to make predictions of y. The exact conditions and steps of LEOA learning algorithm for identifying the T-S fuzzy systems are described in section 4 and 5.

4. LEOA learning

As an online learning algorithm, LEOA evolves its structure and parameters while learning from the data stream. When non-stationary phenomenon is detected, than there will be new fuzzy rules needed to be added to react to this change, because the previous identified model is likely lack of ability to make the accurate prediction at the data around a new state. There are probably some rules seldom being activated since they are built, which may cause over-fitting. In order to deal with this problem, these rules should be regarded as redundant ones and be discarded. Since the membership functions in some rules should be expended and their centers should be updated if more data arrives, this could result that some rules become similar. This would decrease the computation speed and lead to the conflict of the fuzzy rules. For avoiding this, a rule merging procedure is necessary to be designed in the algorithm. Therefore, the structure learning should include three parts: fuzzy rule adding, pruning and merging; furthermore, the parameter learning includes how to determine the antecedent parameters, the centers and radiuses of membership functions, and the consequent parameters.

This section mainly discusses and introduces that, under what conditions LEOA can calculate the weighted least square estimation for ψ^i that minimizes (6) using a special type of updating equations (i.e. extended weighted recursive least square, EWRLS, method). Aiming to obtain the conditions for evolving the structure and learning the parameters of the system, we start from the local error function shown in (6) that we

[1] Because each T-S fuzzy system can be presented as a neural network, the LEOA can also be presented as a neural network with the form almost the same as eTS+ [56] and DPFNN [35].

would like to minimize.

$$Err_i = \sum_{k=1}^{t+1} \theta_{t+1}^i(x_k)(y_k - xe_k\psi^i)^2, \tag{6}$$

where $k = 1,2,\ldots,t+1$ is the time, $x_k = (x_{k,1}, x_{k,2}, \ldots, x_{k,n})$ is the input at time k, $xe_k = (1, x_k)$ is the generalized input at time k, $\theta_{t+1}^i(x_k) = \frac{\gamma_{t+1}^i(x_k)}{\sum_{l=1}^{R_{t+1}} \gamma_{t+1}^l(x_k)}$ is the normalized firing degree of the i-th rule at time $t+1$ obtained by replacing x using x_k in (5) with γ_{t+1}^i calculated by using the parameters of the state of eFS at time $t+1$. The EWRLS estimation for ψ^i could be computed using (7) which is the same as WRLS.

$$\psi^i = \{\overline{R^i}(t+1)\}^{-1} f^i(t+1), \tag{7}$$

where $\overline{R^i}(t+1) = \sum_{k=1}^{t+1} \theta_{t+1}^i(x_k) xe_k {xe_k}^T$, $f^i(t+1) = \sum_{k=1}^{t+1} \theta_{t+1}^i(x_k) xe_k y_k$.

If the structure and parameters of the eFS are unchanged, it is very easy to get the EWRLS formula with the same form as WRLS used in most existing researches. However, the antecedent and consequent parameters of eFSs are actually changing, so WRLS can not help to find the optimal solution of (6). In the remaining parts of this section, we utilize the point of view of minimizing (6) in finding the appropriate conditions for structure learning (i.e. fuzzy rule adding, pruning, merging) and parameter learning (i.e. antecedent parameters and consequent parameters). All the following structure and consequent learning *conditions* and *theorems* are obtained under the assumption of that the system could be approximated by finite number of fuzzy rules ($R_k < \infty$, $k = 1,2,\ldots,t$) in any accuracy in finite steps. Additionally, $\{x_k\}_{k=1}^{\infty}$ and $\{y_k\}_{k=1}^{\infty}$ are bounded.

4.1. Structure learning

Because of that the LEOA is suitable for online learning, so rule base of the EFS is initially empty and expended as well as changed while new data comes. This section gives the detailed description of how LEOA evolves its structure.

4.1.1. Fuzzy rule adding

Adding fuzzy rules is expanding the fuzzy rule base to cover the input space. Once data stream is detected to change to a new state, there always a new fuzzy rule needs

to be added. In order to get the exact fuzzy rule adding condition, we assume that the system keeps unchanged until time t (i.e. $c_i(1) = c_i(2) = \ldots = c_i(k)$, $\sigma_i(1) = \sigma_i(2) = \ldots = \sigma_i(k)$, $c_i(k)$ is the cluster center of the i-th fuzzy rule at time k, $\sigma_i(k)$ is the cluster radius of the i-th fuzzy rule at time k, for $i = 1,2,\ldots,R_k$ and $k = 1,2,\ldots,t$, $R_1 = R_2 = \ldots = R_t$), and a new fuzzy rule is added at time $t+1$ with cluster center $c_{R_{t+1}} = x_{t+1}$. Therefore, for $i = 1,2,\ldots,R_t$, the objective function (6) could be transformed to (8),

$$Err_i = \sum_{k=1}^{t} \theta_{t+1}^i(x_k)(y_k - xe_k\psi^i)^2 + \theta_{t+1}^i(x_{t+1})(y_{t+1} - xe_{t+1}\psi^i)^2, \tag{8}$$

where

$$\theta_{t+1}^i(x_k) = \frac{\gamma_{t+1}^i(x_k)}{\sum_{l=1}^{R_{t+1}} \gamma_{t+1}^l(x_k)} = \frac{\gamma_{t+1}^i(x_k)}{\sum_{l=1}^{R_t} \gamma_{t+1}^l(x_k) + \gamma_{t+1}^{R_{t+1}}(x_k)}. \tag{9}$$

Then, $\overline{R^i}(t+1)$, $f^i(t+1)$ and $\psi^i(t+1)$ could be represented by (10), (11) and (14), respectively.

$$\begin{aligned} \overline{R^i}(t+1) &= \sum_{k=1}^{t} \theta_{t+1}^i(x_k) xe_k xe_k^T + \theta_{t+1}^i(x_{t+1}) x_{et+1} x_{et+1}^T \\ &= \overline{R^i}(t) + \sum_{k=1}^{t} A_{t,k}^i xe_k^T + \theta_{t+1}^i(x_{t+1}) xe_{t+1} xe_{t+1}^T, \end{aligned} \tag{10}$$

and

$$\begin{aligned} f^i(t+1) &= \sum_{k=1}^{t} \theta_{t+1}^i(x_k) xe_k y_k + \theta_{t+1}^i(x_{t+1}) xe_{t+1} y_{t+1} \\ &= f^i(t) + \sum_{k=1}^{t} A_{t,k}^i y_k + \theta_{t+1}^i(x_{t+1}) xe_{t+1} y_{t+1}, \end{aligned} \tag{11}$$

where

$$A_{t,k}^i = \frac{\gamma_{t+1}^i(x_k)(-\gamma_{t+1}^{R_{t+1}}(x_k))}{(\sum_{l=1}^{R_{t+1}} \gamma_{t+1}^l(x_k))(\sum_{l=1}^{R_t} \gamma_{t+1}^l(x_k))} xe_k, \tag{12}$$

$$\overline{R^i}(t) = \sum_{k=1}^{t} \theta_t^i(x_k) xe_k xe_k^T \quad and \quad f^i(t) = \sum_{k=1}^{t} \theta_t^i(x_k) xe_k y_k. \tag{13}$$

$$\begin{aligned}\psi^i_{t+1} &= \bar{R}^{-1}(t+1)f(t+1) \qquad (14)\\ &= \Psi^i_t - \overline{R}^{-1}(t+1)\{[\sum_{k=1}^{t} A^i_{t,k} x e^T_k + \theta^i_{t+1}(x_{t+1}) x e_{t+1} x e^T_{t+1}]\psi^i_t \\ &\quad - \sum_{k=1}^{t} A^i_{t,k} y_k - \theta^i_{t+1}(x_{t+1}) x e_{t+1} y_{t+1}\}.\end{aligned}$$

According to 14, if *condition 4.1*, is satisfied, then we can get a simplified version of ψ^i updating equation shown in (15).

Condition 4.1. *If $\gamma^i_t(x_{t+1}) < \varepsilon$, then a new cluster would be added with center $c_{R_{t+1}} = x_{t+1}$, radius $\sigma^2_{R_{t+1},j} = n\min\{\frac{\min\{(c_{i,j}-k_0\sigma_{i,j}-c_{R_{t+1},j})^2,\ (c_{i,j}+k_0\sigma_{k,j}-c_{R_{t+1},j})^2\}}{-2\log\frac{\varepsilon_0}{t}}\}$, where $j = 1,2,\ldots,n$, $i = 1,2,\ldots,R$, $k_0 = \sqrt{-\frac{2\log(\varepsilon)}{n}} > 0$, $\varepsilon > 0$, $\varepsilon_0 > 0$ is the tolerance degree.*

In *condition 4.1*, the radius $\sigma_{R_{t+1},j}$ of the new cluster is initialized to make sure that these fuzzy clusters have low overlapping degree and avoid rule confliction. Therefore, data points in other clusters have low activation degree in the new added cluster with center $c_{R_{t+1}}$. To ensure the optimality, it is necessary to make sure any historical data x has the activation degree smaller than $\frac{\varepsilon_0}{t}$ in the new cluster (i.e. $\gamma^{R_{t+1}}(x) \leq \frac{\varepsilon_0}{t}$). Further, as $\gamma^{R_{t+1}}$ is a multivariate Gaussian density, hence, on each dimension, the fact that $\sigma^2_{R_{t+1},j} = n\min\{\frac{\min\{(c_{i,j}-k_0\sigma_{i,j}-c_{R_{t+1},j})^2,\ (c_{i,j}+k_0\sigma_{k,j}-c_{R_{t+1},j})^2\}}{-2\log\frac{\varepsilon_0}{t}}\}$ holds could meet the requirement of $\gamma^{R_{t+1}}(x) \leq \frac{\varepsilon_0}{t}$.

Furthermore, control parameter ε is set based on the α-cut [57] of the fuzzy sets. The value for ε is 0.05. In this paper the α-cut of each fuzzy set i could be presented by $\mathscr{A}^i_\alpha = \{x \in \Gamma^i | \gamma^i(x) \geq \alpha\}$, where $i = 1,2,\ldots,R$, $\alpha = \varepsilon = 0.05$. *Condition 4.1* indicates that if there exists a data point x^* which is outside all the α-cuts of fuzzy sets Γ^i, $i = 1,2,\ldots,R$, then there is a need to build a new fuzzy rule ($R+1$-th rule) such that $x^* \in \mathscr{A}^{R+1}_\alpha$. This condition assures that $\{x_t\}^T_{t=1} \subset \bigcup_i \mathscr{A}^i_\alpha$, in which $\{x_t\}^T_{t=1}$ stands for the set formed by all the historical data points. Besides, tolerance degree ε_0 is a small value to ensure the optimality of consequent parameters, but ε_0 can not be set too small in case the radius of the new added rule becomes 0.

Definition 4.1. ε-completeness *[55]: A fuzzy partition for feature X_i which contains*

fuzzy sets $\widetilde{A}_1, \widetilde{A}_2, \ldots, \widetilde{A}_R$ *is regarded to be* ε*-completeness, if there does not exist a point* $x \in [\min(X_i), \max(X_i)]$ *such that* $\mu_{\widetilde{A}_i}(x) < \varepsilon$*, where* $\varepsilon > 0$ *is a small number.*

Theorem 4.1. *Let* condition 4.1 *be satisfied and* R_{t+1}*-th rule be the new added rule. Then* ψ^i_{t+1} *which minimizes (6) could be updated by*

$$\psi^i_{t+1} = \psi^i_t + \overline{R^i}(t+1)^{-1}\theta^i_{t+1}(x_{t+1})xe_{t+1}\{y_{t+1} - xe^T_{t+1}\psi^i_t\}, \tag{15}$$

for all $i = 1, 2, \ldots, R_{t+1} - 1$*, and* ε*-completeness in* definition 4.1 *holds.*

4.1.2. Fuzzy rule pruning

In order to avoid over-fitting, a fuzzy rule pruning procedure is proposed. Similar to section 4.1.1, we are aiming to get the appropriate condition, under which the optimum solution of Err_i could be calculated by simple recursive steps. Assuming that the i^*-th fuzzy rule is pruned at time $t+1$ and for $k = 1, 2, \ldots, t$, $R_1 = R_2 = \ldots = R_t$ holds, and $R_{t+1} = R_t - 1$. In order to optimize (6), the form of $\overline{R^i}(t+1)$ and $f^i(t+1)$ are listed in (16) and (17),

$$\overline{R^i}(t+1) = \overline{R^i}(t) + \sum_{k=1}^{t} B^i_{t,k}xe^T_k + \theta^i_{t+1}(x_{t+1})xe_{t+1}xe^T_{t+1}, \tag{16}$$

$$f^i(t+1) = f^i(t) + \sum_{k=1}^{t} B^i_{t,k}y_k + \theta^i_{t+1}(x_{t+1})xe_{t+1}y_{t+1}, \tag{17}$$

in which

$$B^i_{t,k} = \frac{\gamma^i_{t+1}(x_k)\gamma^{i^*}_t(x_k)}{(\sum_{l=1}^{R_t}\gamma^l_t(x_k))(\sum_{l=1,\, l\neq i^*}^{R_t}\gamma^l_{t+1}(x_k))}xe_k. \tag{18}$$

From the expressions of $\overline{R^i}(t+1)$ and $f^i(t+1)$, it can be seen that if *condition 4.2* is satisfied, then the EWRLS optimum solution of (6) could be obtained by (19) in *theorem 4.2*.

Condition 4.2. *If* $\sum_{p=t^*+1}^{k}\gamma^{i^*}_p(x_p) < \varepsilon_p$*, then the* i^**-th fuzzy rule is removed where* $k - t^* > N$*,* $N > 0$ *is a large number,* t^* *is the time that the* i^**-th fuzzy rule built,* k *is the step,* $\varepsilon_p > 0$ *is the tolerance degree.*

Threshold ε_p for cumulative activation degree is set to be a small value 0.05. The threshold N is suggested to be larger than 50, as sample size above 50 could be regarded as large sample. Too small threshold may lead to some useful fuzzy rules being deleted too early, which makes it hard to guarantee the optimality of the consequent parameters. Too large threshold allows the redundant fuzzy rules being used for a long time, which increases the computational burden. Whereas, for online tracking applications, N could be set smaller than 50 to satisfy the need of giving quick one-step-ahead predictions and putting the global optimality on a secondary status.

Theorem 4.2. *Let* condition 4.2 *be satisfied, and the i^*-th rule be removed at step $t+1$. Then ψ_{t+1}^i which minimizes (6) could be updated by*

$$\psi_{t+1}^i = \psi_t^i + \overline{R^i}(t+1)^{-1}\theta_{t+1}^i(x_{t+1})xe_{t+1}\{y_{t+1} - xe_{t+1}^T\psi_t^i\}, \tag{19}$$

for all $i = 1, 2, \ldots, R_{t+1} - 1$.

4.1.3. Fuzzy rule merging

Because of the updating procedure of the cluster centers and their radiuses, so there are likely some clusters evolved similar after certain steps. In this situation, it is no longer necessary and appropriate to apply several fuzzy rules to approximate the data points in the same cluster, as this may cause conflicting of rules and increase the computation burden. Following the same process as section 4.1.1 and 4.1.2, we assume that the l_1-th and l_2-th fuzzy rule are merged in step $t+1$. The merged fuzzy rule is the l_0-th and $l_0 = l_1$ numerically. Therefore, $\overline{R^i}(t+1)$ and $f(t+1)$ could be presented by (20) and (21),

$$\overline{R^i}(t+1) = \overline{R^i}(t) + \sum_{k=1}^{t} C_{t,k}^i xe_k^T + \frac{\gamma_{t+1}^i(x_{t+1})}{\sum_{l=1}^{R_{t+1}} \gamma_{t+1}^l(x_{t+1})} xe_{t+1}xe_{t+1}^T, \tag{20}$$

$$f^i(t+1) = f^i(t) + \sum_{k=1}^{t} C_{t,k}^i y_k + \frac{\gamma_{t+1}^i(x_{t+1})}{\sum_{l=1}^{R_{t+1}} \gamma_{t+1}^l(x_{t+1})} xe_{t+1}y_{t+1}, \tag{21}$$

$$C_{t,k}^i = \frac{\gamma_t^i(x_k)[\gamma_t^{l_1}(x_k) + \gamma_t^{l_2}(x_k) - \gamma_t^{l_0}(x_k)]}{[\sum_{l=1}^{R_{t+1}} \gamma_{t+1}^l(x_k)][\sum_{l=1}^{R_t} \gamma_t^l(x_k)]} xe_k. \tag{22}$$

From (20) and (21), we can get *theorem 4.3* under *condition 4.3*, .

Condition 4.3. *When $k = t+1$ and there exist $\varepsilon_c > 0$ and $\varepsilon_\sigma > 0$ such that $\|c_{l_1} - c_{l_2}\| < \varepsilon_c$ and $\|\sigma_{l_1}^2 - \sigma_{l_2}^2\| < \varepsilon_\sigma$, where $\|\cdot\|$ is the Euclidean norm, then l_1-th and the l_2-th fuzzy rule are merged to l_0-th fuzzy rule ($l_1 = l_0$ numerically). The center and the radius of the l_0-th cluster is calculated by (23) and (24),*

$$c_{l_0} = \frac{\sum\limits_{k_0} x_{k_0}}{N_{l_0}} = \frac{N_{l_1} c_{l_1} + N_{l_2} c_{l_2}}{N_{l_1} + N_{l_2}}, \tag{23}$$

$$(\sigma_{l_0,j})^2 = \frac{\sum\limits_{k_0} (x_{k_0,j} - c_{l_0,j})^2}{N_{l_0} - 1} = \frac{(N_1 - 1)(\sigma_{l_1,j})^2 + (N_2 - 1)(\sigma_{l_2,j})^2 + \frac{N_{l_1} N_{l_2} (c_{l_1,j} - c_{l_2,j})^2}{N_{l_1} + N_{l_2}}}{N_{l_1} + N_{l_2} - 1}. \tag{24}$$

The statistical theory behind the above two formulas (23) and (24) used in *condition 4.3* are the unbiased estimators $\bar{X}$ and S^2 for expected value $E(X)$ and variance $Var(X)$ of a random variable X, separately. The exact formulas are displayed in (25).

$$\bar{X} = \frac{1}{N} \sum_{i=1}^{N} X_i, \quad S^2 = \frac{1}{N-1} \sum_{i=1}^{N} (X_i - \bar{X})^2, \tag{25}$$

where $X_1, \ldots, X_N$ are independent and identically distributed (i.i.d) random variables. Mathematical deduction for (24) is presented in the appendix.

In practice, it is hard to choose very small control parameters ε_c and ε_σ to assure that two clusters have high overlapping level. Therefore, an alternative criterion (26) for merging rule l_1 and l_2 is given. From (26), it can be seen that the more ε_c^* and ε_σ^* close to 1, the larger overlapping part of the corresponding two clusters.

$$\begin{aligned} &1 - \max_{k=1,2,\ldots,n} \{\max\{\frac{\|c_{l_1,k} - c_{l_2,k}\|}{\sigma_{l_1,k}}, \frac{\|c_{l_1,k} - c_{l_2,k}\|}{\sigma_{l_2,k}}\}\} > \varepsilon_c^*, \\ &\min_{k=1,2,\ldots,n} \{\min\{\frac{\sigma_{l_1,k}}{\sigma_{l_2,k}}, \frac{\sigma_{l_2,k}}{\sigma_{l_1,k}}\}\} > \varepsilon_\sigma^*, \end{aligned} \tag{26}$$

where ε_c^* and ε_σ^* are control parameters indicating the similarity of two clusters. ε_c^* is selected based on the α-cut ($\alpha = 0.95$) of the fuzzy set with $\varepsilon_c^* = \sqrt{-\frac{2\log(0.95)}{n}}$, and ε_σ^* is 0.9.

Theorem 4.3. *Let* condition 4.3 *be satisfied, and the l_1-th and the l_2-th fuzzy rule be merged to l_0-th ($l_0 = l_1$ numerically) rule at time $t+1$. Then ψ^i_{t+1} which minimizes (6) could be updated by (27) to (30),*

$$\psi^i_{t+1} = \widetilde{\psi}^i_{l_2}(t) + \widetilde{R^i_{l_2}}(t)^{-1}\theta^i_{t+1}(x_{t+1})xe_{t+1}\{y_{t+1} - xe^T_{t+1}\widetilde{\psi}^i_{l_2}(t)\}, \tag{27}$$

where $\widetilde{\psi}^i_{l_2}(t)$ and $\widetilde{P^i_{l_2}}(t) = \widetilde{R^i_{l_2}}(t)^{-1}$ are shown in (28) and (30)

$$\widetilde{\psi}^i_{l_p}(t) = \widetilde{\psi}^i_{l_p}(t-1) + \widetilde{L^i_{l_p}}(t)\{y_t - xe^T_t\widetilde{\psi}^i_{l_p}(t-1)\}, \tag{28}$$

$$\widetilde{P^i_{l_p}}(t) = \widetilde{P^i_{l_p}}(t-1) - \frac{M^i_{l_p}(x_t)\widetilde{P^i_{l_p}}(t-1)xe_t xe^T_t \widetilde{P^i_{l_p}}(t-1)}{1 + M^i_{l_p}(x_t)xe^T_t\widetilde{P^i_{l_p}}(t-1)xe_t}, \tag{29}$$

$$\widetilde{L^i_{l_p}} = \frac{M^i_{l_p}(x_t)\widetilde{P^i_{l_p}}(t)xe_t}{1 + M^i_{l_p}(x_t)xe^T_t\widetilde{P^i_{l_p}}(t-1)xe_t}, \tag{30}$$

with $\widetilde{R^i_{l_p}}(t) = \sum_{k=1}^{t} M^i_{l_p}(x_k)xe_k xe^T_k$, $M^i_{l_p}(x_k) = \theta^i_t(x_k) + \frac{\gamma^i_t(x_k)\gamma^{l_p}_t(x_k)}{[\sum_{l=1}^{R_t}\gamma^l_t(x_k)][\sum_{l=1}^{R_t}\gamma^l_t(x_k) - \gamma^{l_p}_t(x_k)]}$, $l_p = l_2$.

4.2. Parameters learning

Not only the structure of LEOA is changing overtime, but the parameters of the system are updating while there is new information comes. This section mainly introduces how LEOA updates its parameters using the new information from the new coming inputs and the known outputs. This process includes learning the antecedent parameters and the consequent parameters.

4.2.1. Antecedent parameters learning

Based on the antecedent learning procedure for minimizing (6), the following condition, *condition 4.4*, could be obtained to update the existing cluster centers and radiuses.

Condition 4.4. *If there exists an $i^* \in \{1,2,\ldots,R_t\}$ and $\gamma^{i^*}_t(x_{t+1}) \geq \varepsilon$ ($\varepsilon > 0$ is the same value as what is used in* condition *4.1), then cluster center c_{i^*} and radius σ_{i^*} can be*

updated by (31),

$$\begin{aligned} c_{i^*,j}(t+1) &= c_{i^*,j}(t) + \frac{x_{t+1,j} - c_{i^*,j}(t)}{N_{i^*}(t)+1} \\ (\sigma_{i^*,j}(t+1))^2 &= (\sigma_{i^*,j}(t))^2 + \frac{(x_{t+1,j} - c_{i^*,j}(t+1))^2 - (\sigma_{i^*,j}(t))^2}{N_{i^*}(t)} \\ &\quad + (c_{i^*,j}(t+1) - c_{i^*,j}(t))^2, \end{aligned} \tag{31}$$

in which $N_{i^*}(t)$ is the total number of data points which are satisfies $\gamma_t^{i^*}(x_{t+1}) \geq \varepsilon$ until time t. (Note that what only have to be assured are $c_{i^*} \to c_{i^*}^0$ and $\sigma_{i^*} \to \sigma_{i^*}^0$ when $N_{i^*} \to \infty$, where $c_{i^*}^0$ and $\sigma_{i^*}^0$ are the real cluster center and radius of the i^*-th cluster.)

Updating formulas of cluster centers and radiuses in (31) are also deducted from (25).

4.2.2. Consequent parameters learning

Recursive least square methods are widely used in control theory to minimize the error function [58–61]. In order to serve for the same optimality purpose as the antecedent learning procedure of LEOA in section 4.1, the consequent parameters are obtained by a special type of weighted recursive least square method known as extended weighted recursive (EWRLS) method. According to *theorem 4.1, 4.2, 4.3* in section 4.1, the form of EWRLS would be influenced by whether there is a merging process happened. Detailed EWRLS updating formulas are shown in (32) and (33). If there is no fuzzy rule merging process, consequent parameters should be learned by formulas (32),

$$\begin{aligned} \psi_{t+1}^i &= \psi_t^i + L^i(t+1)(y_{t+1} - xe_{t+1}^T \psi_t^i), \\ L^i(t+1) &= \frac{\theta_{t+1}^i(x_{t+1}) P^i(t) xe_{t+1}}{1 + \theta_{t+1}^i(x_{t+1}) xe_{t+1}^T P^i(t) xe_{t+1}}, \\ P^i(t+1) &= P^i(t+1) - \frac{\theta_{t+1}^i(x_{t+1}) P^i(t) xe_{t+1} xe_{t+1}^T P^i(t)}{1 + \theta_{t+1}^i(x_{t+1}) xe_{t+1}^T P^i(t) xe_{t+1}}. \end{aligned} \tag{32}$$

Otherwise, assume that two fuzzy rules l_q and l_p are merged to l_0, the EWRLS updating

formulas are changed to (33)

$$\begin{aligned} \psi^i_{t+1} &= \widetilde{\psi^i_{l_p}}(t) + \overline{L^i}(t+1)(y_{t+1} - xe^T_{t+1}\widetilde{\psi^i_{l_p}}(t)), \\ \overline{L^i}(t+1) &= \frac{\theta^i_{t+1}(x_{t+1})\widetilde{P^i_{l_p}}(t)xe_{t+1}}{1+\theta^i_{t+1}(x_{t+1})xe^T_{t+1}\widetilde{P^i_{l_p}}(t)xe_{t+1}}, \\ P^i(t+1) &= \widetilde{P^i_{l_p}}(t) - \frac{\theta^i_{t+1}(x_{t+1})\widetilde{P^i_{l_p}}(t)xe_{t+1}xe^T_{t+1}\widetilde{P^i_{l_p}}(t)}{1+\theta^i_{t+1}(x_{t+1})xe^T_{t+1}\widetilde{P^i_{l_p}}(t)xe_{t+1}}, \end{aligned} \tag{33}$$

where $\widetilde{\psi^i_{l_p}}(t)$ and $\widetilde{P^i_{l_p}}(t)$ are updated by (28) and (29). It needs to be noticed that $\widetilde{\psi^i_{l_p}}(t)$ and $\widetilde{P^i_{l_p}}(t)$ should be calculated and updated since $k = 3$, and each step some extra information needs to be recorded. For each $i = 1,2,\ldots,R_k$, $l = 1,2,\ldots,R_k$, $l \neq i$, $\widetilde{\psi^i_l}(t)$ and $\widetilde{P^i_l}(t)$ are recorded and used when fuzzy rule merging happens.

5. Mechanism of LEOA

Following the description of the structure and parameter learning method of LEOA in section 4, the overall online learning and predicting process of LEOA could be summarized in the learning steps and the flowchart (Figure 1) below.

Before displaying the specific steps, there are two parameters, which act as indicators, need to be introduced firstly. Parameter ind_a is the indicator for fuzzy rule adding. The default value of ind_a is 0 which stands for no new rule is added. Once there is a new fuzzy rule being built, then the value of ind_a should be turned to $ind_a = 1$. Similar to ind_a, with the default value 0, ind_m is the indicator of fuzzy rule merging process. If there exist two fuzzy rules being merged at the current time step, then the value of ind_m should be changed to 1. Otherwise, ind_m keeps the default value with $ind_m = 0$.

step 1 Read new input data x_{t+1}, set $ind_a = 0$ and $ind_m = 0$.

step 2 *Rule adding and updating*

If $\gamma^i_t(x_{t+1}) < \varepsilon$ for all $i = 1,2,\ldots,R_t$, then add a new fuzzy rule $R_{t+1} = R_t + 1$, change $ind_a = 1$. Otherwise, update the cluster center and radius of the i^*-th rule by (31), where $i^* = \arg\max_i\{\gamma^i_t(x_{t+1})\}$.

step 3 *Rule pruning*

If $t_{max} = \max(t+1-t_i^*) > N$, $i^* = \arg\max_i (t+1-t_i^*)$ and $\sum_{k=t_{i^*}^*}^{t+1} \gamma_t^{i^*}(x_k) < \varepsilon_p$, then remove the i^*-th rule.

step 4 *Rule merging*

If $ind_a = 0$, and the i^*-th rule antecedent is updated, then this merging step is considered. If there is an $l_q \in \{1,2,\ldots,R_t\}$ and $l_q \neq i^*$ such that c_{l_p}, c_{l_q}, σ_{l_p} and σ_{l_q} satisfies (26), then the l_q-th and the i^*-th fuzzy rule are merged to l_0-th fuzzy rule ($i^* = l_0$ numerically) with center and the radius calculated by (23) and (24), $ind_m = 1$.

step 5 *Output*

Compute output by $\hat{y}_{t+1} = \sum_{i=1}^{R_{t+1}} \theta_{t+1}^i(x_{t+1}) xe_{t+1} \psi_t^i$.

step 6 *RLS*

If $ind_m = 0$, then use (32) to update the consequent parameters ψ_l^i, otherwise apply (33). For each $i = 1,2,\ldots,R_{t+1}$ and $l = 1,2,\ldots,R_{t+1}$ with $l \neq i$, update $\widetilde{\psi_l^i}(t)$ and $\widetilde{P_l^i}(t)$ using (28) to (29).

6. Complexity Analysis

In this section, the computational burden of proposed algorithm LEOA is discussed. Similar to many previous online learning algorithms, the complexity of LEOA depends on the number of fuzzy rules R and the dimension of input n. From the structure of LEOA and the details in each step, it can be seen that the complexity of the whole process of fuzzy rule adding, merging, pruning process is $O(Rn)$. Whereas, the new parameter updating procedure has the complexity $O(R^2)$, which will increase the complexity if the number of fuzzy rules R is larger than the dimension of the input space n. The reason behind this is that the new proposed EWRLS in section 4.2.2 requires to remember more information than the widely used WRLS method, in order to meet the requirement of global accuracy. Comparing with other state-of-the-art algorithms such as eTS [10], DENFIS [9], FLEXFIS[12], DEC [51], PANFIS [36], Gen-Smart-EFS [39], LEOA requires to store more historical knowledge and has more complex

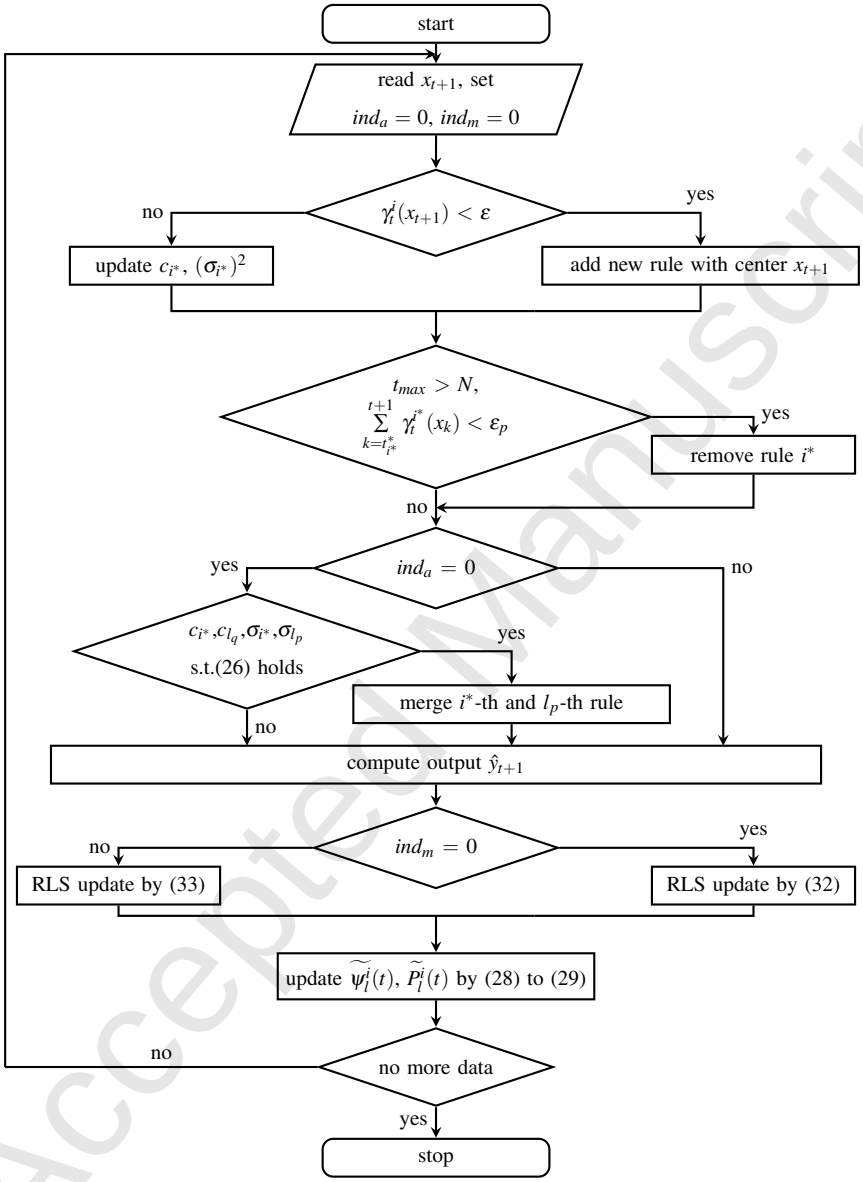


Figure 1: Computational steps of LEOA.

structure when $R > n$. Nonetheless, LEOA takes the affecting of antecedent structure changing to the convergency of consequent parameters into consideration, and assures the optimality of consequent parameters instead of sub-optimality presented in most of

previous existing approaches. The high computational complexity is a price to pay for achieving the optimality.

7. Sensitivity Analysis

There are two control parameters (ε_0 and N) need to be selected when using LEOA to make predictions. Varies values of ε_0 and N are chosen to assure LEOA is not problem-dependent. Thus, the well known Box-Jenkins gas furnace data set [36, 51, 62–67] is cast to investigate whether LEOA is sensitive to ε_0 and N. The Box-Jenkins data set consists of 290 input-output pairs. Methane flow rate $u(t)$, and CO_2 concentration in off gas $y(t)$ comprise the input of each data sample. The output of the process is $y(t)$. From previous studies, it can be seen that the best model of this process is (34),

$$y(t) = f(y(t-1), u(t-4)). \tag{34}$$

LEOA runs in an online mode on this data set. All 290 outputs are used to evaluate the performance of LEOA. In *section 4.1.1*, control parameter ε_0 is suggested to be a small value, hence ε_0 is varying in $[10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}]$. Besides, sample size control parameter N is opted as 50, 70, 90, 110, 130. Remark that other control parameters are kept as their default settings. Predicting results of the testing data are presented in Table 1. To test the global accuracy of LEOA, then we use the final model learned from the data to make estimations of all the historical data. Table 2 displays the results of fitting all the historical data. Both predictions of testing data and historical data are evaluated by fuzzy rule numbers, which are shown in the bracket in Table 1 and 2, and non-dimensional error index (NDEI).

It is obvious from Table 1 that different values of ε_0 and N affect little to the performance of LEOA. Table 2 shows that larger N helps the model learned by LEOA keeping more historical information, thus giving smaller NDEIs. However, it always causes a more complex fuzzy rule base of the system and heavier computational burden. Besides, too small value of ε_0 does not promote the learned system to give better global fitting results. As a result, in the following numerical examples ε_0 is chosen from 0.1, 0.01, 0.001, and N is selected smaller than 100, in order to keep balance between achieving accuracy and lowering down computational burden.

Table 1: Sensitivity of predefined parameters.

	$\varepsilon_0 = 10^{-1}$	$\varepsilon_0 = 10^{-2}$	$\varepsilon_0 = 10^{-3}$	$\varepsilon_0 = 10^{-4}$	$\varepsilon_0 = 10^{-5}$
$N = 50$	0.2192(42)	0.2197(39)	0.2199(38)	0.2199(38)	0.2200(38)
$N = 70$	0.2206(59)	0.2209(60)	0.2209(58)	0.2209(58)	0.2209(58)
$N = 90$	0.2209(78)	0.2199(78)	0.2200(78)	0.2200(78)	0.2200(78)
$N = 110$	0.2210(91)	0.2161(90)	0.2185(89)	0.2182(92)	0.2199(92)
$N = 130$	0.2185(107)	0.2141(105)	0.2163(104)	0.2162(107)	0.2179(108)

Table 2: Sensitivity of predefined parameters to global optimality.

	$\varepsilon_0 = 10^{-1}$	$\varepsilon_0 = 10^{-2}$	$\varepsilon_0 = 10^{-3}$	$\varepsilon_0 = 10^{-4}$	$\varepsilon_0 = 10^{-5}$
$N = 50$	0.2265(42)	0.2698(39)	0.2896(38)	0.2896(38)	0.2896(38)
$N = 70$	0.2210(59)	0.2554(60)	0.2744(58)	0.2745(58)	0.2745(58)
$N = 90$	0.2067(78)	0.2411(78)	0.2412(78)	0.2446(78)	0.2458(78)
$N = 110$	0.1921(91)	0.2256(90)	0.2338(89)	0.2362(92)	0.2364(92)
$N = 130$	0.1871(107)	0.2264(105)	0.2340(104)	0.2362(107)	0.2363(108)

8. Numerical examples

Four numerical examples include classical benchmark examples and real-world data predictions are carried out to assess LEOA. The data sets are *DJIA* daily closing prices, data sets generated by two non-linear dynamic systems, and Mackey-Glass chaotic time series. The motivation for testing LEOA on these data sets are the essence of non-stationary, non-linear, as well as uncertainty of these data sets. These data sets can effectively evaluate the learning ability and the global optimality of LEOA. In the following subsections, LEOA is compared against varies kinds of state-of-the-art algorithms. The performances of the algorithms are evaluated by the rooted mean square error (RMSE) and non-dimensional error index (NDEI), which are

$$RMSE = \frac{\sum_{k=1}^{n} (y_k - \hat{y}_k)^2}{n}, \tag{35}$$

$$NDEI = \sqrt{\frac{RMSE}{std(y)}}, \tag{36}$$

respectively. $std(y)$ is the standard deviation of $\{y_k\}_{k=1}^{n}$. Results of LEOA are calculated in the environment of intel(R) core (TM) i7-4790 CPU with a 3.6 GHz processor and 16.0 GB memory.

8.1. Example 1: online prediction of DJIA daily closing price

This example is an online learning example applied to demonstrate that LEOA can make both one-step-ahead predictions and global approximations accurately. In this example, LEOA and MEPL (model 1) [48] are used to make online prediction of Dow Jones Industrial Average (DJIA) daily closing price. The DJIA data are collected from Wharton Research Data Services (WRDS) between 04.01.1960 and 31.12.2007, which provide 12118 data points for this experiment. Both LEOA and MEPL make predictions of the logarithm of the data set based on the following three models, $M1$, $M2$ and $M3$, shown in (37), (38) and (39), respectively.

$$M1: \ y_{t+1} = f(y_{t-3}, y_{t-2}, y_{t-1}, y_t), \tag{37}$$

$$M2: \ y_{t+1} = f(y_{t-4}, y_{t-3}, y_{t-2}, y_{t-1}, y_t), \tag{38}$$

$$M3: \ y_{t+1} = f(y_{t-5}, y_{t-4}, y_{t-3}, y_{t-2}, y_{t-1}, y_t), \tag{39}$$

where y_k presents the logarithm closing price of DJIA, and $\{y_k\}_{k=1}^{12118}$ stands for all these data. Time delays are chosen based on the cases that there are not very huge difference between the number of fuzzy rules and parameters used to make predictions. With this model chosen, it is more visible to compare the one-step-ahead and global accuracy of these two algorithms. All of the results of these two algorithms are obtained by using the same group of control parameters. Both one-step-ahead and global prediction results are listed in Table 3. The RMSEs in Table 3 is calculated based on the one-step-ahead predictions with the structure and parameters of the model used in

Table 3: RMSEs of example 1

	Method	model type	RMSEs	global RMSEs	rule num.	parameter num.
$M1$ (37)	MEPL[48]	T-S	0.0119	0.0417	16	64
	LEOA	T-S	0.0133	0.0133	27	108
	Method	model type	RMSEs	global RMSEs	rule num.	parameter num.
$M2$ (38)	MEPL[48]	T-S	0.0119	0.0423	20	80
	LEOA	T-S	0.0139	0.0138	22	88
	Method	model type	RMSEs	global RMSEs	rule num.	parameter num.
$M3$ (39)	MEPL[48]	T-S	0.0121	0.0395	16	64
	LEOA	T-S	0.0145	0.0144	22	88

prediction are varies from time to time; while global RMSEs is computed by using the estimation of every historical data point. The estimation of historical data points are computed by the final system learned from the data without any structure and parameters updating of the system. Large global RMSEs indicates that there is an "unlearning effect". The control parameters of LEOA are $\varepsilon_0 = 10^{-3}$, $N = 20$. RMSEs in Table 3 shows that both LEOA and MEPL can make accurate one-step-ahead predictions, and MEPL always performs a little bit better than LEOA in making one-step-ahead predictions. Besides, LEOA gives accurate global prediction results which are similar to its one-step-ahead prediction results for all these three models. Nonetheless, it is obvious that the number of fuzzy rules for LEOA used to make predictions is always larger than MEPL. This is cause by that MEPL minimize the error function (6) to get the estimation of the consequent parameters by supposing the fuzzy rules are unchanged. Besides, MEPL mainly focuses on tracking the behavior of the most recent data, and puts global behavior of the algorithm in an secondary place. But the starting point for designing LEOA is how to make accurate one-step-ahead predictions without losing too much memory of the knowledge learnt from the historical data. Furthermore, take

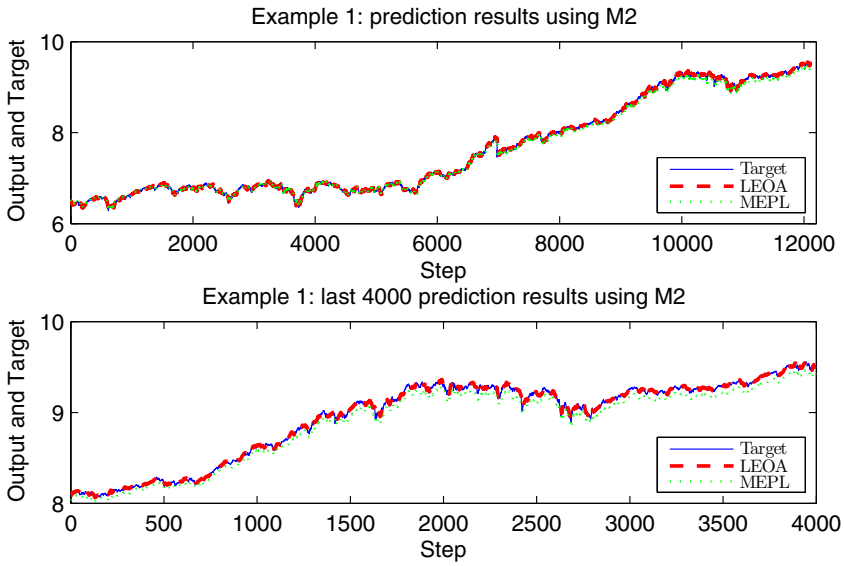


Figure 2: Example 1: predictions for all historical data using $M2$ (38).

$M2$ as an exmple, Figure 2 depicts the estimation results of all the historical data using the final models learned by LEOA and MEPL. It can be seen from Figure 2, it is significant that LEOA reserves a more accurate prediction model than MEPL. As a result, this example further verifies LEOA can achieve satisfactory local and global accuracy.

8.2. Example 2: nonlinear dynamic plant

A nonlinear system identification problem is applied to evaluate the performance of LEOA. The model form of nonlinear dynamic plant is shown in Eq.(40),

$$y(t+1) = \frac{y(t)y(t-1)[y(t)-0.5]}{1+y^2(t)+y^2(t-1)} + u(t), \tag{40}$$

where $u(t) = \sin(2t/25)$, $t \in [1,5200]$. The first 5000 pairs of $(y(t-1), y(t), u(t))$ are used for training and the remaining 200 pairs are applied for forecasting. Although LEOA does not need a training phase, in order to make comparison with other algorithms, LEOA runs in an online mode on the training data set and stops to evolve its structure after this phase. The numerical results for predicting the testing data samples

Table 4: RMSEs of example 2

Method	model type	RMSEs	rule num.	parameter	run time(s)
eTS[10]	T-S	0.0212	49	490	–
simpl_eTS[62]	T-S	0.0225	22	220	–
SAFIS[13]	RBF	0.0221	17	119	–
FLEXFIS var A[12]	T-S	0.0176	5	50	–
FLEXFIS var B[12]	T-S	0.0171	8	80	–
SOFMLS[11]	mamdani	0.0201	5	35	–
eMG[29]	T-S	0.0058	5	80	–
OSAMNN[44]	neural network	0.0206	40	400	–
DeTS[51]	T-S	0.0172	4	40	–
LEOA	T-S	0.0029	20	200	39.021154

are displayed in Table 4[2]. It can be seen from Table 4, LEOA proposed in this paper has much higher accuracy than other previous methods according to the predicting RMSEs. Figure 3 plots the prediction results for the 200 testing data. Figure 3 indicates that LEOA is a powerful tool to make accurate predictions. Furthermore, we use the final model to make estimations of all the 5200 historical data points, the predicting RMSE reaches at 0.0033 which is still a small value. Figure 4 presents the prediction errors of LEOA for the first 1000 historical data. It is obvious that LEOA only gives inaccurate predictions for the first 10 data, and after that the prediction errors fluctuate in a very low level. The second figure in Figure 4 portrays the trace of the prediction errors from the 101st to 1000th data, which shows that prediction errors become stable and fluctuate between -0.0085 and 0.0073.

[2]"–" stands for having not been listed in the source paper.

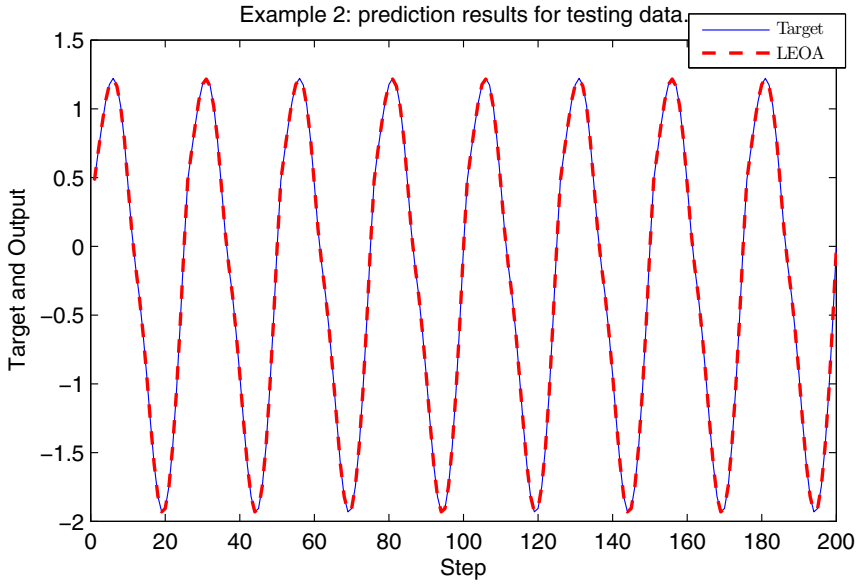


Figure 3: Example 2: prediction results for testing data.

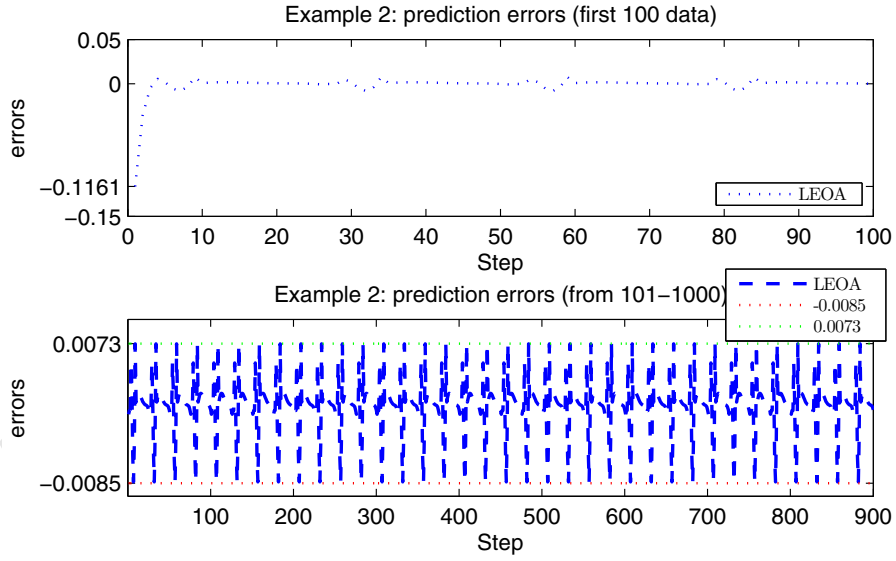


Figure 4: Example 2: prediction errors for all historical data.

8.3. Example 3: high-dimensional system

In this example, LEOA is tested on identifying a high-dimensional system. The system is defined by the following equation (40):

$$y(t) = \frac{\sum_{i=1}^{m} y(t-i)}{1+\sum_{i=1}^{m} y^2(t-i)} + u(t-1), \tag{41}$$

where $u(t) = \sin(2\pi t/20)$, $m = 10$, $y(j) = 0$, for $j = 1, 2, \ldots, m$. The prediction model is of the form shown in (42),

$$\hat{y}(t) = f(y(t-1), y(t-2), \ldots, y(t-10), u(t-1)). \tag{42}$$

There are 3300 data points produced with $t \in [1, 3300]$, in which only the last 300 data points are used to evaluate the performance of the algorithms. In this example, $(y(t-1), y(t-2), \ldots, y(t-10), u(t-1))$ are used to forecast $y(t)$. Similar to example 2 in section 8.2, LEOA also runs in an online mode. The prediction results for the testing data are displayed in Table 5. It is obvious that LEOA can make better predictions than other previous methods judged by RMSEs. It should be noticed that we use 61 fuzzy rules which is larger than other models that being used to make comparison. The reason behind this is that LEOA could make sure that for each of the fuzzy rule its consequent parameters are selected as the global optimum. To verify this, the final system learned from the data is kept unchanged in both structure and parameters to make prediction of the whole 3300 data and get predicting RMSE of 0.0620. This means that LEOA remembers its previous behavior well while evolving its structure and updating its parameters applying the information extracting from the new data. Figure 5 shows the prediction results for the 300 testing data and the estimation results of first 300 training data. Figure 5 and the numerical results indicate that LEOA is a powerful algorithm for making significantly accurate predictions without forgetting too much historical information.

8.4. Example 4: Mackey-Glass chaotic time series.

Mackey-Glass chaotic time series [10, 14, 37, 42, 51] is a classical benchmark example to assess different evolving fuzzy systems. Mackey-Glass time series is gen-

Table 5: RMSEs of example 3

Method	model type	RMSEs	rule num.	parameter	run time(s)
FLEXFIS[12]	T-S	0.0085	15	510	–
eTS[10]	T-S	0.0075	14	476	–
eMG($\sum_{init} = 2 \times 10^{-1} I_{11}$)[29]	T-S	0.0288	9	1296	–
eMG($\sum_{init} = \times 10^{-1} I_{11}$)[29]	T-S	0.0050	13	1872	–
Gen-smart-EFS(fac=0.6)[39]	T-S	0.0042	9	1296	–
LEOA	T-S	0.0027	61	2074	358.790208

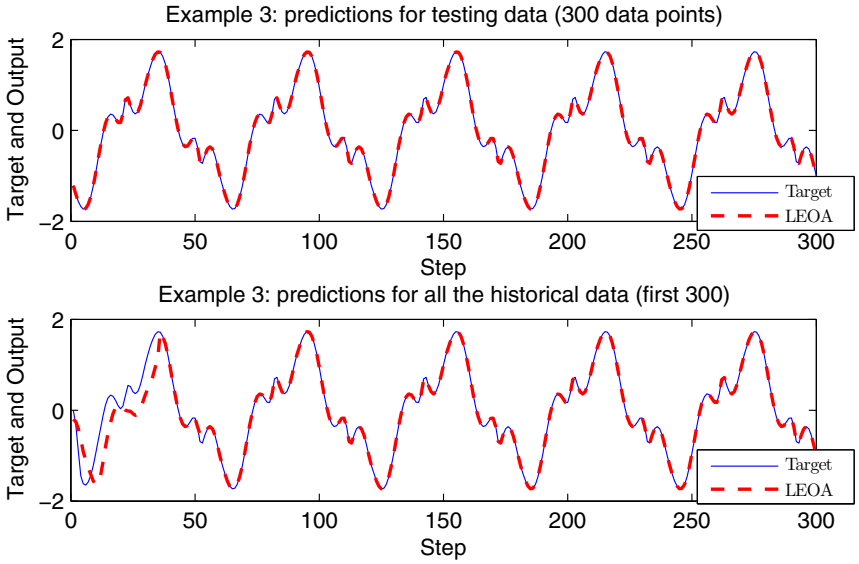


Figure 5: Example 3: prediction results for testing and all historical data.

erated from the following differential equation (43) with time delay.

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{1+x^{10}(t-\tau)} - bx(t), \tag{43}$$

in which $a = 0.2$, $b = 0.1$, $\tau \geq 17$. The same as most of previous researches we use $\tau = 17$. There are 3500 input-output pairs generated by the fourth-order Range-Kutta method with $x(0) = 1.2$. The first 3000 training samples are in the range of $t = 201$ to

$t = 3200$, and the remaining testing samples are produced from $t = 5001$ to $t = 5500$. The prediction model is shown below,

$$\hat{x}(t+85) = f(x(t), x(t-6), x(t-12), x(t-18)), \tag{44}$$

where $\hat{x}(t+85)$ is the estimated value of $x(t+85)$. In order to make comparisons, LEOA is compared with other exaisting EFSs approaches, DENFIS [9], eTS+[27], Simple_eTS+ [68], GENEFIS [37], PANFIS [36], eT2RFNN [42], GSETSK [14], SPLAFIS [69] and DeTS [51]. Performances of these algorithms are judged by NDEIs. Figure 6 portrays the error trace of LEOA while running online on the training data set. From Figure 6 we can see that absolute value of prediction error decreases from nearly 0.3 to approximately 0.1 after 1500 training steps. Further, the errors are varying approximately between -0.1 and 0.1 from 1501 to 3000 training steps. Table 6 summarizes the numerical results of all state-of-the-art algorithms and LEOA. These algorithms are compared by NDEIs, fuzzy rule numbers as well as execution time. The results of LEOA listed in Table 6 is obtained by chosen the control parameters as $\varepsilon_0 = 0.1$ and $N = 50$. Furthermore, in order to test the global optimality of the online learning algorithm LEOA, we use the final model learned from the whole data set to make estimations of all the 3500 historical data with neither fuzzy rule nor parameters updating. Numerical results reports that LEOA presents its characteristics of global optimality highlighted in this paper with $NDEI = 0.2423$.

As can be seen from Table 6, LEOA achieves the best performance judged by NDEI. However, LEOA applies more fuzzy rules being evolved into making predictions and longer execution time than many of the other algorithms. This phenomenon is determined by the designing viewpoint and structure of LEOA. LEOA is always trying to keep more historical information to assure the global optimality while achieving the high testing accuracy.

9. Conclusions

In this paper, we propose a new online learning algorithm referred to as LEOA for identifying eFS. LEOA is mainly featured by the following two main novel aspects.

Table 6: NDEIs of example 4

Method	model type	NDEIs	rule num.	parameter	run time(s)
LEOA	T-S	0.2480	42	546	144.781848
DENFIS [9]	T-S	0.278	58	886	–
eTS+[27]	T-S	0.392	10	130	–
Simple_eTS+[68]	T-S	0.375	15	150	–
GENEFIS(C)[37]	T-S	0.280	19	475	4.46
GENEFIS(B)[37]	T-S	0.339	9	225	3.02
PANFIS[36]	T-S	0.301	19	475	4.5
eT2RFNN[42]	T-S	0.32	3	108	–
GSETSK[14]	T-S	0.347	19	247	–
SPLAFIS[69]	T-S	0.279	30	390	–
DeTS[51]	T-S	0.440	3	39	–

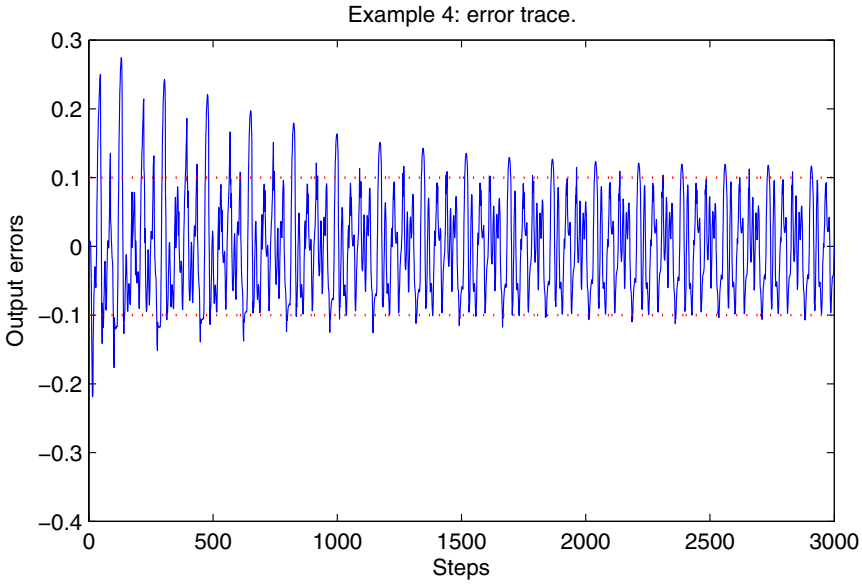


Figure 6: Example 4: error trace.

On one hand, antecedent evolving approaches (i.e. structure evolving strategies and antecedent parameters adaptation method) and consequent parameters updating methods (i.e. EWRLS) are derived together for the purpose of minimizing the local error functions. Exact calculation and proofs are given to verify this optimality and the ε-completeness property. On the other hand, LEOA achieves in not only local optimality but also significant global fitting accuracy. The feasibility and accuracy are validated by both artificial and real-world data sets across both system identification and time series prediction area. Although LEOA is usually more complex than many of other online recursive algorithms, it is still worthwhile in its optimality of consequent parameters and the ability for learning the global behavior of a data stream. For the future research, we will develop further methods for reducing the algorithm complexity of LEOA and feature selection method. Further, the extension of LEOA to the generalized T-S fuzzy systems with non axis-paralleled rules proposed by [36] (PANFIS) and [39] (Gen-Smart-EFS) will be considered.

Appendix A.

Appendix A.1. proof of theorem 4.1

Proof If *condition 4.1* holds, then $\sigma^2_{R_{t+1},j} \leq \frac{n(x_{k,j}-c_{R_{t+1},j})^2}{-2\log\{\frac{\varepsilon_0}{t}\}}$ is satisfied, which further implies $\exp\{-\sum_{j=1}^{n}\frac{(x_{k,j}-c_{R_{t+1},j})^2}{2\sigma^2_{R_{t+1},j}}\} \leqslant \frac{\varepsilon_0}{t}$. Because each $\frac{\gamma^i_{t+1}(x_k)}{(\sum_{l=1}^{R_{t+1}}\gamma^l_{t+1}(x_k))(\sum_{l=1}^{R_t}\gamma^l_{t+1}(x_k))}$ is bounded, so $\exists\ \varepsilon > 0$ such that $\sum_{k=1}^{t} A^i_{t,k}x^T_{ek} < \varepsilon$ and $\sum_{k=1}^{t} A^i_{t,k}y_k < \varepsilon$. Then, ψ^i could be estimated recursively by (15). Besides, it is easy to verify that ε-completeness is satisfied.

□

Appendix A.2. proof of theorem 4.2

Proof As there exists a finite number $N_1 > 0$ such that $|\frac{\gamma^i_{t+1}(x_k)}{(\sum_{l=1}^{R_t}\gamma^l_t(x_k))(\sum_{l=1,\ l\neq i^*}^{R_t}\gamma^l_{t+1}(x_k))}| < N_1$ for $i = 1,2,\ldots,R_t$, $i \neq i^*$, and $\sum_{p=t^*+1}^{k}\gamma^{i^*}_p(x_p) < \varepsilon_p$, so there exists an $\varepsilon > 0$ such that $\sum_{k=1}^{t} B^i_{t,k}x^T_{ek} < \varepsilon$ and $\sum_{k=1}^{t} B^i_{t,k}y_k < \varepsilon$.

□

Appendix A.3. proof of theorem 4.3

Proof As *condition 4.3* is satisfied, so $\gamma_t^{j_1}(x_k) \approx \gamma_t^{j_2}(x_k) \approx \gamma_t^{j_0}(x_k)$, thus we can use $\gamma_t^{j_1}(x_k)$ to replace $\gamma_t^{j_0}(x_k)$ in (20) and (21), then these two equations could be approximated by (A.1) and (A.2),

$$\overline{R^i}(t+1) = \overline{R^i}(t) + \sum_{k=1}^{t} C_{t,k}^{i,l_2} x e_k x e_k^T + \frac{\gamma_{t+1}^i(x_{t+1})}{\sum_{l=1}^{R_{t+1}} \gamma_{t+1}^l(x_{t+1})} x e_{t+1} x e_{t+1}^T, \tag{A.1}$$

$$f^i(t+1) = f^i(t) + \sum_{k=1}^{t} C_{t,k}^{i,l_2} x e_k y_k + \frac{\gamma_{t+1}^i(x_{t+1})}{\sum_{l=1}^{R_{t+1}} \gamma_{t+1}^l(x_{t+1})} x e_{t+1} y_{t+1}, \tag{A.2}$$

in which

$$C_{t,k}^{i,l_2} = \frac{\gamma_t^i(x_k)\gamma_t^{l_2}(x_k)}{[\sum_{l=1}^{R_t} \gamma_t^l(x_k) - \gamma_t^{l_2}(x_k)][\sum_{l=1}^{R_t} \gamma_t^l(x_k)]}. \tag{A.3}$$

Let

$$\widetilde{R_{l_2}^i}(t+1) = \overline{R^i}(t) + \sum_{k=1}^{t} C_{t,k}^{i,l_2} x e_k x e_k^T = \sum_{k=1}^{t} \{\theta_t^i(x_k) + C_{t,k}^{i,l_2}\} x e_k x e_k^T, \tag{A.4}$$

$$\widetilde{f_{l_2}^i}(t+1) = f^i(t) + \sum_{k=1}^{t} C_{t,k}^{i,l_2} x e_k y_k = \sum_{k=1}^{t} \{\theta_t^i(x_k) + C_{t,k}^{i,l_2}\} x e_k y_k, \tag{A.5}$$

thus we can get that (28), (29) and (30) hold. Further, use (A.6) and (A.7),

$$\overline{R^i}(t+1) = \widetilde{R_{l_2}^i}(t) + \frac{\gamma_{t+1}^i(x_{t+1})}{\sum_{l=1}^{R_{t+1}} \gamma_{t+1}^l(x_{t+1})} x e_{t+1} x e_{t+1}^T, \tag{A.6}$$

$$\overline{f^i}(t+1) = \widetilde{f_{l_2}^i}(t) + \frac{\gamma_{t+1}^i(x_{t+1})}{\sum_{l=1}^{R_{t+1}} \gamma_{t+1}^l(x_{t+1})} x e_{t+1} y_{t+1}, \tag{A.7}$$

it is easy to get that (27) holds.

□

Appendix A.4. deduction of formula (24)

Once l_1-th and l_2-th fuzzy rule are merged into the l_0-th as shown in (24), the sample size of the l_0-th cluster becomes $N_{l_1}+N_{l_2}$. Based on (25), the radius $\sigma_{l_0,j}$ could be estimated using (A.8),

$$\begin{aligned}(\sigma_{l_0,j})^2 &= \frac{\sum_{k=1}^{N_{l_1}+N_{l_2}}(x_{k,j}-c_{l_0,j})^2}{N_{l_1}+N_{l_2}-1} \\ &= \frac{\sum_{i_1=1}^{N_{l_1}}(x_{i_1,j}-c_{l_0,j})^2}{N_{l_1}+N_{l_2}-1}+\frac{\sum_{i_2=1}^{N_{l_1}}(x_{i_2,j}-c_{l_0,j})^2}{N_{l_1}+N_{l_2}-1},\end{aligned} \tag{A.8}$$

where $c_{l_0,j}$ is the merged cluster center shown in (23), $j=1,2,\ldots,n$. The numerator of the first term could be presented by (A.9),

$$\begin{aligned}\sum_{i_1=1}^{N_{l_1}}(x_{i_1,j}-c_{l_0,j})^2 &= \sum_{i_1=1}^{N_{l_1}}(x_{i_1,j}-c_{l_1,j}+c_{l_1,j}-c_{l_0,j})^2 \\ &= (N_{l_1}-1)(\sigma_{l_1,j})^2+N_{l_1}(c_{l_1,j}-c_{l_0,j})^2.\end{aligned} \tag{A.9}$$

Similar formula could be used to present $\sum_{i_2=1}^{N_{l_1}}(x_{i_2,j}-c_{l_0,j})^2$. Therefore, based on (A.8) and (A.9), (24) is easy to be obtained.

References

[1] R. Ghazali, A. Hussain, N. Nawi, B. Mohamad, Non-stationary and stationary prediction of financial time series using dynamic ridge polynomial neural network 72 (10) (2009) 2359–2367.

[2] L. Maciel, R. Ballini, F. Gomide, Evolving possibilistic fuzzy modeling for realized volatility forecasting with jumps, IEEE Transactions on Fuzzy Systems.

[3] A. Ghazikhani, R. Monsefi, H. S. Yazdi, Ensemble of online neural networks for non-stationary and imbalanced data streams, Neurocomputing 122 (2013) 535–544.

[4] Z. Hu, Y. Bodyanskiy, O. Tyshchenko, O. Boiko, Adaptive forecasting of non-stationary nonlinear time series based on the evolving weighted neuro-neo-fuzzy-anarx-model, arXiv preprint arXiv:1610.06486.

[5] J. Tomczak, A. Gonczarek, Decision rules extraction from data stream in the presence of changing context for diabetes treatment, Knowledge and Information Systems 34 (3) (2013) 521–546.

[6] D. Lowne, S. Roberts, R. Garnett, Sequential non-stationary dynamic classification with sparse feedback, Pattern Recognition 43 (3) (2010) 897–905.

[7] J. Gama, Knowledge discovery from data streams, CRC Press, 2010.

[8] M. Sayed-Mouchaweh and E. Lughofer, Learning in non-stationary environments: methods and applications, Springer, 2012.

[9] N. Kasabov, Q. Song, Denfis: dynamic evolving neural-fuzzy inference system and its application for time-series prediction, IEEE Transactions on fuzzy systems 10 (2) (2002) 144–154.

[10] 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.

[11] J. de Jesús Rubio, Sofmls: online self-organizing fuzzy modified least-squares network, IEEE Transactions on Fuzzy Systems 17 (6) (2009) 1296–1309.

[12] E. Lughofer, Flexfis: a robust incremental learning approach for evolving takagi-sugeno fuzzy models, IEEE Transactions on fuzzy systems 16 (6) (2008) 1393–1410.

[13] H. Rong, N. Sundararajan, G. Huang, P. Saratchandran, Sequential adaptive fuzzy inference system (safis) for nonlinear system identification and prediction, Fuzzy sets and systems 157 (9) (2006) 1260–1275.

[14] N. Nguyen, W. Zhou, C. Quek, Gsetsk: a generic self-evolving tsk fuzzy neural network with a novel hebbian-based rule reduction approach, Applied Soft Computing 35 (2015) 29–42.

[15] P. X. Liu, M. Q. H. Meng, Online data-driven fuzzy clustering with applications to real-time robotic tracking, IEEE Transactions on Fuzzy Systems 12 (4) (2004) 516–523.

[16] P. Hore, L. Hall, D. Goldgof, A fuzzy c means variant for clustering evolving data streams, in: Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on, IEEE, 2007, pp. 360–365.

[17] E. Lughofer, P. Angelov, Handling drifts and shifts in on-line data streams with evolving fuzzy systems, Applied Soft Computing 11 (2) (2011) 2057–2068.

[18] N. Kasabov, Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 31 (6) (2001) 902–9182.

[19] P. Angelov, R. Buswell, Identification of evolving fuzzy rule-based models, IEEE Transactions on Fuzzy Systems 10 (5) (2002) 667–677.

[20] F. Lin, C. Lin, P. Shen, Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive, IEEE transactions on fuzzy systems 9 (5) (2001) 751–759.

[21] C. Juang, C. Lin, An online self-constructing neural fuzzy inference network and its applications, IEEE Transactions on Fuzzy Systems 6 (1) (1998) 12–32.

[22] N. Kasabov, On-line learning, reasoning, rule extraction and aggregation in locally optimized evolving fuzzy neural networks, Neurocomputing 41 (1) (2001) 25–45.

[23] G. Leng, T. McGinnity, G. Prasad, An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network, Fuzzy sets and systems 150 (2) (2005) 211–243.

[24] C. Juang, Y. Tsao, A self-evolving interval type-2 fuzzy neural network with online structure and parameter learning, IEEE Transactions on Fuzzy Systems 16 (6) (2008) 1411–1424.

[25] G. Leng, G. Prasad, T. McGinnity, An on-line algorithm for creating self-organizing fuzzy neural networks, Neural Networks 17 (10) (2004) 1477–1493.

[26] W. Wang, J. Vrbanek, An evolving fuzzy predictor for industrial applications, IEEE Transactions on Fuzzy Systems 16 (6) (2008) 1439–1449.

[27] P. Angelov, Evolving takagi-sugeno fuzzy systems from streaming data (ets+), Evolving intelligent systems: Methodology and applications 12 (2010) 21.

[28] E. Lughofer, Flexible evolving fuzzy inference systems from data streams (FLEXFIS++), in: Learning in Non-Stationary Environments, Springer, 2012, pp. 205–245.

[29] A. Lemos, W. Caminhas, F. Gomide, Multivariable gaussian evolving fuzzy modeling system, IEEE Transactions on Fuzzy Systems 19 (1) (2011) 91–104.

[30] J. de Jesús Rubio and A. Bouchachia, MSAFIS: an evolving fuzzy inference system, Soft Computing (2015) 1–10.

[31] Y. Pan and Y. Liu and B. Xu and H. Yu, Hybrid feedback feedforward: an efficient design of adaptive neural network control, Neural Networks 76 (2016) 122–134.

[32] J. de Jesús Rubio and J.H. Pérez-Cruz, Evolving intelligent system for the modelling of nonlinear systems with dead-zone input, Applied Soft Computing 14 (2014) 289–304.

[33] A. Bouchachia and C. Vanaret, GT2FC: An online growing interval type-2 self-learning fuzzy classifier, IEEE Transactions on Fuzzy Systems 22 (4) (2014) 999–1018.

[34] L. Maciel, F. Gomide, R. Ballini, Enhanced evolving participatory learning fuzzy modeling: an application for asset returns volatility forecasting, Evolving Systems 5 (2) (2014) 75–88.

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

[36] M. Pratama, S. Anavatti, P. Angelov, E. Lughofer, Panfis: a novel incremental learning machine, IEEE Transactions on Neural Networks and Learning Systems 25 (1) (2014) 55–68.

[37] M. Pratama and S. G Anavatti and E. Lughofer, GENEFIS: toward an effective localist network, IEEE Transactions on Fuzzy Systems 22 (3) (2014) 547–562.

[38] Y. Xu and K. W. Wong and C.S. Leung, Generalized RLS approach to the training of neural networks, IEEE Transactions on Neural Networks 17 (1) (2006) 19–34.

[39] E. Lughofer, C. Cernuda, S. Kindermann, M. Pratama, Generalized smart evolving fuzzy systems, Evolving Systems 6 (4) (2015) 269–292.

[40] E. Lughofer and M. Pratama, On-line Active Learning in Data Stream Regression using Uncertainty Sampling based on Evolving Generalized Fuzzy Models, IEEE Transactions on Fuzzy Systems PP (2017) 1. doi:10.1109/TFUZZ.2017.2654504.

[41] M. Pratama and J. Lu and G. Zhang, Evolving type-2 fuzzy classifier, IEEE Transactions on Fuzzy Systems 24 (3) (2016) 574–589.

[42] M. Pratama and J. Lu and E. Lughofer and G. Zhang and M. J. Er, Incremental Learning of Concept Drift Using Evolving Type-2 Recurrent Fuzzy Neural Network, IEEE Transactions on Fuzzy Systems PP (2016) 1. doi:10.1109/TFUZZ.2016.2599855.

[43] A. Silva, W. Caminhas, A. Lemos, F. Gomide, A fast learning algorithm for evolving neo-fuzzy neuron, Applied Soft Computing 14 (2014) 194–209.

[44] J. Qiao, Z. Zhang, Y. Bo, An online self-adaptive modular neural network for time-varying systems, Neurocomputing 125 (2014) 7–16.

[45] S. Tung, C. Quek, C. Guan, et2fis: An evolving type-2 neural fuzzy inference system, Information Sciences 220 (2013) 124–148.

[46] Y. Lin, S. Liao, J. Chang, C. Lin, Simplified interval type-2 fuzzy neural networks, IEEE transactions on neural networks and learning systems 25 (5) (2014) 959–969.

[47] Y. Lin, J. Chang, C. Lin, A tsk-type-based self-evolving compensatory interval type-2 fuzzy neural network (tscit2fnn) and its applications, IEEE Transactions on Industrial Electronics 61 (1) (2014) 447–459.

[48] D. Ge, X. Zeng, Modified evolving participatory learning algorithms for takagi-sugeno fuzzy system modelling from streaming data, in: Advances in Computational Intelligence Systems, Springer, 2017, pp. 145–163.

[49] E. Lughofer, Evolving fuzzy systems-methodologies, advanced concepts and applications, Vol. 53, Springer, 2011.

[50] E. Lughofer, Evolving Fuzzy Systems—Fundamentals, Reliability, Interpretability, Useability, Applications, in: P. Angelov (Ed.), Handbook of Computational Intelligence, World Scientific, New York, 2016, pp. 67–135.

[51] R. D. Baruah and P. Angelov, DEC: Dynamically evolving clustering and its application to structure identification of evolving fuzzy models, IEEE transactions on cybernetics 44 (9) (2014) 1619–1631.

[52] R. Hyde and P. Angelov and AR MacKenzie, Fully online clustering of evolving data streams into arbitrarily shaped clusters, Information Sciences 382 (2017) 96–114.

[53] A. Shaker, Ammar and E. Lughofer, Edwin, Self-adaptive and local strategies for a smooth treatment of drifts in data streams, Evolving Systems 5 (4) (2014) 239–257.

[54] E. Lughofer, EFS Approaches for Regression and Classification, Evolving Fuzzy Systems–Methodologies, Advanced Concepts and Applications (2011) 93–164.

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

[56] P. Angelov, Evolving Takagi-Sugeno Fuzzy Systems from Streaming Data (eTS+), Evolving Intelligent Systems: Methodology and Applications (2010) 21–50.

[57] H. Hamrawi and S. Coupland and R. John, A novel alpha-cut representation for type-2 fuzzy sets, in: Fuzzy Systems (FUZZ), 2010 IEEE International Conference on, IEEE, 2010, pp. 1–8.

[58] J. de Jesús Rubio, Least square neural network model of the crude oil blending process, Neural Networks 78 (2016) 88–96.

[59] Q. Liu and J. Yin and V.CM Leung and J.H. Zhai and Z. Cai and J. Lin, Applying a new localized generalization error model to design neural networks trained with extreme learning machine, Neural Computing and Applications 27 (1) (2016) 59–66.

[60] J. de Jesús Rubio, Adaptive least square control in discrete time of robotic arms, Soft Computing 19 (12) (2015) 3665–3676.

[61] J. Viola and L. Angel, Identification, control and robustness analysis of a robotic system using fractional control, IEEE Latin America Transactions 13 (5) (2015) 1294–1302.

[62] P. Angelov, D. Filev, Simpl_ets: a simplified method for learning evolving takagi-sugeno fuzzy models, in: The 14th IEEE International Conference on Fuzzy Systems, 2005. FUZZ'05., IEEE, 2005, pp. 1068–1073.

[63] H. Du and N. Zhang, colorblueApplication of evolving Takagi–Sugeno fuzzy model to nonlinear system identification, Applied soft computing 8 (1) (2008) 676–686.

[64] G. Leng and X.J. Zeng and J.A Keane, A hybrid learning algorithm with a similarity-based pruning strategy for self-adaptive neuro-fuzzy systems, Applied Soft Computing 9 (4) (2009) 1354–1366.

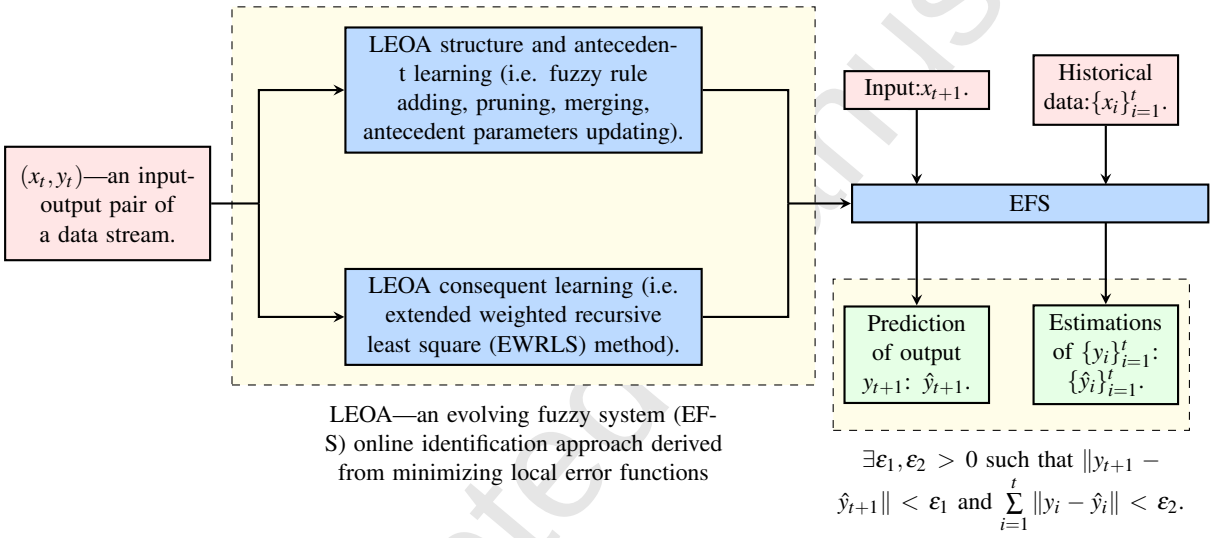
[65] K. Subramanian, S. Suresh, A meta-cognitive sequential learning algorithm for neuro-fuzzy inference system, Applied soft computing 12 (11) (2012) 3603–3614.

[66] L. Lin and F. Guo and X. Xie and B. Luo, Novel adaptive hybrid rule network based on TS fuzzy rules using an improved quantum-behaved particle swarm optimization, Neurocomputing 149 (2015) 1003–1013.

[67] C. Li and J. Zhou and L. Chang and Z. Huang, T-S fuzzy model identification based on a novel hyper-plane-shaped membership function, IEEE Transactions on Fuzzy Systems PP (2016) 1. doi:10.1109/TFUZZ.2016.2598850.

[68] P. Angelov, Fuzzily connected multimodel systems evolving autonomously from data streams, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 41 (4) (2011) 898–910.

[69] R.J Oentaryo and M. J. Er and S. Linn and X. Li, Online probabilistic learning for fuzzy inference system, Expert Systems with Applications 41 (11) (2014) 5082–5096.



LEOA—an evolving fuzzy system (EFS) online identification approach derived from minimizing local error functions

$\exists \varepsilon_1, \varepsilon_2 > 0$ such that $\|y_{t+1} - \hat{y}_{t+1}\| < \varepsilon_1$ and $\sum_{i=1}^{t} \|y_i - \hat{y}_i\| < \varepsilon_2$.