Proceedings of the International Symposium on Evolving Intelligent Systems, Plamen Angelov, Dimitar Filev, and Nikola Kasabov (Eds.), at the AISB 2010 convention, 29 March – 1 April 2010, De Montfort University, Leicester, UK

An Online Predictor Model as Adaptive Linear model and Evolving Takagi-Sugeno Model

Ahmad Kalhor, Babak N. Araabi and Caro Lucas[1]

Abstract—**In this paper, we suggest an online predictor model to follow the variations of a process with significant uncertainties. This model is defined as an adaptive linear model initially; however, it starts to evolve into a TS model and continues the process while the modeling error is higher than an adaptive threshold. The model habitually returns to an adaptive linear model when the modeling error becomes lower than the adaptive threshold. Evolving from a linear model to a nonlinear one and then returning to a linear model is performed through specially designed split and merge procedures. Finally, a real-world case study is conducted: prediction of monthly number of sunspots demonstrates the capabilities of the proposed approach.**

I. INTRODUCTION

Modeling processes with time varying behaviors, where stationary and fixed models are not suitable, is a demanding task. Online modeling is a fitting solution, where structure and parameters of model can be changed adaptively with respect to the real-time process operation.

Due to their interpretable and flexible structures, Takagi-Sugeno (TS) fuzzy models are suitable tools for online modeling, [1] and [2]: To adapt a TS fuzzy model for time-varying behaviors of a process, fuzzy rules can be easily increased or decreased.

An adaptive online learning algorithm named Dynamic Evolving Neural-Fuzzy Inference System (DENFIS) is proposed in [3]. In [4], a hierarchical on-line self-organizing learning algorithm is proposed to identify a TSK fuzzy model. An online identification approach of the TS model is introduced in [5], where a subtractive clustering and a potential concept are used to define the antecedent parts of the rules. In [6], an online identification of fuzzy neural networks is proposed, which is then applied to NARMAX time series prediction in two different feed-forward and recurrent models. Another online identification approach for the TS model is introduced in [7] where the input space of system is partitioned into a lattice, and local linear models are validated to keep the structure intact. In [8], an approach to online identification of fuzzy rules of an extended TS fuzzy model is proposed. Fuzzy rule structures are formed through an incremental clustering approach and parameters are updated, recursively, through a non-iterative learning process. An online growing and pruning strategy is proposed in [9] based on contribution of fuzzy rules to the modeling accuracy. In [10], a modeling of switching systems is proposed in which parameters and the order of systems can change when the system switches. In this approach, the local functioning modes were tuned through using an online clustering method. In [11], a fast online neuro-fuzzy modeling is proposed. In this paper, an online fuzzy clustering is introduced which is independent of common spatial features.

In many online modeling approaches, such as the above mentioned ones, the initial defined model evolves gradually to follow the variations of a process. This strategy often makes favorable results in modeling the processes with no significant uncertainties. Insufficient data points, abrupt time-varying behaviors, undetermined effective inputs and un-modeled dynamics of a process accounts for significant uncertainties in an online modeling procedure. In these cases, the model diverges from the process, since the model inertly changes structures and lacks the necessary agility to follow the resulting erratic variations in the process.

Here, we take advantage of linear models as an easy-to-adapt structure to follow the erratic variations occurring in a nonlinear process, whereas the required complexity is provided by allowing each linear model to evolve transiently to a nonlinear one. The proposed Adaptive Habitually Linear and Transiently Nonlinear Model (AHLTNM) can follow fast and significant structural variations in the processes with significant uncertainties. The resulting model is linear until the difference between the model and the process increases beyond an adaptive bound. In this situation, the model evolves into a TS nonlinear model to localize the effect of the errors. The complexity of the nonlinear model increases gradually by adding more rules through splitting existing ones. Splitting is maintained, until the output error complies with the adaptive bound. While the output error is mitigated, the AHLTNM starts to reduce the complexity by merging the existing rules. The AHLTNM, thus, tends to return to an adaptive linear model. As a result, the merging of rules continues as long as error bounds are satisfied. The process of evolving from an adaptive linear model to an adaptive TS nonlinear model and then returning to another linear model is not necessarily a monotonic one. That is, the split and merge processes may switch a few

The authors are with the Control and Intelligent Processing Center of Excellence, School of Electrical and Computer Engineering, University of Tehran, P.O. Box 14395/515, Tehran 14395, Iran (emails: akalhor@ut.ac.ir, lucas@ipm.ir, araabi@ut.ac.ir)

Caro Lucas and Babak N. Araabi are also with the School of Cognitive Sciences, Institute for Studies in Theoretical Physics and Mathematics, Tehran, Iran.

times.

The rest of this paper is organized as follows: Section II describes the AHLTNM along with its learning algorithm. Section III the proposed HLTNM is used to model a real-world process: prediction of monthly sunspots number. Also, the performance of the HLTNM is compared with two other online modeling approaches. The paper is concluded in Section IV.

II. The Structure of AHLTNM and its Learning Procedures

A. The Structure of the model in AHLTNM

Each transiently nonlinear model in AHLTNM is described as a TS model:

$$if \ \ \mathbf{x} \text{ is } A_j \text{ then } y^j = \mathbf{z}^T \mathbf{w}_j \quad j = 1,2,\ldots,M \qquad (1)$$

where $\mathbf{x} \in \mathbf{R}^n$ is the input vector and $\mathbf{z}^T = [1 \ \mathbf{x}^T]$, y^j is the local output of jth LLM, and $\mathbf{w}_j \in \mathbf{R}^{n+1}$ is the linear parameters vector of jth LLM (more generally, linear parameters of jth rule). The validity function of the jth rule is A_j, which is a fuzzy subset of the input space $\mathbf{R}^n$. The scalar output of model is computed as follows:

$$\hat{y} = \sum_{j=1}^{M} y^j \phi_j(\mathbf{x}), \quad \phi_j(\mathbf{x}) = \frac{A_j(\mathbf{x})}{\sum_{h=1}^{M} A_j(\mathbf{x})} \qquad (2)$$

where $\phi_j(\mathbf{x})$ denotes the membership function of normalized A_j. In AHLTNM, A_j is defined as the sum of N_j standard Gaussian functions:

$$A_j(\mathbf{x}) = \sum_{i=1}^{N_j} A_{ji}(\mathbf{x})$$
$$A_{ji}(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{c}_{ji})^T \mathbf{S}_{ji}^{-1}(\mathbf{x}-\mathbf{c}_{ji})\right) \qquad (3)$$

where $\mathbf{c}_{ji} \in \mathbf{R}^n$ and $\mathbf{S}_{ji} \in \mathbf{R}^{n \times n}$ denote the mean and the diagonal covariance matrix of Gaussian function A_{ji}, respectively.

A Typical contour of A_{ji}, as a standard Gaussian function, is a hyper-ellipse; therefore, A_{ji} can be defined by the center ($\mathbf{c}_{ji}$) and the diameter vector ($\mathbf{L}_{ji}$) of a contour

$$A_{ji}(\mathbf{x}) = \exp\left(-\sum_{k=1}^{n}\left(\gamma(\mathbf{x}(k)-\mathbf{c}_{ji}(k))/\mathbf{L}_{ji}(k)\right)^2\right) \qquad (4)$$

It is easy to see that $(\gamma^{-1}\mathbf{L}_{ji}(k))^2$ equals kth diagonal element of diagonal covariance matrix $\mathbf{S}_{ji}$, where γ^{-1} is an interpolation coefficient for membership functions, chosen typically 1/3 to make a reasonable interpolation [12]. Now, each ($\mathbf{L}_{ji}$, $\mathbf{c}_{ji}$) pair defines a standard hyper-rectangle b_{ji}, centered at $\mathbf{c}_{ji}$ as shown in Fig. 1.

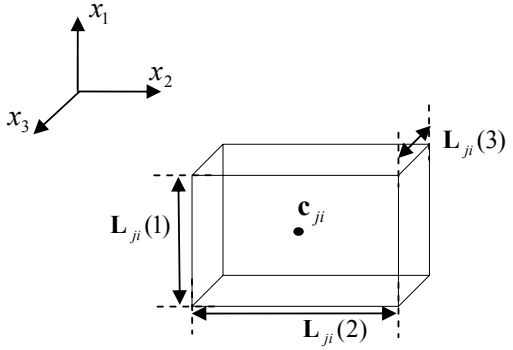

Fig. 1. A three-dimensional scheme for a standard hyper rectangle b_{ji} with center $\mathbf{c}_{ji}$ and length $\mathbf{L}_{ji}$ assigned to A_{ij}

A cluster $\mathbf{C}_j$ is defined as the union of N_j standard hyper-rectangles $b_{ji}; i = 1,2,\ldots,N_j$. As we will see in Section B, these b_{ji} s are disjoint subsets of $\mathbf{R}^n$ which make a single connected cluster, $\mathbf{C}_j$. Each crisp cluster $\mathbf{C}_j$, is associated with a membership function $\phi_j(.)$. Note that $\phi_j(.)$ s are not disjoint and each one constitutes a fuzzy partition on $\mathbf{R}^n$. Fig. 2 shows a transiently nonlinear model in AHLTNM as a TS-type model.

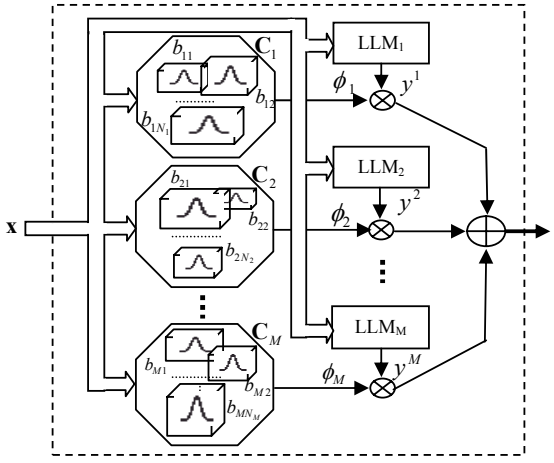

Fig. 2. A block diagram of the transiently nonlinear structure used in AHLTNM. The overall output of model is a weighted sum of local outputs of LLMs, weighted by validity function values.

B. An overview of the AHLTNM learning

Before going on into the details of our learning algorithm for AHLTNM, at first, we give an overview of the proposed algorithm. Assume, at one stage of our online

modeling, the AHLTNM consists of *M* rules. The newly observed input-output data is $(\mathbf{x}_t, y_t)$. We also consider the $y_{t-\tau}$, where $\tau \geq 1$ is a time delay which the output of system becomes observable after. Since, in the online prediction procedure, the observed data are obtained one by one, it is better to update only the part of the model that is exposed to the current data point directly. Hence, it is assumed that the *l*th rule, whose validity function in $\mathbf{x}_t$ has the biggest value, is updated and becomes the candidate for more changes.

$$l = \arg\max_j (A_j(\mathbf{x}_t)),\ j \in \{1,2,\ldots,M\} \quad (5)$$

The main parts of the online learning algorithm are:

1- Compute the initial modeling error ($\tilde{e}_t$) as the difference between the output of model, $\hat{y}_t$, and the actual observed output, y_t.

2- Use $\tilde{e}_t$ to update the linear parameters of the LLM corresponding to the *l*th rule with maximum validation function, $A_l(\mathbf{x}_t)$.

3- Re-compute the modeling error (e_t) after updating the linear parameters.

4- Make an adaptive threshold, $E_t = \eta \,|\, y_t - y_{t-\tau} \,|$, where $\eta > 0$ is introduced to characterize the uncertainty of the process and is adjusted by a heuristic search method.

5- If $|e_t| \geq E_t$ and $M < M_{\max}$, run the split procedure, where $M_{\max}$ denotes the maximum number of rules allowed by the designer. Here, the *l*th rule is split into two rules.

6- If $|e_t| < E_t$ and $M > 1$, run the merge procedure. Here, the *l*th rule is merged with the rule nearest to it, and a combined rule is formed.

The split and merge procedures are aimed to change the structure of the model. The process —and as a result, the model— is assumed to be habitually linear ($M = 1$). Adaptation of the linear model parameters provides some degree of flexibility. The model evolves by splitting into a nonlinear one ($M > 1$) only if the local error cannot be mitigated by parameter adaptation. This is a structural change. However, the nonlinear structure is a transient one which facilitates copies with nonlinear transition in the process. The merging procedure is adopted as a conservative operation that tends to bring the structure of the model back into the adaptive linear structure. Thus, the model tends to simplify the structure and eventually shape back into an adaptive linear model.

C. Details of the AHLTNM learning algorithm

In this section, we provide mathematical and computational details of the procedures introduced in Section B at different steps of the AHLTNM learning algorithm.

C.1. Updating the linear parameters

The parameters of the *l*th rule are updated through the Recursive Least Squares (RLS) algorithm applied to weighted input-output data, which are defined as follows:

$$\begin{aligned} \mathbf{z}_{tl} &= \mathbf{z}_t (\phi_l(\mathbf{x}_t))^{0.5} \\ y_{tl} &= y_t (\phi_l(\mathbf{x}_t))^{0.5} \end{aligned} \quad (6)$$

To update $\mathbf{w}_t$, last local linear parameters vector $\mathbf{w}_l^{t-1}$ is used as follows:

$$\begin{aligned} \mathbf{w}_l^t &= \mathbf{w}_l^{t-1} - \mathbf{P}_l^{t-1}\mathbf{z}_{tl} e_{tl} \\ e_{tl} &= \tilde{e}_t \left(\phi_l(\mathbf{x}_t)\right)^{0.5},\ \tilde{e}_t = y_t - \hat{y}_t \\ \mathbf{P}_l^t &= \mathbf{P}_l^{t-1} - \frac{\mathbf{P}_l^{t-1}\mathbf{z}_{tl}\mathbf{z}_{tl}^T\mathbf{P}_l^{t-1}}{1 + \mathbf{z}_{tl}^T\mathbf{P}_l^{t-1}\mathbf{z}_{tl}} \end{aligned} \quad (7)$$

C.2. Split Procedure

There are two different cases for split operation on *l*th rule:

a) The cluster $\mathbf{C}_l$ of lth rule consists of only one hyper-rectangular block ($N_l = 1, \mathbf{C}_l = \{b_{l1}\}$)

In this case, the $\mathbf{C}_l$ is halved into two clusters through orthogonal-to-axis division of b_{l1}, where both new clusters consist of equal hyper-rectangular blocks. As a result, there are *n* different pairs of clusters corresponding with *n* input dimensions, and n orthogonal divisions. In AHTNM, among *n* possible splits, we choose the one that better localizes $\mathbf{x}_t$ and its corresponding error. Hence, among all possible divisions ($i = 1,2,\ldots,n$) the division whose one of two resulting hyper-rectangles is closest to $\mathbf{x}_t$ is selected:

$$\begin{aligned} i^* &= \arg\min_i (D_i^l) \\ D_i^l &= \min(\|\mathbf{x}_t - \mathbf{c}_{r1}^i\|, \|\mathbf{x}_t - \mathbf{c}_{s1}^i\|), \quad i = 1,2,\ldots,n \end{aligned} \quad (8)$$

Where $\mathbf{c}_{r1}^i$ and $\mathbf{c}_{s1}^i$ are the centers of the new blocks in *i*th resulting pair. As a result, new clusters $\mathbf{C}_r = \{b_{r1}\}$ and $\mathbf{C}_s = \{b_{s1}\}$, defined by equation (8), take the place of $\mathbf{C}_l$. Based on orthogonal to axis i^* division, the parameters of b_{r1} and b_{s1} are determined as follows:

$$\begin{aligned} &\mathbf{L}_{r1}(i) = \mathbf{L}_{s1}(i) = \mathbf{L}_{l1}(i) \quad i \neq i^* \\ &\mathbf{L}_{r1}(i^*) = \mathbf{L}_{s1}(i^*) = \mathbf{L}_{l1}(i^*)/2 \\ &\mathbf{c}_{r1}(i) = \mathbf{c}_{s1}(i) = \mathbf{c}_{l1}(i) \quad i \neq i^* \\ &\mathbf{c}_{r1}(i^*) = \mathbf{c}_{l1}(i^*) + \mathbf{L}_{l1}(i^*)/4 \\ &\mathbf{c}_{s1}(i^*) = \mathbf{c}_{l1}(i^*) - \mathbf{L}_{l1}(i^*)/4 \end{aligned} \quad (9)$$

This split operation is shown in Fig. 3. As it is illustrated,

a cluster including one constructor block divides at all possible axis-orthogonal directions and a pair of hyper-rectangles, which is nearest to $\mathbf{x}_t$ is selected.

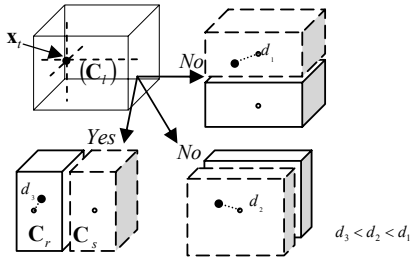

Fig. 3. A scheme of split operation while there is a constructor block in the selected cluster. A pair of clusters is selected, which are closest to $\mathbf{x}_t$.

b) The cluster $\mathbf{C}_l$ of the lth rule consists of more than one constructor block $(N_l > 1, \mathbf{C}_l = \{b_{lj}\}_{j=1}^{N_l})$

In this case, one of the constructor blocks of $\mathbf{C}_l$, which includes $\mathbf{x}_t$, is removed from $\mathbf{C}_l$ and is considered as a new cluster $\mathbf{C}_r = \{b_{r1}\}$ with one block. The rest of blocks in $\mathbf{C}_l$ are considered as a new cluster, $\mathbf{C}_s$. Corresponding with the two newly created $\mathbf{C}_r$ and $\mathbf{C}_s$, two new rules (r, s) are defined and take the place of rule (*l*). The linear parameters of new rules ($\mathbf{w}_r^t$ and $\mathbf{w}_s^t$) are initialized equal to $\mathbf{w}_l^t$ as linear parameters of former rule (*l*). To define $\mathbf{P}_r^t$ and $\mathbf{P}_s^t$ as data matrices of new rules, we can initialize them to be equal to $\mathbf{P}_l^t$ or make a covariance resetting:

$$\mathbf{P}_r^t = \mathbf{P}_s^t = \lambda \mathbf{I}_{(n+1)\times(n+1)} \tag{10}$$

where $\mathbf{I}$ denotes the unit matrix and λ is a large positive constant which is used as the resetting factor. By keeping $\mathbf{P}_j^t; j = r, s$ equal to $\mathbf{P}_l^t$, one forces a slow adaptation of parameters; while covariance resetting will allow for fast tracking of rapid changes.

C.3. Merge procedure

The Merging operation has two main stages: at the first stage, the *l*th rule is inter-merged with the nearest rule, and then the constructor blocks of this new combined rule are intra-merged together.

a) Inter-merge

In an inter merge, the *l*th rule is merged with the nearest rule, which is determined as follows:

$$k = \arg\min_j (\|\bar{\mathbf{c}}_l - \bar{\mathbf{c}}_j\|),\ j \in \{1,2,\ldots,M\} \,\&\, j \neq l$$
$$\bar{\mathbf{c}}_w = \frac{\sum_{x=1}^{N_w} V_{wx}\mathbf{c}_{wx}}{\sum_{x=1}^{N_w} V_{wx}},\ V_{wx} = \prod_{p=1}^{n} \mathbf{L}_{wx}(p), w \in \{l, j\} \tag{11}$$

where $\bar{\mathbf{c}}_w$ denotes the center of mass of the cluster $\mathbf{C}_w$ and V_{wx} denotes the volume of block $b_{wx} \in \mathbf{C}_w$. Through the inter-merge procedure, the new rule *r* takes the place of rules *l* and *k*, and its cluster, the new cluster $\mathbf{C}_r$, is defined as the union of $\mathbf{C}_l$ and $\mathbf{C}_k$. The linear parameters of rule *r*, $\mathbf{w}_r$, is determined as a simple interpolation of its building blocks:

$$\mathbf{w}_r = (V_l \mathbf{w}_l + V_k \mathbf{w}_k)/(V_l + V_k)$$
$$V_l = \sum_{x=1}^{N_l} V_{lx}, \quad V_k = \sum_{x=1}^{N_k} V_{kx} \tag{12}$$

One of matrices $\mathbf{P}_l$ and $\mathbf{P}_k$, which has the largest singular value, is chosen as the new data matrix $(\mathbf{P}_r)$ to retain the larger rate of adaptation for the new rule. The merge procedure is shown in Fig. 4 where two clusters indicated by different colors are merged together and a new cluster is created.

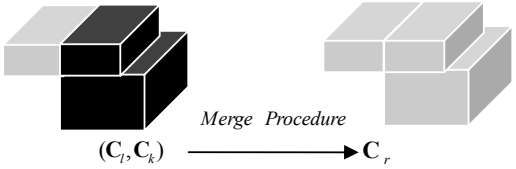

Fig. 4. A scheme of inter-merge operation where two initial clusters with different colors are merged and a new cluster including their constructor blocks is created.

b) Intra-merge

To reduce the complexity of the model and provide a stable learning procedure, it is desirable to decrease the number of internal blocks in each cluster. To this end, every two neighbor hyper-rectangular blocks, which have a common facet, are merged together and a new larger block is made. This type of merging is a simplified version of a former one introduced in [13]. Fig. 5 illustrates this type of merging operation. As it is shown, three constructor blocks are intra-merged together in two steps and a single constructor block is made.

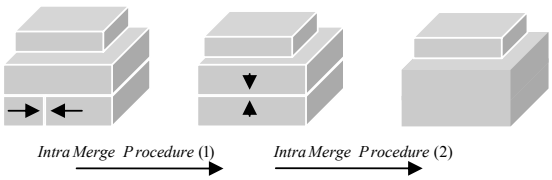

Fig. 5. A simple example that shows intra merging between constructor blocks of a cluster. At each stage, two constructor blocks are merged and the number of constructor blocks is reduced.

C.4. Adaptive threshold

To fulfill the main idea of AHLTNM, adaptive threshold E_t, should be determined appropriately. If E_t is considered too high, no split operation is done and the model always remains an adaptive linear model. In contrast, if E_t is considered too low, the split operations are done continuously and an unfeasible model with a lot of rules is

the result. To prevent this infeasibility of model, $M_{\max}$ is defined as the maximum number of allowed rules in AHLTNM. With respect to $M_{\max}$, E_t is determined as follows:

$$E_t = E_{t-1} + e_0\left(1 - \mathrm{sgn}(M_{\max} - M)\right) \qquad M \le M_{\max} \qquad (13)$$

where e_0 is computed as $\frac{1}{10}$ of the mean value of absolute errors during $N_0 = 20n$ initial observed data points. As it can be seen in (13), when $M = M_{\max}$, E_t increases gradually with a rate of e_0; but when $M < M_{\max}$, E_t does not change. We opt $M_{\max} = 10$ to be used as the default value.

D. The AHLTNM learning Algorithm

Before performing the AHLTNM learning algorithm, an initial hyper-rectangle must be determined and e_0 must be computed through N_0 initial incoming data points. The first hyper-rectangle is determined as large as to include all initial data points as follows:

$$\mathbf{c}_{11} = \frac{\sum_{i=1}^{N_0} \mathbf{x}_t}{N_0},\ \mathbf{L}_{11}(k) = u^k_{\max} - u^k_{\min}, k = 1,2,\ldots,n \qquad (14)$$

$$u^k_{\max} = \max(\{\mathbf{x}_t(k)\}_{t=1}^{N_0}), u^k_{\min} = \min(\{\mathbf{x}_t(k)\}_{t=1}^{N_0})$$

It is also assumed that the output of the model can be observed after $\tau \ge 1$ due to the input-output delay of the system or considering a multi-step-ahead prediction strategy. The overall algorithm is stated in Table I.

Table I
The overall algorithm of AHLTNM

1-Define parameters of first rule at $t=t_0$:

$$\mathbf{w}_1^{t_0} = \mathbf{z}_{(n+1)\times 1}, \mathbf{P}_1^{t_0} = \lambda \mathbf{I}_{(n+1)\times(n+1)}$$

where $\mathbf{z}$ and $\mathbf{I}$ denote a zero vector and an unit matrix respectively and λ is a large positive constant which is used as the resetting factor. In this paper, $\lambda = 100$ is considered in case studies.

2- For initial $N_0 = 20n$ intake data points, $t := 1$ up to $t = N_0$ perform following tasks

2-1. Estimate the outputs of linear model and absolute errors.

2-2. Update the parameters through the RLS technique.

2-3. Compute e_0 as $\frac{1}{10}$ of the mean value of absolute errors.

3-Define the center and lengths of the first hyper-rectangle using the considered initial N_0 data points through (14); also, initialize adaptive threshold as $E_t = e_0$.

4-Let *t*:=*t*+1 and compute the output of the model through (2).

5- Update the parameters of the rule (*l*th one) with highest validation function at $\mathbf{x}_{t'}$ by using equations (5)-(7).

6- Update the adaptive error threshold, E_t, through (13).

7-If $|e_t| \ge E_t$ and the number of rules is smaller than $M_{\max}$ then run split operation for *l*th rule by using equations(8)-(10);

8- If $|e_t| < E_t$ and there are at least two rules, run merge operation for *l*th rule by using equations (11)-(12).

9- Go to step 4.

III. A CASE STUDY

Here, prediction of the monthly smoothed sunspot number time series (as a well studied index of space weather phenomena) is considered. The monthly smoothed sunspot number time series is courtesy of the SIDC (World Data Center for sunspot Index) [14]. In order to compare the prediction results with some of previous studies, all constraints of prediction held before, are considered for AHLTNM too. The training dataset includes samples from 1900 until January 1999 and test dataset includes the part of 23rd solar cycle from January 1999 to May 2001 as explained in [15]. The normalized MSE index (NMSE) is used as the ratio of the sum of square output errors over the sum of square errors between outputs and the averaged values. One-step-ahead prediction of time series ($\tau = 1$) is considered, where y_t is predicted based on $\mathbf{x}_t$ is defined as follows ([15]):

$$\mathbf{x}_t = [y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4}, y_{t-5}]^{\mathrm{T}} \qquad (15)$$

The AHLTNM learning algorithm is applied to this considered prediction problem. After $N_0 = 100$ initial obtained data points, $e_0 = 0.0731$ is computed and then an adaptive threshold is initialized and updated through (13). Diagrams of the adaptive threshold and the absolute predicted error, and also, number of rules are shown in Fig. 6.

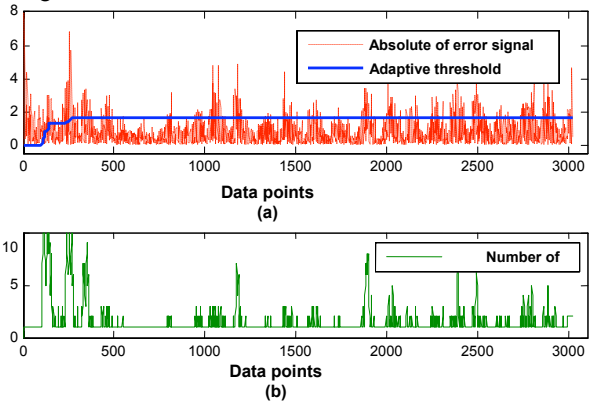

Fig. 6. Plots of adaptive threshold and absolute error, (a), as well as the number of rules, (b), versus the number of obtained data points

As appears in Fig. 6(a), while an absolute error sample (dashed line) is higher than the adaptive threshold (bold

line) a transient nonlinear model is started and is evolved by split operations. As appears in Fig. 6(b), the maximum number of rules in the evolved model is limited to $M_{\max} = 10$. The online model habitually tends to decrease the number of LLMs and to be a linear model. Fig. 7 shows the original and predicted monthly sunspot number time series for the test dataset.

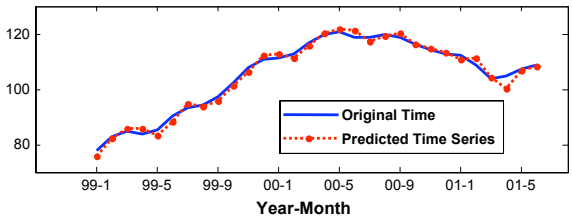

Fig. 7. A graph of the original time series (solid line) and predicted sunspot number time series (dashed-dot line) versus time (year-month)

Table II summarizes the prediction results of AHLTNM learning algorithm and two other online modeling approaches. In all approaches, changing structures for models and parameters is allowed, even through test data. As it can be seen, the AHLTNM has lower NMSE, lower number of rules and a lower execution time compared to the models using from other approaches.

Table II
Global performance measures (NMSE) for recursive prediction of the monthly smoothed sunspot number on January 1999- May 2001

Methods	NMSE	N. fuzzy rules	Execution time
DENFIS [3]	0.0230	11	12 sec
ETS [5]	0.0303	7	15 sec
AHLTNM	**0.018**	**2**	**7sec**

V. CONCLUSION

In this paper, we introduced a predictor model, which can follow the variations of processes with significant uncertainties. The proposed model was a habitually adaptive linear model which can evolve into a TS nonlinear model upon necessity. The necessity basically comes from higher temporal modeling error. Starting with an adaptive linear model helped us to keep the model as simple as possible. A simple model with a few parameters to adapt means more agility in following the variations of the process.

When the linear adaptive model was unable to follow process properly, the model was allowed to evolve into a nonlinear TS model, gradually. The nonlinear model was not a permanent one, however. The proposed AHLTNM learning algorithm is designed with a tendency to return to the linear model. Evolving from a linear model to a nonlinear one and then the return to a linear model were respectively performed through specially designed split and merge procedures. To obtain fast and yet flexible split and merge operations, a modified structure for TS model was considered, which allowed for more complex precedents. An adaptive threshold on temporal modeling error is utilized in the reconstruction of the model. A case study was considered to examine the applicability and performance of proposed Adaptive Habitually Linear and Transiently Nonlinear Model (AHLTNM) along with its learning algorithm.

The AHLTNM was utilized in prediction of monthly sunspots number. The performance of the proposed model and learning algorithm was compared with two other online modeling approaches. The AHLTNM showed reasonably better performance when faced with significant time-varying behaviors.

REFRENCES

[1] N. Kasabov, "The ECOS framework and the ECO learning method for evolving connectionist systems," *J. Adv. Comput. Intell.*, vol. 2, no. 6, 1998, pp. 195–202.

[2] P. Angelov, R. Buswell, "Identification of evolving fuzzy rule based models," *IEEE Trans. on Fuzzy Syst.,* vol. 10, no. 5, 2002, pp. 667-677.

[3] N. Kasabov, "DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and its application for time-series prediction," *IEEE Trans. Fuzzy Syst.*, vol. 10, 2002, pp. 144-154.

[4] M.J. Er and S.Q. Wu, "A fast learning algorithm for parsimonious fuzzy neural systems," *Fuzzy Sets and Syst.*, vol. 126, 2002, pp. 337–351.

[5] P. Angelov, P. Filev, "An approach to online identification of Takagi-Sugeno fuzzy models," *IEEE Trans. Syst., Man, and Cybern. B*, vol. 34, 2004, pp. 484-498.

[6] Y. Gao and M.J. Er, "NARMAX time series model prediction: feedforward and recurrent fuzzy neural network approaches," *Fuzzy Sets and Syst*., vol. 50, 2005, pp. 331–350.

[7] C. Petr, "Online learning of neural Takagi-Sugeno fuzzy model," *in Fuzzy Information Processing Society 2005. NAFIPS 2005. Annual Meeting of the North American,* Jun. 2005, pp. 478-483.

[8] P. Angelov, X.-W. Zhou, Evolving fuzzy systems from data streams in real-time, *in Internat. Symp. on Evolving Fuzzy Systems*, 2006, pp. 29-35.

[9] L. Liao and S. Li, "On-line T-S Fuzzy model identification with growing and pruning rules," *in Proc. of 4th International Symposium on Neural Networks 2007*, Nanjing China, Jun. 2007, pp. 505-511.

[10] K.M. Pekpe, S. Lecoeuche, "Online, clustering of switching models based on a subspace framework," *Nonlinear Analysis: Hybrid Systems*, vol. 2, no. 3 , 2008 , pp. 735-749.

[11] A. kalhor, B. N. Araabi and C. Lucas, "Online identification of a neuro-fuzzy model through indirect fuzzy clustering of data space," *FUZZ-IEEE2009, the 18th Int. conference on fuzzy systems*, 21-24 Aug., 2009, Korea, pp. 356-359.

[12] Oliver Nelles, *Nonlinear System Identification*, New York: Springer, 2001, ch.13.3.1.

[13] A. Kalhor, B. Nadjar Araabi and C. Lucas, "A New Split and Merge Algorithm for Structure Identification in Takagi-Sugeno Fuzzy Model". *7th IEEE International Conference on Intelligent Systems, Design and Applications (ISDA'07)*, Rio de Janeiro, Brazil, October 22-24, 2007. pp. 258- 261.

[14] Solar Influences Data Analysis Center (SIDC), Solar physics research department of the Royal Observatory of Belgium, http://sidc.oma.be/index.php3.

[15] A. Gholipour, B. N. Araabi and C. Lucas, "Predicting chaotic time series using neural and neuro-fuzzy models: A Comparison Study," *Neural Processing Letters*, vol. 24, no. 3, 2006, pp. 217-239.