A TS FUZZY MODEL DERIVED FROM A TYPICAL MULTI-LAYER PERCEPTRON

A. KALHOR, B. N. ARRABI, C. LUCAS AND
B. TARVIRDIZADEH

Abstract. In this paper, we introduce a Takagi-Sugeno (TS) fuzzy model which is derived from a typical Multi-Layer Perceptron Neural Network (MLP NN). At first, it is shown that the considered MLP NN can be interpreted as a variety of TS fuzzy model. It is discussed that the utilized Membership Function (MF) in such TS fuzzy model, despite its flexible structure, has some major restrictions. After modifying the MF, we introduce a TS fuzzy model whose MFs are tunable near and far from focal points, separately. To identify such TS fuzzy model, an incremental learning algorithm, based on an efficient space partitioning technique, is proposed. Through an illustrative example, the methodology of the learning algorithm is explained. Next, through two case studies: approximation of a nonlinear function for a sun sensor and identification of a pH neutralization process, the superiority of the introduced TS fuzzy model in comparison to some other TS fuzzy models and MLP NN is shown.

1. Introduction

Fuzzy models and Artificial Neural Networks (ANNs) are two different modeling tools which have been vastly tested and developed for many applications. Fuzzy models are known for their interpretable structure due to using fuzzy rules and ANNs are known for their organizing and training abilities. Applying advantages of ANNs to fuzzy models, powerful modeling tools are created. Among fuzzy models, TS fuzzy model, [28], can be organized and trained like ANNs; for this reason and also due to its flexible and interpretable structure, they have been deployed increasingly in many applications, [24], [30] and [15].

Consider a TS fuzzy model with M fuzzy rules. We describe the $j-$th fuzzy rule as follows:

$$\text{IF } \ x \text{ is } \mu_j \text{ THEN } \bar{y}^j = f_j(x) \qquad j = 1, 2, ..., M \tag{1}$$

where x denotes an $n-$ dimensional input and μ_j is $j-$th fuzzy set for x and the related Membership Function (MF) is considered as $\mu_j(x)$[1]. $\bar{y}^j$ denotes the output

Received: September 2013; Revised: October 2014; Accepted: January 2015

Key words and phrases: Takagi-Sugeno fuzzy model, Multi layer perceptron, Tunable membership functions, Nonlinear function approximation, pH neutralization process.

[1]The eq. (1) can be considered as a wide definition of the initial TS model introduced by Takagi and Sugeno. Here, we have eased the constraint which says the high-dimensional shape of the rules is t-norm of single one-dimensional fuzzy sets. This allows one to identify linear sub-models in more flexible validity regions.

of direct mapping of input-space as $j-$ th sub-model. The overall output $\hat{y}$ is computed as a weighted mean over all M rules according to:

$$\hat{y}=\sum_{j=1}^{M} A_j(x)\bar{y}^j(x) \quad A_j(x)=\mu_j(x)/\sum_{h=1}^{M}\mu_h(x) \tag{2}$$

Sub-models, if considered linear or nonlinear, result in first order or higher-order TS fuzzy models, respectively. Applying suitable MFs to a TS fuzzy model has a critical impact on its accuracy, complexity, and redundancy. Thus, different types of MFs have been developed and used in TS fuzzy models. To present the stage of knowledge in this area, Table 1 summarizes well-known approaches along with corresponding utilized MFs and optimizations techniques.

In many neuro–fuzzy modeling approaches, simple and concisely-noted MFs are preferred to be used (TS models whose MFs belong to M1 and M2 in Table 1). These often have Gaussian, sigmoid, triangular, trapezoidal or other standard structures. Although using such MFs is straightforward, they are not capable to be tuned near and far from their focal points, independently. This can decrease performance of the resulting model and increase its redundancy.

In contrast with simple MFs, more flexible MFs have been suggested in some neuro–fuzzy modeling approaches. Asymmetric version of simple MFs, multiplying of nonlinear functions and combinations of Gaussian or sigmoid functions are instances of flexible MFs which are more detailed in Table 1 (TS models whose MFs belong to M3, M4 and M5). Although such MFs are flexible, they do not have straightforward structure. This may prevent an easy and interpretable tuning from them and increase the complexity of the model.

In this paper, we introduce a capable MF whose structure is straightforward and tunable. This MF belongs to a TS fuzzy model which is derived from a typical Multi-Layer Perceptron Neural Network (MLP NN). It is revealed that the considered MLP can be interpreted as a variety of TS fuzzy model. Then, it is explained that the utilized MF in the model, despite its straightforward structure, has some restrictions. We modify the TS fuzzy model so that its MF becomes tunable, independently, in both near and far from their focal points. Also, we have designed an efficient learning for the introduced TS fuzzy model. This algorithm uses an incremental partitioning of data space, where the partitions could have flexible convex forms.

The rest of the paper is organized as follows: in the next section, fuzzy interpretation of a typical MLP NN is stated. In Section 3, we introduce a TS fuzzy model which is derived from the typical MLP NN. In Section 4 a new incremental learning algorithm of the TS fuzzy model is proposed. Some discussions about the proposed TS fuzzy model are stated in Section 5. In Section 6, through an illustrative example and two case studies- approximation of a nonlinear function for a sun-sensor and identification of a pH neutralization process- the performance of the proposed approach is shown and compared with some other approaches. The paper is concluded in Section 7.

Approaches	Utilized MFs *					Optimization techniques
	M1	M2	M3	M4	M5	
ANFIS [8]	×	×				Parameter adaptation technique is utilized to tune parameters of the model. This technique can be used with least square technique, too.
LoLiMoT [23], [9]		×				A heuristic incremental learning algorithm is utilized to partition data space and to define MFs.*A heuristic learning algorithm based on an online identification approach.
DENFIS [13]	×	×				Two offline and online implementation of maximum distance clustering have been utilized to define clusters and MFs.
MS-TSKfnn [29]	×	×				ART-like clustering which is called as discrete incremental clustering is utilized to tune MFs.
SOFNN [18]		×				A novel self-organizing approach, based on geometric and generalization criteria, is applied to add and prune rules.
[5], [27]	×	×				Through a genetic algorithm, the structure of the model (including MFs) is represented and optimized with a single chromosome.
NFPL [7]		×				The structure of the model is defined and optimized through c-means clustering algorithm and the idea of participatory learning
Successive [3]	×					Through hierarchical fair competition-based parallel genetic algorithms and information granulation, MFs are defined and tuned.
eTS + [1]		×				An evolving fuzzy clustering approach is applied to add, remove and modify the rules.
NFCRMA [20]		×				An improved version of fuzzy c-regression model clustering are utilized to learn MFs.
[2]			×			Asymmetric triangular fuzzy sets are determined through a fuzzy partitioning and using fuzzy decision trees.
[19]			×			A hybrid learning neuro-fuzzy system with asymmetric fuzzy sets is proposed. The learning methods of random optimization and least square estimation are used in hybrid way to train the system parameters.

continued on the next page

[21]			×			The number and localization of asymmetric Gaussian membership functions are performed through fuzzy partitions in the input space and their parameters are tuned by a supervised gradient descent method.
[6]				×		A generalized TS fuzzy model is introduced whose MFs are equal to the product of sigmoid functions and other un-even nonlinear functions which are computed from clustering results.
DLM [10]				×		A TS fuzzy model is introduced whose MFs are equal to the product of sigmoid functions with quadratic functions. The sigmoid functions are determined through an incremental partitioning and quadratic functions are optimized through a weighted least square optimization.
[17]					×	MFs are defined as linear combination of Gaussian functions, the parameters of MFs are tuned through back propagation error.
IVNFS-A [16]					×	Each MF is formed from parts of four Gaussian functions. The data space is partitioned by a distance clustering and, simultaneously, the parameters of the model are tuned through a parameter adaption technique.
SAMC(1) [11]					×	Each MF is defined as sum of simple Gaussian functions. Such MFs are defined corresponding to flexible shape clusters formed through an online clustering approach.
SAMC(2) [12]					×	Each MF is defined as sum of Sigmoid functions, which can represent a flexible block-based fuzzy subspace. These MFs are obtained through split-and-merge procedures in the premise parts of the fuzzy rules.

*M1: Triangular functions / Trapezoid functions,M2: Gaussians functions /Sigmoid functions, M3: MFs which are asymmetric version of M1 and M2 types,
M4: MFs which are equal to product of two or more nonlinear equations,
M5: MFs which are equal to combination of Gaussian functions.

TABLE 1. A Summary of Well-Known Approaches, Along with the Utilized Membership Functions (MFs) and Corresponding Optimizations Techniques

2. **Fuzzy Interpretation of a Typical MLP NN**

In this section, we interpret a typical MLP NN with a hidden layer as a variety of TS fuzzy model with un-normalized MFs. Then, some restrictions of the utilized MFs are discussed. Figure 1 shows structure of a $n-$dimensional input and single output MLP NN which has a hidden layer with M neurons. where $\sigma(.)$ denotes

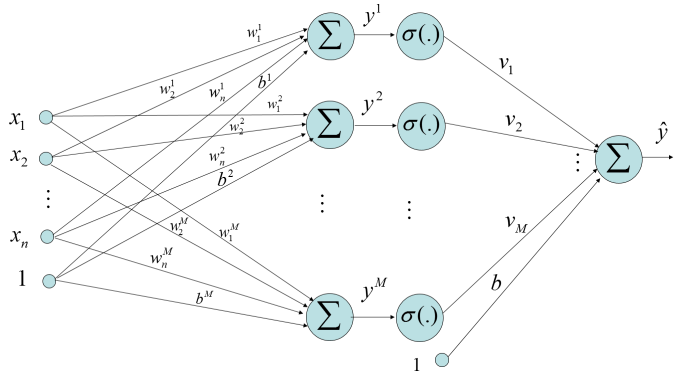


FIGURE 1. Structure of a Typical MLP NN with a Hidden Layer and M Neurons

the activation function and w_i^j $i \in \{1, 2, .., n\}$ $j \in \{1, 2, .., M\}$ and v_i are weights utilized respectively before and after the hidden layer. Considering a symmetric sigmoid function, the following equation states the input-output relation of the model.

$$\hat{y} = \sum_{j=1}^{M} v_j \tanh(\sum_{i=1}^{n} w_i^j x_i + b^j) + b \tag{3}$$

The above equation can be re-written as follows:

$$\hat{y} = \textstyle\sum_{j=1}^{M+1} \psi(\bar{\theta}_j^T z)(\theta_j^T z) \quad z^T = [x^T 1] = [x_1 x_2 \dots x_n \, 1]$$

$$\theta_j^T = v_j \bar{\theta}_j^T = \begin{cases} v_j [w_1^j \; w_2^j \dots w_n^j \;\; b^j] & j \in \{1, 2, \dots, M\} \\ \frac{b}{\psi(1)} [0 \quad 0 \;\; \dots \;\; 1 \;\;] & j = M+1 \end{cases} \tag{4}$$

$$\psi(u) = \begin{cases} \tanh(u)/u & u \neq 0 \\ 1 & u = 0 \end{cases}$$

As it is understood from (4), the simple one-argument nonlinear function $\psi(.)$ generates weighted functions for linear models. Figure 2 shows the plot of function $\psi(u)$.

As it is seen in Figure 2, since $0 \leq \psi(u) \leq 1$, it can be considered as a MF. With respect to (4) and Figure 2, the mentioned typical MLP NN can be interpreted as an extended version of TS fuzzy model with the following $M + 1$ fuzzy rules:

$$\text{Rule } j: \text{ if } \quad x \text{ is } \; A_j \text{ then } \bar{y}^j = \theta_j^T z \quad j = 1, 2, \dots, M+1 \tag{5}$$

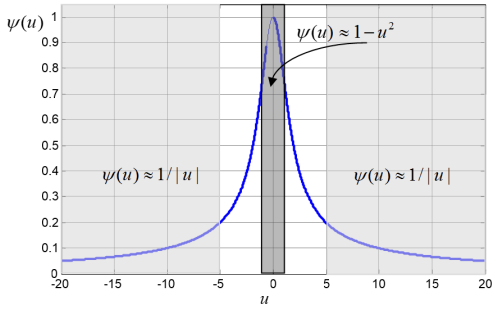


FIGURE 2. A Simple Plot of the Function $\psi(u)$

where A_js are fuzzy sets and the output of model is weighted summing up outputs of rules:

$$\hat{y} = \sum_{j=1}^{M+1} A_j(x)\bar{y}^j \qquad A_j(x) = \psi(\bar{\theta}_j^T z) \tag{6}$$

$A_j(x)$ denotes un-normalized MFs of the model. Although the MLP NN can be interpreted as a variety of TS fuzzy models, the resulting MFs have the following restrictions:

1- Since the arguments of applied MFs are hyper planes, they converge to zero just in the direction of vertical to surfaces of linear models; such MFs can generate just narrow-band validity regions and their corresponding sub-models can not be localized, perfectly.

2- The convergence rate of the MF to zero is not tunable even in the direction of vertical to surfaces of linear models.

3-They are not normalized in the model. This prevents from fitting interpolation between sub-models.

In spite of the above restrictions, the changing rate of $\psi(u)$ around $u = 0$ and when $u \to \pm\infty$ are approximated $1-u^2$ and $1/|u|$, respectively (as indicated in Figure 2); such a characteristic allows for using a mechanism to adjust the changing rate near and far from focal points, independently, which is discussed in the next section.

3. A Derived TS Fuzzy Model from the MLP NN

In this section, at first, we introduce a new MF by modifying the former stated MF in (6). Then, we re-establish the former supposed TS fuzzy model with new MFs. At the end, an incremental learning algorithm is proposed to identify the resulting TS model.

3.1. New Membership Functions.

Here, we modify the former introduced MFs in (6) as follows:

$$\mu_j(x) = \left(\psi(\|x-c_j\|_{S_j}^{n_1})\right)^{n_2} \|x-c_j\|_{S_j}^{n_1} = \left|(x-c_j)^T S_j (x-c_j)\right|^{n1/2} \qquad j = 1,2,..,M \tag{7}$$

where $c_j \in R^n$ and $S_j \in R^{n\times n}$ denote focal points and spread matrices of a supposed partition in input data space, respectively. Two positive coefficients: n_1 and n_2 are used to provide fitting changing rates for MFs when x is near to focal points and when it is far from them. Consider n_1 and n_2 are limited and $n_p = n_1 n_2$,

then, it can easily be shown that when the input of $j-$th function is very far or very near to its focal point, the MF is approximated, respectively $1/\left\|x-c_j\right\|_{S_j}^{n_p}$ and $1-(n_p/n_1)\left\|x-c_j\right\|_{S_j}^{2n_1}$(see Appendix A). Accordingly, the convergence rate of the MF to zero can be adjusted just through n_p, independent of n_1; then, after tuning n_p, the flat top of the MF around the focal point can be adjusted with n_1. Also, different S_j with different eigenvalues and eigenvectors can generate validation regions with various shapes. Figure 3 shows some plots of the MF $(\psi(u^{n_1}))^{n_2}$ for different values of n_p $(n_1=2, S_j=1)$ in (a), different values of n_1 $(n_p=8, S_j=1)$ in (b) and different $S_j(n_1=2, n_2=3)$ in (c). Figure 3 (a) shows that the MF converges to zero with greater rate when n_p increases. Figure 3 (b) shows that the flat top of MF around the focal point becomes bigger while n_p is fixed and n_1 increases. Also, Figure 3 (c) shows some plots of the MF for different spread matrices S_j. As a conclusion, the introduced MF is interpretable, flexible and tunable due to using n_1, n_p and S_j.

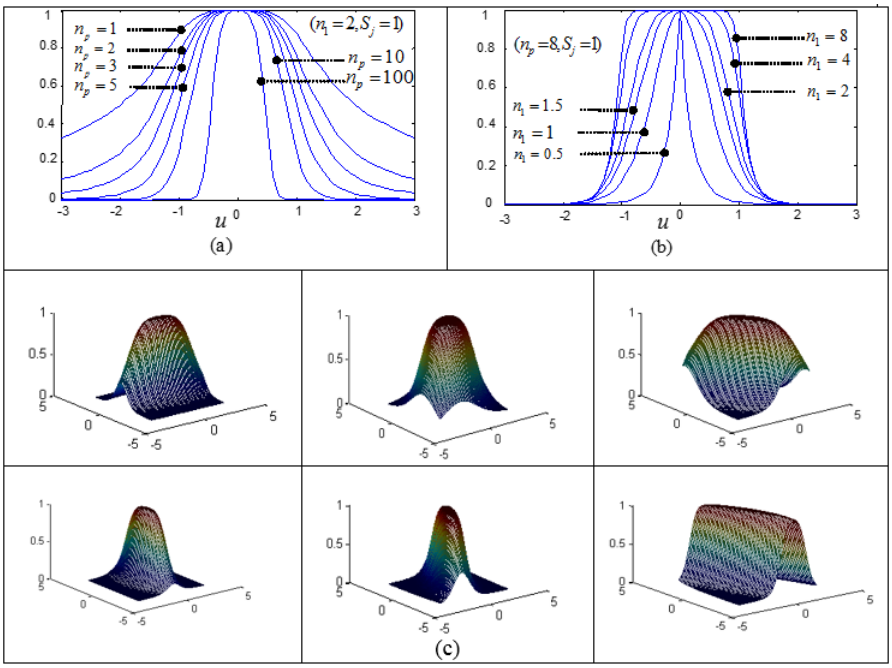


FIGURE 3. Some Plots of One and Two Dimensional MFs for Different Values of n_2 $(n_1=2, S_j=1)$ in (a), Different Values of n_1 $(n_p=8, S_j=1)$ in (b) and Different $S_j(n_1=2, n_p=8)$ in (c)

3.2. **A TS Fuzzy Model with the Proposed MFs.** Here, we use the proposed MFs in (7) in a TS fuzzy model with linear sub-models. A rule of such TS model is defined as follows:

$$\text{Rule } j: \text{ if } \quad x \text{ is } \bar{A}_j \text{ then } \bar{y}^j=\theta_j^T z \quad j=1,2,\ldots,M \tag{8}$$

where $\overline{A}_j$ denotes fuzzy sets; the output of model is the sum of weighted outputs of rules:

$$\hat{y} = \sum_{j=1}^{M} \bar{A}_j(x)\bar{y}^j \quad \bar{A}_j(x) = \mu_j(x) / \sum_{k=1}^{M} \mu_k(x) \qquad \mu_j(x) = \left(\psi(\|x - c_j\|_{S_j}^{n_1})\right)^{n_2} \quad (9)$$

where $\bar{A}_j(x)$ denotes normalized MFs of the model. Figure 4 shows a diagram including details of the resulting TS model.

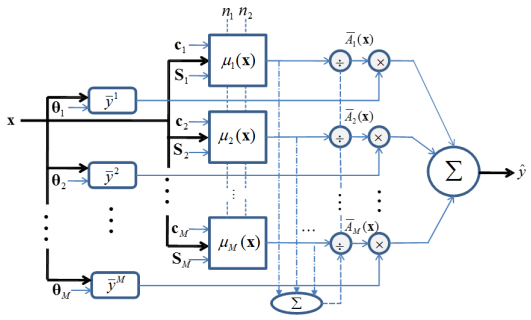


FIGURE 4. Diagram for the TS Model with MFs and Linear Sub-Models

4. **An Incremental Learning Algorithm**

In this section, an incremental learning algorithm is proposed to identify the TS fuzzy model. Figure 5 shows an overview of our proposed overall algorithm via a flowchart diagram. suppose that there are Q pairs of input and output data points:

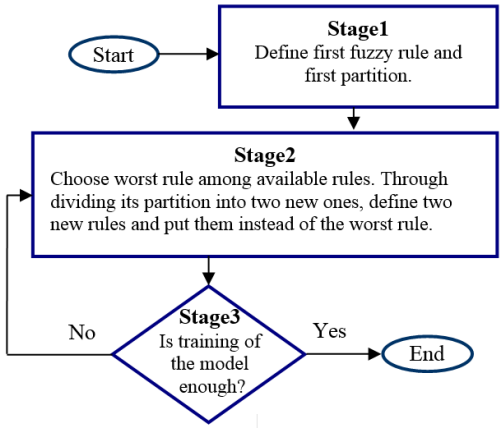


FIGURE 5. An Overview of the Proposed Algorithm

$\{x_i, y_i\}_{i=1}^{Q}$ and that $U \in R^{Q\times n}$ and $Uc_j \in R^{Q\times n}$ are matrices of input data points whose $i-th$ rows denote x_i^T and $x_i^T - c_j^T$, respectively. A partition of input-space is defined through a matrix whose rows denote input data points which are within the partition, geometrically. Now, the algorithm is detailed as follows:

***Stage1*- Define First Fuzzy Rule**

Step1: Let $M = 1$ and define first partition of data space: $U_1 = U$. Then, compute c_1 and $\bar{S}_1$ as mean and covariance of U_1, respectively.

Step2: Compute eigenvectors of $\bar{S}_1$ as $V = \{v_1, v_2, \ldots, v_n\}$, and with respect to c_1 define Uc_1. Regarding to data spread in the first partition; compute the normalization matrix of $\bar{S}_1$ as follows:

$$S_1 = \sum_{k=1}^{n} \frac{1}{L_k^2} v_k v_k^T, \;\; L_k = 0.5(\max(Uc_1^T v_k) - \min(Uc_1^T v_k)) \tag{10}$$

Now, compute the MF of the first fuzzy rule from (7) and (9).

Step3: Compute θ_1 as parameters of first linear model minimizing $J = \sum_{i=1}^{Q} \bar{A}_1(x_i)(y - \theta_1^T z_i)^2$ through weighted least square error:

$$\theta_1 = (\tilde{U}_1 \tilde{U}_1^T)^{-1} \tilde{U}_1^T \tilde{Y}_1 \tag{11}$$

where $\tilde{U}_1 \in R^{Q\times(n+1)}$and $\tilde{Y}_1 \in R^{Q\times 1}$denotes weighted input and output matrices whose $i-$throws are $\sqrt{\bar{A}_1(x_i)}z_i$and$\sqrt{\bar{A}_1(x_i)}y_i$, respectively.

***Stage2*-**Choose a rule in the model and replace it with two new rules.

Step4: Among local linear models of M available fuzzy rules, $j^* - $th rule which has maximum sum of weighted squared errors, is chosen as follows:

$$j^* = \arg_j(\max(SSE_j)), \;\; SSE_j = \sum_{i=1}^{Q} A_j(x_i)(y_i - \hat{y}_i)^2 \quad j = 1, 2, \ldots, M \tag{12}$$

Step5: Corresponding to j^*-th partition, optimize the quadratic function: $\chi_{j^*}(x) = (x - c_{j^*})^T H_{j^*}(x - c_{j^*})$, $H_{j^*} \in R^{n\times n}$. For this purpose, find optimum H_{j^*} minimizing $J = \sum_{\forall x_h^T \in U_{j^*}} A_{j^*}(x_h)(y_h - \bar{y}_h^{j^*}(1 - \chi_{j^*}(x_h))^2$ (see Appendix B).

Remark 4.1. Since H_{j^*} is symmetric, we have $H_{j^*} = \sum_{k=1}^{n} \lambda_k^{j^*}(v_k^{j^*})(v_k^{j^*})^T$, where $\lambda_k^{j^*}$ and $v_k^{j^*}$ are eigenvalues and orthogonal unitary eigenvectors of H_{j^*}. We can, then, rewrite $\chi_{j^*}(x)$ as follows:

$$\chi_{j^*}(x) = \sum_{k=1}^{n} \lambda_k^{j^*}((x - c_{j^*})^T v_k^{j^*})^2 \tag{13}$$

As it is known from (13), $\chi_{j^*}(x)$ is the sum of n convex functions passing from c_{j^*}; each convex function relates to an eigenvector along which $j^* - $th local linear model distances from the process, extremely. Thus, we consider the eigenvectors of H_{j^*} to divide the $j^* - $th partition into new partitions. For example, to divide the partition U_{j^*} into two new partitions U_{j_1} and U_{j_2} with respect to eigenvector $v_k^{j^*}$: (a) we divide input data vectors of U_{j^*} in positive and non-positive vectors with respect to their signs in the plane $\tilde{y}^{j^*} = (x - c_{j^*})^T v_k^{j^*}$. (b) we move positive and non-positive vectors of $U_{j^*}$to U_{j_1} and U_{j_2}, respectively.

Step6: For each eigenvector of H_{j^*} perform 6-1 u to 6-5, separately.
6-1: Divide partition U_{j^*} into partitions U_{j_1} and U_{j_2} with respect to eigenvector of H_{j^*}, as explained in Step 5 (Remark) .

6-2: Compute $(c_{j_1},\bar{S}_{j_1})$ and $(c_{j_2},\bar{S}_{j_2})$ as mean and covariance of U_{j_1} andU_{j_2}, respectively.
6-3: Compute eigenvectors of $\bar{S}_{j_1}$ and $\bar{S}_{j_2}$: $\{v_1^{j_1}, v_2^{j_1}, \ldots, v_n^{j_1}\}$ and $\{v_1^{j_2}, v_2^{j_2}, \ldots, v_n^{j_2}\}$, then define Uc_{j_1} and Uc_{j_2}; Now, compute S_{j_1} and S_{j_2} as follows:

$$\begin{array}{l} S_{j_1} = \sum_{k=1}^{n} \frac{1}{L_{k1}^2} v_k^{j_1} {v_k^{j_1}}^T, \;\; L_{k1} = 0.5(\max(Uc_{j_1}^T v_k^{j_1}) - \min(Uc_{j_1}^T v_k^{j_1})) \\ S_{j_2} = \sum_{k=1}^{n} \frac{1}{L_{k2}^2} v_k^{j_2} {v_k^{j_2}}^T, \;\; L_{k2} = 0.5(\max(Uc_{j_2}^T v_k^{j_2}) - \min(Uc_{j_2}^T v_k^{j_2})) \end{array} \tag{14}$$

6-4: Define two local linear models corresponding to U_{j_1}andU_{j_2}.

$$\begin{array}{l} \theta_{j_1} = (\tilde{U}_{j_1}\tilde{U}_{j_1}^T)^{-1}\tilde{U}_{j_1}^T\tilde{Y}_{j_1} \\ \theta_{j_2} = (\tilde{U}_{j_2}\tilde{U}_{j_2}^T)^{-1}\tilde{U}_{j_2}^T\tilde{Y}_{j_2} \end{array} \tag{15}$$

where $\tilde{U}_{j_1}(\tilde{U}_{j_2})$ and $Y_{j_1}(Y_{j_2})$ denote weighted input and output matrices whose $i-$th rows are $\sqrt{A_{j_1}(x_i)}z_i(\sqrt{A_{j_2}(x_i)}z)_i$ and $\sqrt{A_{j_1}(x_i)}y_i(\sqrt{A_{j_2}(x_i)}y_i)$, respectively.
6-5: Now, replace $j^* -$ thfuzzy rule with fuzzy rules j_1 and j_2, then compute Sum of Square Error (SSE) for the resulting TS fuzzy model.

Step7: Among the above n resulting TS fuzzy models in Step 6 choose the best one which has minimum SSE; put $M := M + 1$and go to the next step.

***Stage 3- Evaluate the* Stop Condition for the Algorithm.**

Step8: Stop the algorithm if the SSE of the model with M fuzzy rules for train data in comparison with SSE of the model with $M - 1$ fuzzy rules is not reduced, considerably; otherwise, go to Step 4.

5. Discussions

In this section the proposed TS model is discussed from some aspects.

5.1. **General Function Approximation Property.** We know general function approximating property for some known models like RBF, MLP and TS fuzzy have been shown under some constraints, [25], [14] and [31]. Fortunately, in our suggested learning algorithm like any other error based incremental learning algorithms, the fuzzy clustering and parameter updating are performed in a way that an error criterion such as sum of squared error is decreased at each step. If we assumed that the number of the observed data points at each operating regime is large enough it is expected that the considered error criterion is decreased continuously step by step.

As it is seen in the given learning algorithm (Section 4), at each step the worst rule which has maximum local error criterion is chosen and then it is replaced with two new rules which make maximum reduction in total error criterion (Figure 6).

Considering enough number of observed data points in operating regime of the chosen worst rule, it is expected that two new replacing rules generate better fitness at this operating regime and hence the error criterion decreased totally. The training procedure can be continued until the required precision is resulted.

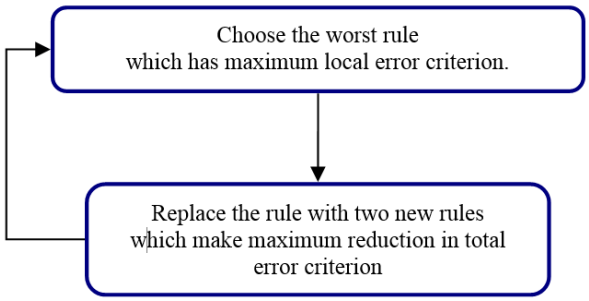


FIGURE 6. The Repeating Procedure in the Given Learning Algorithm for Proposed TS Model

5.2. **Differences with the Conventional TS Model Learning Procedures.** The proposed learning algorithm in comparison with the conventional TS learning procedures ([28]) has following main differences:

(1) In the conventional TS learning procedures, the number of fuzzy rules is predefined as a fixed value but in the proposed TS learning algorithm, the number of fuzzy rules is unknown and it is determined through an incremental learning algorithm.
(2) In the conventional TS learning procedures, some simple functions such as triangles, trapezoids, and normal distributions have been used as MFs at each axis and the weight functions are computed directly by product them. In the proposed TS learning algorithm, each MF is directly defined in n-dimensional space and the weight function is the normalized of the MF.
(3) In the proposed TS learning procedures, both premise and consequent parameters are optimized based on output error but in the conventional TS learning procedures the consequent parameters are updated based on error and the premise parameters are usually determined based on clustering or spatial aspects.

The above two first following differences make the resulting TS model more flexible but with much lower number of rules.

5.3. **Differences with the Other Incremental Learning Algorithms.** There are at least two clear differences between our approach and other incremental learning approaches:

(1) Our approach uses a more flexible way to partition the input space. In many other approaches like LoLiMoT [23] simple validity regions (Rectangular shape) are created by using an incremental axis orthogonal partitioning. Here, at each step, regarding to Eigen values and Eigen vectors the local covariance matrix of the worst rule, we use an oblique partitioning which allows us to create more flexible validity regions.
(2) In many incremental learning algorithms geometric based concepts, such as potential, [1], distance [13], are used to divide the worst cluster in two new clusters. Here an error based concept which is not sensitive to out layers

or noise is utilized for this mean. An error based approach makes better performance because it directly controls the total error of the modeling particularly in function approximation applications.

5.4. **The Computational Loads.** Since the suggested learning algorithm is based on a fast error based incremental algorithm (like LoLiMoT) in spite of using more flexible membership functions, the imposed computational load is not increased significantly. Furthermore, it should be noted that by reducing the number of rules, the number of linear sub-models decreases and bigger linear operating regimes are appeared; this is very important in some analytical applications like control or signal recovering.

In other hand, we know that the computation load is more critical in some real-time applications such as control and the priority of reducing the computation load in offline modeling approaches is not as high as online modeling approaches.

6. Case Studies

Here, some experimental results of applying the proposed learning algorithm to the introduced TS fuzzy model are presented through an illustrative example and two different case studies. The first example illustrates how the algorithm creates partitions and linear models are optimized to approximate a nonlinear function. Then, the performance of the proposed TS model is evaluated and compared in two different case studies: approximation of a nonlinear function for a sun sensor and identification of a TS model for pH neutralization process.

6.1. **An Illustrative Example.** Approximation of a Sinc function through the suggested fuzzy model is considered:

$$y = sinc(x_1, x_2) = \frac{sin(x1)}{x_1}\frac{sin(x2)}{x_2} \tag{16}$$

Seven hundred training data pairs and 350 test data pairs are selected randomly from the grid points of the range [-10, 10]×[-10,10] within the input-space of the Sinc function. At first, we find fitting $n_1$and $n_2$which provide better learning ability and generalization. To this end, we apply the proposed learning algorithm to this example by considering two rather large ranges of n_1 and n_2 when another one is fixed in an optimum value. The No-Dimensional Error Index (NDEI) is defined as the ratio of the root mean square error over the standard deviation of the target data. The optimum value of n_1 and n_2 are related to minimum NDEI of train and test data points. Figure 7 shows the results.

As it is shown in Figure 7, $n_1 = 2$ and $n_2 = 4$ ($n_p = 8$) are optimum values; since this result is repeated in some other cases, we consider $n_1 = 2$ and $n_2 = 4$ for two next case-studies, too. However, for nonlinear sub-models, optimum n_1 and $n_2$may be different (see Appendix C).

Figure 8 shows NDEI plots of training and test data sets versus number of rules for optimum values of n_1 and n_2.

As it is seen in Figure 8, the NDEI decreases when the number of rules is increased; Figure 9 shows six top-views of about forming of 6 initial partitions in

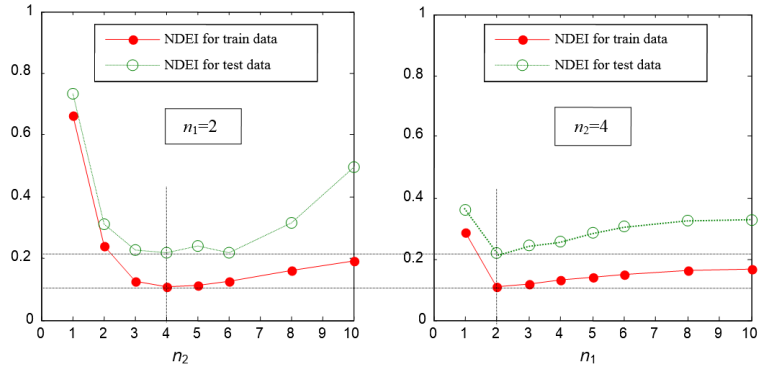


FIGURE 7. Two Plots of NDEI of Train and Test Data for Two Rather Large Ranges of n_1 and n_2 when Another One is Fixed as an Optimum Value (The Sub-Models are Linear)

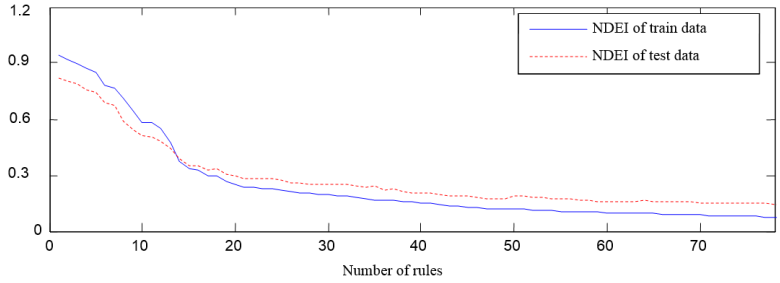


FIGURE 8. NDEI Plots of Train and Test Data Sets Versus Number of Rules for the Sinc Function

the normalized input data space. Also, spreads of data points are seen in each partition. The big circles show the focal points of clusters. The shown partitions can be identified from each other by their boundaries which are marked with bold black lines. It is seen in Figure9 that the created partitions could have convex flexible forms. The focal points of each partition (c_k) are the center points of each local linear models, too. Table 2 presents premise and consequence parameters of six initial rules identified in the TS fuzzy model. They are focal points, spread matrices and linear parameters of six initial rules of the model during performing incremental algorithm.

6.2. **Approximation of a Nonlinear Function for a Sun Sensor.** Due to memory limitations in embedded systems, it is demanded that the large look-up tables of sensors or actuators be replaced by accurate-enough nonlinear models with concise structures. Here, we apply the proposed TS with its learning algorithm to approximate the look-up table of a sun sensor utilized in attitude control of satellites.

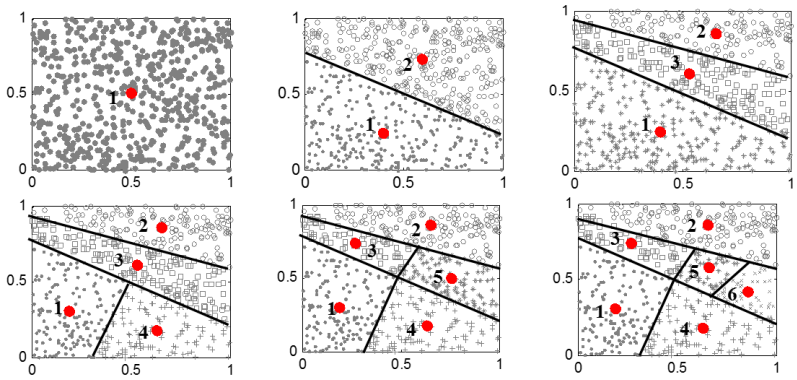


FIGURE 9. Six Top-Views About Forming of 6 Initial Partitions in the Normalized Input Data Space

Rule	c_k	S_k	θ_k
1	$\begin{bmatrix} 0.189 \\ 0.304 \end{bmatrix}$	$\begin{bmatrix} 18.968 & -0.016 \\ -0.016 & 7.322 \end{bmatrix}$	$\begin{bmatrix} 0.1650 \\ 0.0616 \\ -0.0412 \end{bmatrix}$
2	$\begin{bmatrix} 0.652 \\ 0.861 \end{bmatrix}$	$\begin{bmatrix} 4.441 & 4.203 \\ 4.203 & 28.078 \end{bmatrix}$	$\begin{bmatrix} 0.0023 \\ 0.0427 \\ -0.0369 \end{bmatrix}$
3	$\begin{bmatrix} 0.269 \\ 0.743 \end{bmatrix}$	$\begin{bmatrix} 29.595 & 35.257 \\ 35.257 & 77.909 \end{bmatrix}$	$\begin{bmatrix} -0.1235 \\ -1.0004 \\ 0.8150 \end{bmatrix}$
4	$\begin{bmatrix} 0.63 \\ 0.177 \end{bmatrix}$	$\begin{bmatrix} 9.185 & -0.374 \\ -0.374 & 14.034 \end{bmatrix}$	$\begin{bmatrix} -0.0845 \\ 0.1799 \\ 0.0306 \end{bmatrix}$
5	$\begin{bmatrix} 0.659 \\ 0.577 \end{bmatrix}$	$\begin{bmatrix} 61.475 & 3.601 \\ 3.601 & 70.82 \end{bmatrix}$	$\begin{bmatrix} -2.0892 \\ -0.7420 \\ 1.8940 \end{bmatrix}$
6	$\begin{bmatrix} 0.858 \\ 0.4213 \end{bmatrix}$	$\begin{bmatrix} 51.943 & 2.166 \\ 2.166 & 31.076 \end{bmatrix}$	$\begin{bmatrix} 0.3213 \\ 0.0652 \\ -0.3016 \end{bmatrix}$

TABLE 2. Premise and Consequence Parameters of the Rules Identified in the TS Fuzzy Model

This look-up table has 110 columns and 110 rows. Considering the indices of the rows and columns as inputs, a 3-dimensional surface for the look-up table is formed and is shown in Figure 10. Due to the rough region appearing in the last 20 rows of the table, they are memorized directly and the rest of the look-up table is considered for approximation.

The maximum absolute difference of the estimated and original values of the table,Max, is considered for evaluation. In this case, the accuracy of the model

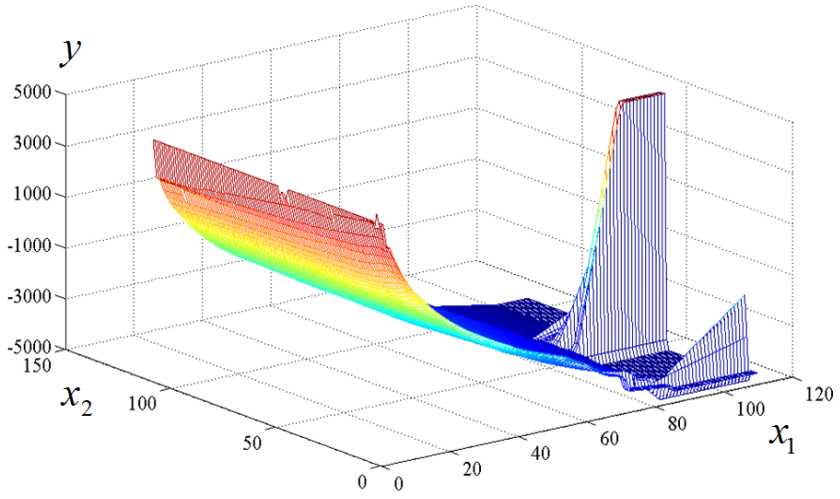


FIGURE 10. A 3-Dimensional Surface of the Look- Up Table of the Sun Sensor (the Variables x_1 and x_2 Denote Respectively Indices of Rows and Columns of the Table and y Denotes the Output of the Table)

providing $Max < 900$ is satisfactory. We apply the proposed identification algorithm to this case. Figure 11 shows NDEI diagram of data versus number of fuzzy rules.

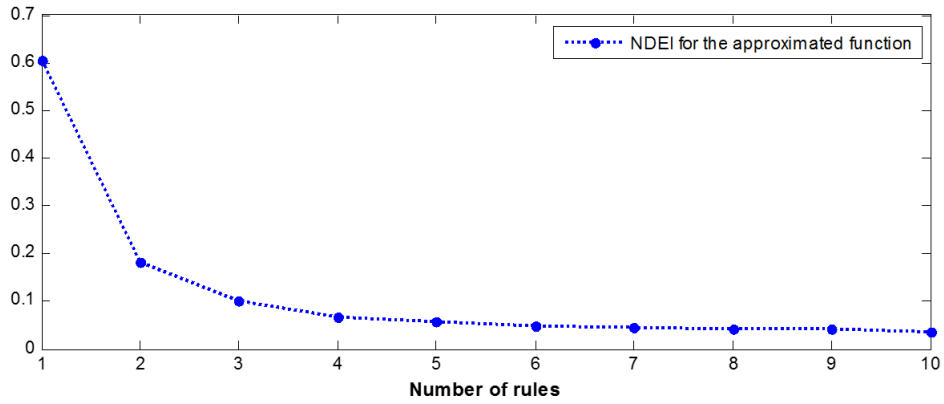


FIGURE 11. NDEI Plot of Data Set Versus Number of Rules Achieved in Approximation of a Nonlinear Function for the Sun-Sensor Case

As it is seen in Figure 11, by increasing the number of rules, the accuracy of the model becomes better with respect to resulting lower NDEI. In this case, there is no evaluation phase and only a model is trained to approximate the look-up table.

Table 3 presents the achieved modeling and approximation results from different methods. As it is seen in Table 3, our identified TS fuzzy model along with three other methods has least number of fuzzy rules. The best *Max* index with considerable distance with other ones belongs to our method. The resulting NDEI for three methods and our method are very closed to each other.

Methods	**Rules (Neurons)**	**Epochs (Iterations)**	**NDEI-train**	**Max**	**Training Time*** (sec)
MLP NN	15	100	0.0562	801	168
ANFIS	**10**	50	0.0368	706	61
DENFIS I	32	15	0.081	871	311
DENFIS II	32	50	0.063	802	721
NFCRMA	20	30	0.0346	726	614
LoLiMoT	10	-	0.033	678	5
TS-SAMC (without merge)	10	-	0.0332	666	9
Our method	**10**	-	0.0349	**557**	**16**

*All the execution times are given for a Dell XPS M1530 2.2 GHz Core 2 Duo notebook with 4GB RAM.

TABLE 3. The Modeling and Approximation Results for the Look-Up Table of the Sun Sensor

6.3. **pH Neutralization Process.** The pH neutralization process is a frequent stage in many chemical processes, which has been highly considered by chemical and material engineers. This process is performed in many applications ranged from wastewater treatments up to food industries. We use the simulation data of a pH Neutralization process in a constant volume stirring tank, [4]. The output of the process is pH of the solution in the tank and there are two exogenous inputs: Acid solution flow in liters and Base solution flow in liters. The volume of the tank is 1100 liters; the concentration of the acid solution is (HAC) 0.0032 Mol/l and the concentration of the base solution (NaOH) is 0.05 Mol/l, [22]. The sampling time for this process is 10 second and totally 2001 data have been sampled. Since this process is inherently dynamic, we add three lags of output of process as extra inputs for the process. We choose 20% of the last part of sampled data points as test data set. Now, we apply our proposed algorithm to this process. Figure 12 shows the test and train NDEI plots of the identified model versus number of fuzzy rules. With respect to the shown plots in Figure 9, the accuracy and generalization of the model become better by increasing the number of rules. We apply all mentioned methods in Table 3 to identify a model for current case study, too. The modeling and identification results are presented in Table 4. It is seen in Table 4 that our method has the least NDEI in both training and testing phases; however, the TS-SAMC and LoLiMoT have least Training times and the number of utilized fuzzy rules in ANFIS is the lowest.

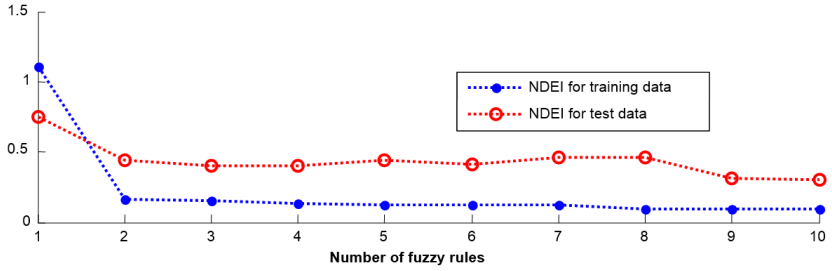


FIGURE 12. NDEI Plots of Train and Test Data Sets Versus Number of Rules Achieved in Identification of the pH Neutralization Process

Methods	**Rules (Neurons)**	**Epochs (Iterations)**	**Training time* (sec)**	**NDEI-train**	**NDEI-test**
MLP NN	15	300	57	0.1305	0.5928
ANFIS	**9**	200	40	0.1108	0.5340
DENFIS I	48	10	81	0.103	0.550
DENFIS II	48	10	84	0.073	0.636
NFCRMA	10	100	44	0.1119	0.6811
LoLiMoT	15	-	**10**	0.0842	0.6816
TS-SAMC (without merge)	12	-	8	0.0633	0.6237
Our method	10	**-**	13	**0.0606**	**0. 3969**

*All the execution times are given for a Dell XPS M1530 2.2 GHz Core 2 Duo notebook with 4GB RAM.

TABLE 4. The Modeling and Identification Results for the pH Neutralization Process

7. **Conclusions**

Inspired by a typical MLP, we introduced new straightforward MFs whose changing rates are tunable near and far from their focal points, independently. For this purpose, we interpreted the MLP NN like a variety of TS fuzzy model. Then, by improving the utilized MF, we considered it in the structure of the TS fuzzy model. We proposed an efficient learning algorithm to identify the model based on an incremental partitioning of the data space. Through an illustrative example, we explained the performance and methodology of the algorithm. Also, in approximation of a nonlinear function for a sun sensor and identification of a pH neutralization process, we demonstrated its superiority in comparison to a general MLP NN and some other well-known TS fuzzy models. Actually, it was shown that our proposed TS model, with fairly low number of fuzzy rules and execution times, provided favorable accuracy and generalization.

8. **Appendix**

8.1. **Appendix A.** Since $\psi(u)$ when $u \to 0$ is approximated with $\psi(u) \approx 1 - u^2$**,** we have:

$$\mu_j(x) = \underset{\|x-c_j\|\to 0}{\left(\psi(\|x-c_j\|_{S_j}^{n_1})\right)^{n_2}} \cong \underset{\|x-c_j\|\to 0}{\left(1-\|x-c_j\|_{S_j}^{2n_1}\right)^{n_2}} \tag{a1}$$

Also, since n_1 and n_2 are limited it is resulted: $\|x-c_j\|_{S_j}^{2n_1} \to 0$ and then we have form binomial formula:

$$\begin{aligned}\mu_j(x) &\cong \underset{\|x-c_j\|_{S_j}^{2n_1}\to 0}{\left(1-\|x-c_j\|_{S_j}^{2n_1}\right)^{n_2}} = \sum_{k=0}^{n_2} \underset{\|x-c_j\|_{S_j}^{2n_1}\to 0}{\binom{n_2}{k}} (\|x-c_j\|_{S_j}^{2n_1})^k \\ &= 1 - n_2\|x-c_j\|_{S_j}^{2n_1} - \sum_{k=2}^{n_2} \underset{\|x-c_j\|_{S_j}^{2n_1}\to 0}{\binom{n_2}{k}} (\|x-c_j\|_{S_j}^{2n_1})^k \cong 1 - n_2\|x-c_j\|_{S_j}^{2n_1}\end{aligned} \tag{a2}$$

We know $n_p = n_1 n_2$, so as a conclusion we have the following result:

$$when \;\; \|x-c_j\| \to 0 \qquad \Rightarrow \qquad \mu_j(x) \cong 1 - (n_p/n_1)\|x-c_j\|_{S_j}^{2n_1} \tag{a3}$$

In contrast, when $u \to \pm\infty$ we know$\psi(u) \cong 1/u$, and since n_1 and n_2 are positive limited values, the following result is obtained easily:

$$when \;\; \|x-c_j\| \to \infty \qquad \Rightarrow \qquad \mu_j(x) \cong 1/\|x-c_j\|_{S_j}^{n_p} \tag{a4}$$

8.2. **Appendix B.** Here, we find optimum H_{j^*} for the quadratic function: $\chi_{j^*}(x) = (x-c_{j^*})^T H_{j^*}(x-c_{j^*})$, $H_{j^*} \in R^{n\times n}$ minimizing J as

$$J = \sum_{\forall x_h^T \in U_{j^*}} A_{j^*}(x_h)(y_h - y_h^{j^*}(1-\chi(x_h))^2$$

$$\begin{aligned}J &= \sum_{\forall x_h^T \in U_{j^*}} A_{j^*}(x_h)(y_h - y_h^{j^*}(1-\chi(x_h))^2 \\ &= \sum_{\forall x_h^T \in U_{j^*}} A_{j^*}(x_h)((y_h - y_h^{j^*}) + y_h^{j^*}\chi(x_h))^2, \\ &\qquad\qquad b_h = (y_h - y_h^{j^*}) \;\; a_h = y_h^{j^*} \;\; u_h = (x_h - c_{j^*}) \\ &= \sum_{\forall x_h^T \in U_{j^*}} A_{j^*}(x_h)(b_h - a_h u_h^T H_{j^*} u_h)^2 \\ &= \sum_{\forall x_h^T \in U_{j^*}} A_{j^*}(x_h)(b_h - a_h u_h^T H_{j^*} u_h)^2 \qquad (b1) \\ &= \sum_{\forall x_h^T \in U_{j^*}} A_{j^*}(x_h)(b_h^2 - 2b_h a_h u_h^T H_{j^*} u_h + a_h^2 u_h^T H_{j^*} u_h u_h^T H_{j^*} u_h), \; T_{hh} = u_h u_h^T \\ &= \sum_{\forall x_h^T \in U_{j^*}} A_{j^*}(x_h) b_h^2 - 2\sum_{\forall x_h^T \in U_{j^*}} A_{j^*}(x_h) b_h a_h u_h^T H_{j^*} u_h \\ &\qquad + \sum_{\forall x_h^T \in U_{j^*}} A_{j^*}(x_h) a_h^2 u_h^T H_{j^*} u_h u_h^T H_{j^*} u_h \\ &= \sum_{\forall x_h^T \in U_{j^*}} A_{j^*}(x_h) b_h^2 - 2\sum_{\forall x_h^T \in U_{j^*}} A_{j^*}(x_h) b_h a_h trace(T_{hh} H_{j^*}) \\ &\qquad + \sum_{\forall x_h^T \in U_{j^*}} A_{j^*}(x_h) a_h^2 trace(T_{hh} H_{j^*} T_{hh} H_{j^*})\end{aligned}$$

Here, we compute the extremum of J as follows:

$$\frac{\partial J}{\partial H_{j^*}} = 0$$

$$\Rightarrow \sum_{\forall x_h^T \in U_{j^*}} A_{j^*}(x_h) b_h a_h T_{hh} - \sum_{\forall x_h^T \in U_{j^*}} A_{j^*}(x_h) a_h^2 T_{hh} H_{j^*} T_{hh} = 0 \tag{b2}$$

$$\Rightarrow vec(\sum_{\forall x_h^T \in U_{j^*}} A_{j^*}(x_h) b_h a_h T_{hh}) - \sum_{\forall x_h^T \in U_{j^*}} A_{j^*}(x_h) a_h^2 (T_{hh} \otimes T_{hh}) vec(H_{j^*}) = 0$$

$$\sum_{\forall x_h^T \in U_{j^*}} A_{j^*}(x_h) a_h^2 (T_{hh} \otimes T_{hh}) vec(H_{j^*}) = vec(\sum_{\forall x_h^T \in U_{j^*}} A_{j^*}(x_h) b_h a_h T_{hh})$$

The $\otimes$ denotes the Kronecker product and vec is an operator used to vectorize a matrix, [26].

Equation (b2) presents the resulting set of linear equations; thus H_{j^*} can be computed easily form it after removing repeating linear equations because the matrix H_{j^*} is symmetric. Also, since $\partial^2 J / \partial H_{j^*}^2 = 2 \sum_{\forall x_h^T \in U_{j^*}} A_{j^*}(x_h) a_h^2 (T_{hh} \otimes T_{hh})$ is a positive definitive matrix, the resulting H_{j^*} from equation (b2) minimizes J.

8.3. **Appendix C.** Here, for quadratic sub-models in our introduced TS model, we find optimum n_1 and n_2. For this purpose, the explained setup in the illustrative example (Figure 7) is repeated. Figure 13 shows two plots of NDEI of train and test data for two rather large ranges of n_1 and n_2 when another one is fixed as an optimum value.

As it is seen, the minimum computed NDEI, for resulting TS models of different n_1 and n_2, are related to $n_1 = 2$ and $n_2 = 5$. This result, in comparison to obtained result in Figure 7, shows that the optimum $n_1$and n_2 for sub-models with different degrees can be different.

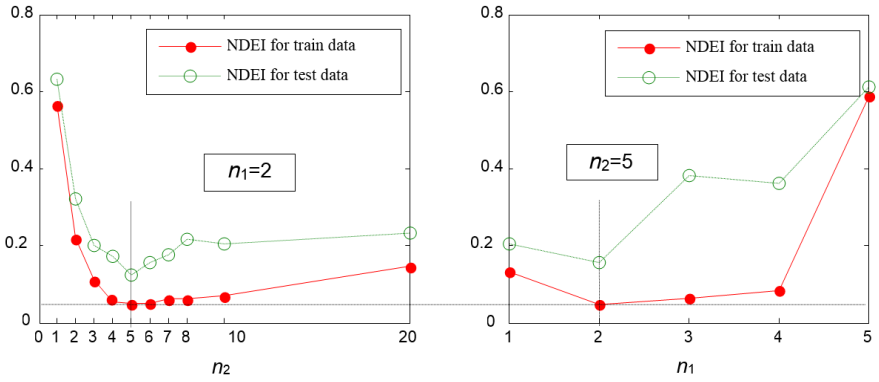


FIGURE 13. Two Plots of NDEI of Train and Test Data for Two Rather Large Ranges of n_1 and n_2 when Another One is Fixed as an Optimum Value (The Sub-Models are Quadratic)

REFERENCES

[1] P. Angelov and X. Zhou, *On line learning fuzzy rule-based system structure from data streams,* In 2008 IEEE International Conference on Fuzzy Systems within the IEEE World Congress on Computational Intelligence, Hong Kong., (2008), 915-922.

[2] J. F. Baldwin and S. B. Kandarake, *Asymmetric triangular fuzzy sets for classification model*, In Lecture Notesin Artificial Intelligent, (2003), 364-370.
[3] J. N. Choi, S. K. Oh and W. Pedrycz., *Identification of fuzzy models using a successive tuning method with a variant identification ratio*, Fuzzy Sets and Systems, **159** (2008), 2873-2889.
[4] B. L. R. De Moor, ed., *DaISy: Database for the Identification of Systems, Department of Electrical Engineering*, 2008, ESAT/SISTA, K.U.Leuven, Belgium, URL: http://homes.esat.kuleuven.be/˜smc/daisy/. (visited on Oct. 10, 2010).
[5] H. Du and N. Zhang, *Application of evolving Takagi-Sugeno fuzzy model to nonlinear system identification*, Applied soft computing, **8** (2007), 676-686.
[6] A. Fiordaliso, *A constrained Takagi-Sugeno fuzzy system that allows for better interpretation and analysis*, Fuzzy Sets and Systems, **118** (2001), 307-318.
[7] M. Hell, S. P. Campinas, R. Ballini, Jr. P. Costa and F. Gomid, *Training neurofuzzy networks with participatory learning*, In: proceeding of 5th Conference of the EUSFLAT, (2007), 231-236, .
[8] J. S. R. Jang, *ANFIS: Adaptive-network-based fuzzy inference system*, IEEE Trans. Syst. Man and Cybern, **23** (1993), 665-685.
[9] A. Kalhor, B. N. Araabi and C. Lucas, *A new systematic design for habitually linear evolving TS fuzzy model*, Journal of Expert systems with applications, **39** (2012), 1725-1736.
[10] A. Kalhor, B. N. Araabi and C. Lucas, *An online predictor model as adaptive habitually linear and transiently nonlinear model*, Evolving Systems, **1** (2010), 29-41.
[11] A. Kalhor, B. N. Araabi and C. Lucas, *A new high-order Takagi-Sugeno fuzzy model based on deformed linear models*, Amirkabir Int. J. of Modeling Identification, Simulation and Control, **42** (2010), 43-52.
[12] A. Kalhor, B. N. Araabi and C. Lucas, *Reducing the number of local linear models in neuro–fuzzy modeling: A split-and-merge clustering approach*, Applied Soft Computing Journal, **11** (2011), 5582–5589.
[13] N. Kasabov, *DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and its application for time-series prediction*, IEEE Trans. Fuzzy Syst., **10** (2012) 144-154.
[14] V. Krkov: Kolmogorov's theorem and multilayer neural networks, *Neural networks* 5: 501-506, 1992.
[15] D. H. Lee, Y. H. Joo and M. H. Tak, *Local stability analysis of continuous-time Takagi–Sugeno fuzzy systems: A fuzzy Lyapunov function approach,* Information Sciences, **257** (2014), 163-175.
[16] C. H. Lee and H. Y. Pan, *Performance enhancement for neural fuzzy systems using asymmetric membership functions*, Fuzzy Sets and Systems, **160** (2009), 949-971.
[17] C. H. Lee and C. C. Teng, *Fine tuning of membership functions for fuzzy neural systems*, Asian J. Control, **3** (2001), 216-225.
[18] G. Leng, T. M. Mc Ginnity and G. Prasad, *An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network*, Fuzzy Sets and Systems, **150** (2005), 211-243.
[19] C. Li C, K. H. Cheng and J. D. Lee, *Hybrid learning neuro fuzzy approach for complex modeling using asymmetric fuzzy sets*, In: Proc. of the17th IEEE International Conf. on Tools with Artificial Intelligence, (2005), 397-401.
[20] C. Li, J. Zhou, X. Xiang, Q. Li and X. An, *T-S fuzzy model identification based on a novel fuzzy c-regression model clustering algorithm*, Engineering Applications of Artificial Intelligence, **22** (2009), 646-653.
[21] C. J. Lin and W. H. Ho, *An asymmetric-similarity-measure-based neural fuzzy inference system, Fuzzy Sets and Systems*, **152** (2005), 535–551.
[22] T. J. Mc Avoy, E. Hsu and S. Lowenthal, Dynamics *of pH in controlled stirred tank reactor*, Ind. Eng. Chem. Process Des. Develop., **11** (1972), 71-78.
[23] O. Nelles, *Nonlinear System Identification*, In: New York: Springer, (2001), Section 13.3.1, 365-37.
[24] P. Nikdel, M. Hosseinpour, M. A. Badamchizadeh and M. A. Akbari, *Improved Takagi–Sugeno fuzzy model-based control of flexible joint robot via Hybrid-Taguchi genetic algorithm*, Engineering Applications of Artificial Intelligence, **33** (2014), 12-20.

[25] J. Park and I. W. Sandberg, *Universal Approximation Using Radial-Basis-Function Networks*, Neural Computation, **3** (1991), 246-257.
[26] K. B. Petersen and M. S. Pedersen, *The matrix cookbook*, http://matrixcookbook.com, Version: Nov. 14, 2008.
[27] Y. Shi, R. Eberhart and Y. Chen, *Implementation of evolutionary fuzzy systems*, IEEE Trans. on Fuzzy Sys., **7** (1999), 109-118.
[28] T. Takagi and M. Sugeno, *Fuzzy identification of systems and its applications to modeling and control*, IEEE Trans. Syst., Man, and Cybern., **15** (1985), 116-132.
[29] D. Wang, C. Quek and G. S. Ng, MS-*TSKfnn: Novel Takagi-Sugeno-Kang fuzzy neural network using ART like clustering.*, In: proceeding of IEEE international joint conference on Neural Networks., (2004), 2361-2366.
[30] X. Xie, L. Lin and S. Zhong, *Process Takagi–Sugeno model: A novel approach for handling continuous input and output functions and its application to time series prediction*, Knowledge-Based Systems, **63** (2014), 46-58.
[31] H. Ying, *General SISO Takagi–Sugeno fuzzy systems with linear rule consequent are universal approximators*, IEEE Trans. on Fuzzy Systems, **6** (1998), 582-587.

A. Kalhor*, System Engineering and Mechatronics Group, Faculty of New Sciences and Technologies, University of Tehran, Tehran, Iran
E-mail address: akalhor@ut.ac.ir

B. N. Aarabi, Control and Intelligent Processing Center of Excellence, School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran
E-mail address: araabi@ut.ac.ir

C. Lucas, Control and Intelligent Processing Center of Excellence, School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran
E-mail address: lucas@ut.ac.ir

B. Tarvirdizadeh, System Engineering and Mechatronics Group, Faculty of New Sciences and Technologies, University of Tehran, Tehran, Iran
E-mail address: bahram@ut.ac.ir

*Corresponding author