NeuroFAST: On-Line Neuro-Fuzzy ART-Based Structure and Parameter Learning TSK Model

Spyros G. Tzafestas and Konstantinos C. Zikidis

***Abstract*—NeuroFAST is an on-line fuzzy modeling learning algorithm, featuring high function approximation accuracy and fast convergence. It is based on a first-order Takagi–Sugeno–Kang (TSK) model, where the consequence part of each fuzzy rule is a linear equation. Structure identification is performed by a fuzzy adaptive resonance theory (ART)-like mechanism, assisted by fuzzy rule splitting and adding procedures. The well known δ rule continuously performs parameter identification on both premise and consequence parameters. Simulation results indicate the potential of the algorithm. It is worth noting that NeuroFAST achieves a remarkable performance in the Box and Jenkins gas furnace process, outperforming all previous approaches compared.**

***Index Terms*—δ rule, fuzzy ART learning, structure/parameter identification, Takagi–Sugeno–Kang (TSK) fuzzy reasoning model.**

I. Introduction

Fuzzy set theory [1] was initially proposed as a tool for the expression and manipulation of human-like, expert knowledge. Combined with the learning ability of artificial neural networks, it was proved to be a powerful mathematical construct, enabling the symbolic expression of machine learning results. In the last few years, the application of neuro-fuzzy methods to nonlinear process identification using input–output (I/O) data is a very active area. A comprehensive survey can be found in [2], with a plethora of references.

One of the most influential fuzzy reasoning models was proposed by Takagi and Sugeno in [4]. In this model, the consequent part of each fuzzy rule is expressed as a linear function of the input variables, instead of a fuzzy set [3], reducing the number of required fuzzy rules. Since then, Sugeno and his colleagues established what is called today the *Takagi–Sugeno–Kang* (TSK) model [5], [6].

Fuzzy modeling involves structure and parameter identification. The second is usually (and easily) addressed by some gradient descent variant, e.g., the least squares algorithm or back-propagation. Structure identification is a more difficult task, often tackled by off-line, trial-and-error approaches, like the unbiasedness criterion [5], [7]. One of the most common methods for structure initialization is uniform partitioning of each input variable range into fuzzy sets, resulting to a fuzzy grid. This approach is followed in ANFIS, a well-known TSK model learning algorithm [8]. In [10]–[12], [20] the TSK model was used for designing various neurofuzzy controllers.

In the proposed approach, the human way of thinking is exploited as much as possible, and is incorporated into an automated procedure. NeuroFAST is a fuzzy modeling learning algorithm based on the TSK model and features on-line structure and parameter identification, very good numerical accuracy and fast convergence, for use in supervised, real time function approximation tasks. The input space is automatically partitioned into fuzzy subsets, using a modified fuzzy ART (adaptive resonance theory) [19] mechanism. Fuzzy rules that tend to give high output error are split in two, by a specific fuzzy rule splitting procedure, resulting in a *fuzzy k-d tree* structure. To cope with "hard" regions of extreme nonlinearities, a new fuzzy rule is created wherever the output error exceeds a dynamic threshold. At the same time, all adaptive parameters are tuned by the δ rule. The proposed algorithm is an extension to [18], where a fuzzy ARTMAP [25] module was employed. A more technically detailed presentation of the proposed architecture appears in [9].

The performance of NeuroFAST is verified by two examples: the fuzzy modeling of a nonlinear function and the prediction of the Box and Jenkins gas furnace process [21], a famous benchmark for system identification algorithms, where NeuroFAST exhibits outstanding performance over other existing methods.

Manuscript received January 16, 1998; revised May 31, 2001. This paper was recommended by Editor K. Pattipati.

The authors are with the Intelligent Robotics and Automation Laboratory, Electrical and Computer Engineering Department, National Technical University of Athens, Athens, Greece (e-mail: tzafesta@softlab.ece.ntua.gr; kzikidis@softlab.ece.ntua.gr).

Publisher Item Identifier S 1083-4419(01)08545-4.

II. Presentation of NeuroFAST

It is assumed that the input and output variables are known. In this study, issues concerning the choice of input variables from all possible variables are not dealt with. For discussions on this subject the reader is referred to [5], [13], [16], and [32].

A. General Description and Basic Concepts

The core of the present system is the TSK model [4], namely a set of IF…THEN rules with fuzzy implications and first-order functional consequence parts, which was proved to be a universal approximator [26]. The format of the fuzzy rule $\mathrm{R_i}$ is

$$\mathrm{R_i}\text{: If } x_1 \text{ is } A_{i1} \text{ AND } \ldots \text{ AND } x_M \text{ is } A_{iM} \text{ then } y_i = c_{i0} + c_{i1}x_1 + \cdots + c_{iM}x_M.$$

The number of rules is determined by the user, depending on the task and the available or expected training data. The algorithm creates linear models that approximate locally the function to-be-learned. Structure identification sets a coarse fuzzy partitioning of the domain, while parameter identification optimally adjusts premise and consequent parameters.

A modified fuzzy ART mechanism [19] is employed for domain partitioning. The idea of utilizing fuzzy ART concepts for structure learning was introduced by Lin *et al.* [22]–[24]. Fuzzy ART is an unsupervised algorithm, which receives a stream of input patterns and automatically creates recognition categories or hyperboxes. These recognition categories start as points in the input space and increase in size to incorporate new points that are presented, until the whole input space is covered. The maximum size of these hyperboxes and implicitly the number of the required hyperboxes can be adjusted by a parameter called "vigilance." In our case, the input stream is formed by the input variable vector at every time step. The (crisp) hyperboxes are fuzzified, providing the implications of the fuzzy rules.

Fuzzy ART does not allow for any dependence on the output error, and results into a more-or-less uniform hyperbox allocation. This is not desired, because the function-to-be-learned is not assumed to exhibit uniform "difficulty" in its whole domain. To help the algorithm learn better in "hard" areas, a *fuzzy rule splitting* technique is employed: periodically, all rules are examined and the rule with the worst local performance index is split in two rules. By "split" it is meant that the hyperbox associated with the fuzzy implication of the rule is divided into two hyperboxes, after a *guillotine cut* across one dimension. This results to a *fuzzy k-d tree* structure. The question that arises is across which dimension the cut should be made. Various strategies have been proposed, including the *balanced sampling criterion* [15], *direct evaluation* [5] and *regional linearity* [15], employed in the proposed approach. The fuzzy rule splitting procedure adds to the algorithm computational complexity and overhead; this is the price for on-line structure identification.

Apart from fuzzy rule splitting, another procedure is *fuzzy rule adding*: a new rule is created wherever the output error exceeds a high threshold.

After all available rules have been used, the rule splitting and adding procedures are terminated. However, fuzzy ART learning remains active as a "watchdog," preventing uncovering of the input space. The δ rule, which is active from the beginning, continues and fine tunes all parameters, until the algorithm reaches some stopping criterion.

The presentation of the algorithm will be made through a multi-input single-output example. A certain familiarity with the fuzzy ART model [19] is assumed.

B. Adaptive Parameters

The adaptive parameters are: the weights associated with the membership functions (which will be called input weights), the slopes of the membership functions, and the weights of the consequence parts of the rules (which will be called output weights). At the beginning, there are N uncommitted or available nodes (each node corresponds to a hyperbox, a fuzzy subset, and a fuzzy rule). Suppose that there are M input variables. Let

$$\begin{aligned}\mathbf{w_i} &= (w_{i1},\ \ldots,\ w_{iM},\ w_{iM+1},\ \ldots,\ w_{i2M}) \\ &= (u_{i1},\ \ldots,\ u_{iM},\ v_{i1}^c,\ \ldots,\ v_{iM}^c) \in [0,\ 1]^{2M}\end{aligned}$$

be the input weights defining the (i)th hyperbox. Note that: $v_{ij}^c = 1 - v_{ij}$ (complement coding form [19]).

As input vectors are presented, hyperboxes are created (which means that nodes become committed) and expand to cover the input space. These hyperboxes are fuzzified, forming the implications of the fuzzy rules. The fuzzy implication of the (i)th rule is defined by $\mathbf{w_i}$ and $\mathbf{f_i}$, where $\mathbf{f_i}$ is the vector of the slopes of the associated membership functions

$$\mathbf{f_i} = (f_{i1},\ \ldots,\ f_{iM},\ f_{iM+1},\ \ldots,\ f_{i2M}) \in \Re_+^{2M}.$$

Details on the membership functions will be given in the simulation section. The consequence parts of the fuzzy rules are linear equations of the input variables (with a bias term), as mentioned above. Let

$$\mathbf{c_i} = (c_{i0},\ c_{i1},\ \ldots,\ c_{iM}) \in \Re^M \qquad i = 1,\ \ldots,\ \mathrm{N}$$

be the output weights of the (i)th fuzzy rule.

C. Learning Parameters and Performance Indices

The ART learning parameters are the choice α, vigilance ρ, and learning rate β. The learning rate parameters for the δ rule are lr_1, lr_2, and lr_3, and are associated with the updating of $\mathbf{c}$, $\mathbf{w}$, and $\mathbf{f}$, respectively. All δ rule learning rates decrease slowly with time. Another parameter is P, which adjusts how often the check for the "worst" fuzzy rule takes place.

Initialization of these parameters with some standard values is expected to provide acceptable performance. However, optimal setting is not a trivial task and usually requires some trial-and-error testing.

The global performance index is the *mean square error* (MSE). A number of local performance indices are kept for each fuzzy rule, utilized by the rule splitting procedure.

D. Main Body of the Algorithm

Let $\mathbf{x} = (x_1,\ \ldots,\ x_M)$ be the input vector at a given time step and y the associated desired output value. All these variables must be normalized to [0,1].

1) Calculation of Node Activation: The first step is to calculate the activation of each committed node for this input vector. Let $M_i(x)$ be the fuzzy implication membership function value or firing strength of the (i)th rule. The rules/nodes whose firing strength $M_i(x)$ is higher than a small fixed threshold, e.g., 0.001, are activated and take part in output calculation and weight updating. All other nodes remain idle and are ignored. The sum of all rule firing strengths is defined as

$$S(\mathbf{x}) = \sum_{\text{all activated rules}} M_i(\mathbf{x}).$$

If $S(\mathbf{x})$ is zero (which means that there are no active rules), the algorithm imperatively performs fuzzy ART learning (hyperbox expansion/creation) to incorporate this "new" input vector. In order to prevent possible uncovering of input space, fuzzy ART learning is called for even if $S(\mathbf{x})$ is below a certain level (set to 0.1).

2) Fuzzy ART Learning: During the first learning stage or whenever the sum of the rule firing strengths is low, the fuzzy ART mechanism operates as follows [19].

The *choice function* of every committed node is calculated: $|\mathbf{x} \wedge \mathbf{w_i}|/(\alpha + |\mathbf{w_i}|)$. The node with the highest choice value is selected. If node i is selected, the *match function* $|\mathbf{x} \wedge \mathbf{w_i}|/|\mathbf{x}|$ is calculated. This value is compared to the vigilance ρ. If $|\mathbf{x} \wedge \mathbf{w_i}|/M < \rho$, *mismatch reset* occurs, the choice value of this category is set to -1, and a new search starts. If no committed node satisfies the vigilance criterion (becomes *resonant*), an uncommitted one is chosen and initialized. If a committed node, e.g., the (i)th node, is chosen and satisfies the vigilance criterion, then input weight updating is made

$$\mathbf{w_i^{new}} = \beta \cdot (\mathbf{x} \wedge \mathbf{w_i^{old}}) + (1 - \beta) \cdot \mathbf{w_i^{old}}.$$

3) Recalculation of Node Activation: If fuzzy ART learning was performed, the node activations $M_i(x)$ are recalculated. If $S(\mathbf{x})$ is nonzero, the algorithm carries on to the following steps. Otherwise, it stops and proceeds to the next input, a case not so unusual during the first learning stage.

4) Output Calculation: The output value is calculated for each active rule

$$\text{If } M_i(\mathbf{x}) > 0.001, \qquad y_i = c_{i0} + c_{i1}\overline{x}_1 + \cdots + c_{iM}\overline{x}_{iM}$$

where $\overline{x}_1,\ \ldots,\ \overline{x}_M$ are the input variables $x_1,\ \ldots,\ x_M$ after a simple linear transformation, intended to intensify their small "variations" inside each fuzzy set. The global output value is the weighted average of the output values of the activated fuzzy rules

$$\begin{aligned}output &= \frac{\sum_{\text{all activated rules}} M_i(\mathbf{x}) \cdot y_i}{\sum_{\text{all activated rules}} M_i(\mathbf{x})} \\ &= \frac{\sum_{\text{all activated rules}} M_i(\mathbf{x}) \cdot y_i}{S(x)}\end{aligned}$$

provided that

$$S(\mathbf{x}) = \sum_{\text{all activated rules}} M_i(\mathbf{x}) > 0.$$

5) Training Error and Performance Index: The training error is the desired output value minus the actual output

$$error = y - output$$

MSE is updated as follows:

$$MSE^{new} = 0.9995 \cdot MSE^{old} + 0.0005 \cdot error^2.$$

This is a convenient way of storing and updating *MSE*. The factor 0.9995 may depend on the task.

6) Weight Updating: Only the activated fuzzy rules take part in the weight updating procedure. First, the input weights and the slopes of the membership functions are updated, according to the δ rule. The updating equations for one membership function are derived in the Appendix, while analytical details can be found in [9]. The output weights

TABLE I
MEMBERSHIP FUNCTIONS AND FUZZY IMPLICATIONS

	Membership function corresponding to the (j) th variable of the (i) th rule	Fuzzy implication membership function of the (i) th rule
1	This fuzzy implication derives directly from fuzzy ART values. No membership function can be defined for each separate input variable.	$_1M_i(\mathbf{x}) = \max(0,1 - f_i + f_i \cdot \frac{\lvert\mathbf{x} \wedge \mathbf{w_i}\rvert}{\lvert\mathbf{w_i}\rvert + a})$
2	$_2\mu_{ij}(x_j) = \begin{cases} \max(0,1 - f_{ij} \cdot (u_{ij} - x_j)), \text{if } (x_j < u_{ij}) \\ \max(0,1 - f_{i(M+j)} \cdot (v_{ij} - x_j^c)), \text{if } (x_j^c < v_{ij}) \\ 1, \text{otherwise} \end{cases}$	$_2M_i(\mathbf{x}) = \min_j[_2\mu_{ij}(x_j)]$ $_3M_i(\mathbf{x}) = \prod_j {}_2\mu_{ij}(x_j)$
3	$_3\mu_{ij}(x_j) = \begin{cases} \frac{1}{1+\exp[-f_{ij} \cdot (x_j - u_{ij})]}, \text{if } x_j < x_{join} \\ \frac{1}{1+\exp[-f_{iM+j} \cdot (x_j^c - v_{ij}^c)]}, \text{otherwise} \end{cases}$	$_4M_i(\mathbf{x}) = \prod_j {}_3\mu_{ij}(x_j)$
4	$_4\mu_{ij}(x_j) = \begin{cases} \frac{1+\exp[-f_{ij} \cdot (mean_{ij} - u_{ij})]}{1+\exp[-f_{ij} \cdot (x_j - u_{ij})]}, \text{if } x_j < x_{join} \\ \frac{1+\exp[-f_{iM+j} \cdot (mean_{ij}^c - v_{ij}^c)]}{1+\exp[-f_{iM+j} \cdot (x_j^c - v_{ij}^c)]}, \text{otherwise} \end{cases}$	$_5M_i(\mathbf{x}) = \prod_j {}_4\mu_{ij}(x_j)$ $mean_{ij} = (u_{ij} + v_{ij})/2$
5	$_5\mu_{ij}(x_j) = \begin{cases} \frac{1}{(d - u_{ij} - v_{ij}) \cdot \{1+\exp[-f_{ij} \cdot (x_j - u_{ij})]\}}, \text{if } x_j < x_{join} \\ \frac{1}{(d - u_{ij} - v_{ij}) \cdot \{1+\exp[-f_{iM+j} \cdot (x_j^c - v_{ij}^c)]\}}, \text{otherwise} \end{cases}$	$_6M_i(\mathbf{x}) = \prod_j {}_5\mu_{ij}(x_j)$ $_7M_i(\mathbf{x}) = \min_j[_5\mu_{ij}(x_j)]$

are also updated according to the δ rule (in this case equivalent to the LMS algorithm)

$$c_{ij}(t+1) = c_{ij}(t) + lr_1 \cdot \frac{M_i(\mathbf{x})}{S(\mathbf{x})} \cdot error \cdot \overline{x}_j$$

where $i = 0, \ldots, N, j = 0, \ldots, M$.

For every rule, there are M possible pairs of fuzzy rules that could result after a potential cut. All the possible consequence parts are trained, in order to obtain tentative performance indices. For the (i)th rule, there are $M \times 2 \times (M+1)$ tentative weights: M possible cuts by two resulting rules by $M+1$ output weights for each rule. These weights are updated in the same manner as the output weights. Finally, the local tentative performance indices and counter are updated.

7) Rule Splitting Procedure: Every P time steps, all committed nodes are checked and the one with the highest local MSE (worst performance) is split, provided that it has been activated at least P times. The best possible cut is determined by the highest tentative performance index. The output weight vectors $\mathbf{c_i}$ and $\mathbf{c_k}$ of the corresponding fuzzy rules are initialized with the values of vector $\mathbf{c_i}$ before the cut. Finally, all relevant local performance indices and counters are reset.

8) Rule Adding: If the mean square error exceeds a threshold equal to $10 \cdot MSE$, an uncommitted node is used, and is initialized. This is actually a safety feature, originating from the fuzzy ARTMAP algorithm [18], [25] having little or no effect in most of the cases.

III. SIMULATION RESULTS

A. Membership Functions and Inference Methods

Five membership functions will be used, combined either with Mamdani's or Larsen's inference methods. The first two membership functions are simple, piecewise linear functions, which use quantities already calculated in the fuzzy ART module. The output of these functions is equal to one when the input vector lies in the associated (crisp) hyperbox and decrease to zero as the distance between the hyperbox and the input vector point increases. This is determined by the values of $|\mathbf{x} \wedge \mathbf{w_i}|$ and $|\mathbf{w_i}|$: if the input vector is contained in the (i)th hyperbox, then $|\mathbf{x} \wedge \mathbf{w_i}| = |\mathbf{w_i}|$, otherwise $|\mathbf{x} \wedge \mathbf{w_i}| < |\mathbf{w_i}|$. The remaining three membership functions are smooth, nonlinear functions, based on the logistic function: $y = 1/[1 + \exp(-x)]$. These functions have higher computational requirements but in some cases yield better results.

All membership functions apart from the first one apply to one dimension. The inference methods used to perform fuzzy reasoning in more than one dimensions, combining these one-dimensional (1-D) membership functions, are Mamdani's method (min) and Larsen's method (algebraic product), resulting in the fuzzy implication membership functions. Product inference usually offered better results. All membership functions and fuzzy implication membership functions used in the simulation are defined in Table I.

1) First Membership Function: The first membership function is the simpler membership function, with the fewer degrees of freedom (DOF). It is similar to the one used in [18] and gives the membership value of each fuzzy implication directly from the fuzzy ART variable values, without the use of any inference method.

2) Second Membership Function: This is a trapezoidal function and is used with both min and product inference. The range of this function is also [0,1]. This membership function has similar computational requirements (in terms of computer time) with the first one, but offers better results, since there are more DOF.

3) Third Membership Function: It is the double logistic function, i.e., two logistic functions joined together forming a bell-like curve. The two logistic functions are allowed to have different slopes. In the case of different slopes, the joining point x_{join} is the point where the

TABLE II
COMPARISON RESULTS: PREVIOUS APPROACHES AND NEUROFAST USING DIFFERENT FUZZY IMPLICATIONS

	Model	Number of rules	identification data	Prediction data	50 data randomly selected				
1	Linear model	-	12.7	11.11	-				
2	FMM [5]	3	1.5	2.1	4.2				
3	SOFIA [7]	4	1.8	2.9	3.4				
4	FNN Type I [14]	8	0.84	1.22	-				
5	FNN Type II [14]	4	0.73	1.28	-				
6	FNN Type III [14]	8	0.63	1.25	-				
	Proposed algorithm membership function	number of rules	identification data	prediction data	data from the whole domain	Percentage of successful runs	stopping criterion	f	lr_3
7	$M(\mathbf{x})$	2	1.22	2.62	4.33	50%	150000	1	1
8	${}_2M(\mathbf{x})$	2	1.47	3.24	3.18	65%	100000	1	1
9	${}_3M(\mathbf{x})$	2	0.76	1.31	2.53	54%	100000	1	1
10	${}_4M(\mathbf{x})$	2	0.90	1.45	3.77	100%	30000	2	100
11	${}_5M(\mathbf{x})$	2	0.79	1.84	6.91	100%	30000	3	100
12	${}_6M(\mathbf{x})$	2	0.71	1.36	3.03	100%	30000	2	100

N=2, M=3. Averaged over 100 runs. $P = 5000$, $= 1.7$, $= 0.01$, $\beta = 0.1$, $\rho = 0$, $lr_1 = 0.1$, $lr_m = 0.0$, $lr_2 = 0.1$.

two logistic functions have equal values. A drawback of this membership function is that it is not normalized, since the logistic function never reaches unity or zero.

4) Fourth Membership Function: In an attempt to normalize the double logistic membership function, each logistic function is divided by its value at the joining point. In this way, both logistic functions attain the value of 1 at this point. The joining point x_{join} is on the middle between the input weights u_{ij} and v_{ij} defining the category ($mean_{ij}$).

5) Fifth Membership Function: If a category is relatively "small," the corresponding membership function should desirably attain relatively "higher" values, in order to have a considerable contribution to the global output value and therefore efficiently learn the function behavior in its area. Following this idea, the fifth function is not exactly a membership function, since its range exceeds unity. This function is the double logistic function divided by a term increasing with the width of the function (see Table I). It is noted that d should be larger than zero.

B. Modeling of a Static Three-Variable Function

A three-variable function is learned from a small set of input–output data. This function was used as a testing example in [5], [7], [14] and is defined as

$$y = (1.0 + x_1^{0.5} + x_2^{-1} + x_3^{-1.5})^2.$$

The system was trained with the same input–output data used in previous works. In order to avoid overtraining, a system with only two rules was used, while convergence was stopped prematurely.

The comparison results appear in Table II. NeuroFAST performs relatively very well, using only two rules. The task is treated as if it were on-line, while most of the previous approaches used off-line, trial-and-error methods. However, it is noted that the piecewise linear nature of the first three fuzzy implication membership functions does not fit well to this task and sometimes prevents the algorithm from reaching a satisfactory performance index.

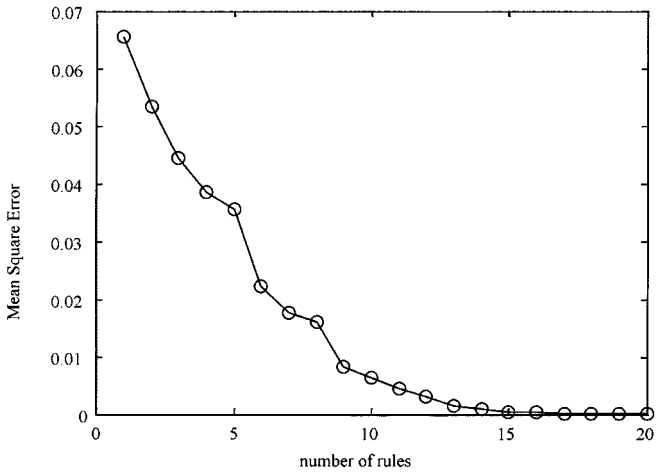

Fig. 1. Mean square error *(MSE)* of the proposed algorithm applied to the Box and Jenkins gas furnace process prediction versus the number of fuzzy rules, using the fuzzy implication ${}_3M(\mathbf{x})$. Each run lasted approximately 50 000 epochs. All learning parameters are kept fixed, except for the vigilance ρ, which increases by the number of fuzzy rules: $\alpha = 0.001$, $\beta = 0.001$, $\rho \in [0, 0.85]$, $lr_1 = 0.5$, $lr_2 = 0.1$, and $lr_3 = 100$.

C. Box and Jenkins Gas Furnace Process Modeling

This is a common benchmark for testing system identification techniques. The data are from a furnace, where air and methane are combined. The input feed rate of methane and the concentration of CO_2 in the output gases are sampled, giving 296 data pairs, which can be found in [21]. This is a dynamical process with one input $x(t)$ and one output $y(t)$. The aim is to predict current output $y(t)$ using past input and output values, with the lowest mean square error.

As in some of the previous approaches, the following six input variables were used: $x(t-1)$, $x(t-2)$, $x(t-3)$, $y(t-1)$, $y(t-2)$, and $y(t-3)$.

In Fig. 1 is the mean square error obtained versus the number of fuzzy rules, using the fuzzy implication membership function

TABLE III
COMPARISON RESULTS FOR THE BOX AND JENKINS GAS FURNACE PROCESS IDENTIFICATION [21]

Model	Number of inputs	Number of rules	Mean Square Error
Box and Jenkins [21]	6	-	0.202
Tong [27]	2	19	0.469
Pedrycz [28]	2	81	0.320
Xu [29]	2	25	0.328
Sugeno and Yasukawa [13]	3	6	0.190
Sugeno and Tanaka [6]	6	2	0.068
Wang and Langari [30]	6	2	0.066
Zikidis and Vasilakos [31]	6	2	0.064
Lin and Cunningham [16]	5	4	0.071
Kim *et al.* [17]	6	2	0.055
Tzafestas and Zikidis [9]	6	2	0.049
ANFIS [8] taken from [33]	2	25	0.00073
Kim and Kasabov [33]	2	15	0.00042
NeuroFAST	6	1	0.06544
NeuroFAST	6	15	0.00040
NeuroFAST	6	20	0.00001

${}_3M(\mathbf{x})$. Comparison results with previous approaches are provided in Table III. It is noted that NeuroFAST with 20 rules attains the best performance reported up to now. It is also worth noting that even with one rule (6-input linear system) it outperforms many approaches with more rules.

IV. CONCLUSION

A new method was proposed for on-line structure and parameter learning of a functional reasoning fuzzy system. Structure identification is executed by a fuzzy ART module. Specific fuzzy rule splitting and adding procedures, provide better coverage of "difficult" areas of the input space. Premise and consequent parameters are fine tuned by the use of the δ rule. Simulation results demonstrate the remarkable capabilities of the proposed method.

Future work will be dealing with a metalearning scheme for automatic adjustment of the learning parameters employed in this algorithm.

APPENDIX

Consider the cost function

$$E(\mathbf{W}) = \tfrac{1}{2}(y - output)^2 = \tfrac{1}{2}error^2$$

where y and *output* are the current desired and actual output of the system and $\mathbf{W}$ is a generalized vector containing all free parameters of the learning process (in our case it should contain $\mathbf{w}$, $\mathbf{c}$, and $\mathbf{f}$). The aim is to iteratively minimize the cost function $E(\mathbf{W})$ over the whole input space. According to the δ rule, in order to perform gradient descent, the change to each parameter W_k should be proportional to the negative of the gradient of $E(\mathbf{W})$ with respect to W_k

$$\Delta\mathbf{W} = -lr \cdot \nabla_{\mathbf{W}} E(\mathbf{W}) \quad \text{or} \quad \Delta W_k = -lr \cdot \frac{\partial E(\mathbf{W})}{\partial W_k}$$

where lr is the learning rate.

Using this rule, deriving the updating equations of the output weights is relatively easy. However, the updating equations of the premise parameters (input weights $\mathbf{w}$ and slope values $\mathbf{f}$) are more complex and depend on the membership functions and the inference method. The study of the input weights updating equation will be made using the fuzzy implication ${}_3M(\mathbf{x})$, which is a characteristic example and provides flexibility (many adaptive parameters, i.e., DOF and low computational overhead (no nonlinear calculations).

For each variable j, the weights w_{ij} and w_{iM+j} and the slopes f_{ij} and f_{iM+j} define the membership function associated with this variable. All these parameters should be positive. Besides, $w_{ij} \leq 1 - w_{iM+j}$. Therefore, the δ rule should not be allowed to make an update that would violate any of these two requirements. Furthermore, for this fuzzy implication, as well as for ${}_1M(\mathbf{x})$ and ${}_2M(\mathbf{x})$, the premise parameters are changed only if the current input vector $\mathbf{x}$ does not lie in the associated hyperbox, since inside the hyperbox these functions are constant. Hence, updating takes place only if $|\mathbf{x} \wedge \mathbf{w_i}| < |\mathbf{w_i}|$.

Considering w_{ij}, we use the chain rule

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial output} \cdot \frac{\partial output}{\partial {}_3M_i(\mathbf{x})} \cdot \frac{\partial {}_3M_i(\mathbf{x})}{\partial {}_2\mu_{ij}} \cdot \frac{\partial {}_2\mu_{ij}}{\partial w_{ij}}.$$

To calculate $\partial E/\partial w_{ij}$, we consider each term of the second part. From the cost function

$$\frac{\partial E}{\partial output} = (y - output) \cdot (-1) = -error.$$

As mentioned earlier, the output of the algorithm is

$$output = \frac{\sum_{i,\text{ rule } i \text{ active}} M_i(\mathbf{x}) \cdot y_i}{\sum_{i,\text{ rule } i \text{ active}} M_i(\mathbf{x})} = \frac{\sum_{i,\text{ rule } i \text{ active}} M_i(\mathbf{x}) \cdot y_i}{S(\mathbf{x})}.$$

Consequently

$$\begin{aligned}\frac{\partial output}{\partial {}_3M_i(\mathbf{x})} &= \frac{\partial\left[\dfrac{\sum_{k,\text{ rule } k \text{ active}} {}_3M_k(\mathbf{x}) \cdot y_k}{S(\mathbf{x})}\right]}{\partial {}_3M_i(\mathbf{x})} \\ &= \frac{S(\mathbf{x}) \cdot y_i - \sum_{k,\text{ rule } k \text{ active}} {}_3M_k(\mathbf{x}) \cdot y_k}{S(\mathbf{x})^2} \\ &= \frac{y_i - \dfrac{\sum_{k,\text{ rule } k \text{ active}} {}_3M_k(\mathbf{x}) \cdot y_k}{S(\mathbf{x})}}{S(\mathbf{x})} \\ &\Rightarrow \frac{\partial output}{\partial {}_3M_i(\mathbf{x})} = \frac{y_i - output}{S(\mathbf{x})}.\end{aligned}$$

Considering the term $\partial {}_3M_i(\mathbf{x})/\partial {}_2\mu_{ij}$

$$\begin{aligned}\frac{\partial {}_3M_i(\mathbf{x})}{\partial {}_2\mu_{ij}(x_j)} &= \frac{\partial[{}_2\mu_{i1}(x_1) \cdot \cdots \cdot {}_2\mu_{ij}(x_j) \cdot \cdots \cdot {}_2\mu_{iM}(x_M)]}{\partial {}_2\mu_{ij}(x_j)} \\ &= {}_2\mu_{i1}(x_1) \cdot \cdots \cdot {}_2\mu_{i(j-1)}(x_{j-1}) \\ &\quad \cdot {}_2\mu_{i(j+1)}(x_{j+1}) \cdot \cdots \cdot {}_2\mu_{iM}(x_M) \\ &= \frac{{}_3M_i(\mathbf{x})}{{}_2\mu_{ij}(x_j)}.\end{aligned}$$

Finally, from the definition of ${}_2\mu_{ij}(x_j)$ (Table I)

$$\partial {}_2\mu_{ij}(x_j)/\partial w_{ij} = -f_{ij}.$$

Combining all the above results

$$\frac{\partial E}{\partial w_{ij}} = -error \cdot \frac{y_i - output}{S(\mathbf{x})} \cdot \frac{{}_3M_i(\mathbf{x})}{{}_2\mu_{ij}(x_j)} \cdot (-f_{ij}).$$

Therefore, the input weight updating rule is

$$\Delta w_{ij} = -lr_2 \cdot error \cdot \frac{y_i - output}{S(\mathbf{x})} \cdot \frac{{}_3M_i(\mathbf{x})}{{}_2\mu_{ij}(x_j)} \cdot f_{ij}.$$

For every input variable, only one "side" of the corresponding membership function is updated: if $x_j < u_{ij}$ the following two equations apply directly, otherwise if $x_j^c < v_{ij}^c$, j should be replaced by $M + j$ in the indices of x, w, and f.

Following the same lines for f_{ij}, we observe that the only difference is that the term $\partial_2 \mu_{ij}(x_j)/\partial f_{ij}$ is used instead of $\partial_2 \mu_{ij}(x_j)/\partial w_{ij}$, giving the rule

$$\Delta f_{ij} = -lr_3 \cdot error \cdot \frac{y_i - output}{S(\mathbf{x})} \cdot \frac{{}_3M_i(\mathbf{x})}{{}_2\mu_{ij}(x_j)} \cdot (w_{ij} - x_j).$$

Using these guidelines, one can obtain the updating equations for each fuzzy implication, which can also be found in [9].

REFERENCES

[1] L. A. Zadeh, "Fuzzy sets," *Inform. Control*, vol. 8, pp. 338–352, 1965.

[2] S. Mitra and Y. Hayashi, "Neuro-fuzzy rule generation: Survey in soft computing framework," *IEEE Trans. Neural Networks*, vol. 11, pp. 748–767, May 2000.

[3] E. H. Mamdani and S. Assilian, "Applications of fuzzy algorithms for control of simple dynamic plant," *Proc. Inst. Elec. Eng.*, vol. 121, pp. 1585–1588, 1974.

[4] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans Syst., Man, Cybern.*, vol. SMC-15, pp. 116–132, Jan./Feb. 1985.

[5] M. Sugeno and G. T. Kang, "Structure identification of fuzzy model," *Fuzzy Sets Syst.*, vol. 28, pp. 15–33, 1988.

[6] M. Sugeno and K. Tanaka, "Successive identification of a fuzzy model and its applications to prediction of a complex system," *Fuzzy Sets Syst.*, vol. 42, pp. 315–334, 1991.

[7] K. Tanaka, M. Sano, and H. Watanabe, "Modeling and control of carbon monoxide concentration using a neuro-fuzzy technique," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 271–279, June 1995.

[8] J. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 665–685, Mar. 1993.

[9] S. G. Tzafestas and K. C. Zikidis, "Fuzzy and neuro-fuzzy systems in modeling, control and robot path planning: An on-line self constructing fuzzy modeling architecture based on neural and fuzzy concepts and techniques," in *Soft Computing in Systems and Control Technology*, S. G. Tzafestas, Ed. Singapore: World Scientific, 1999, pp. 119–168.

[10] K. Watanabe *et al.*, "Mean-value-based functional reasoning and its realization as a fuzzy-neural-network controller," in *Proc. IEEE First Asian Control Conf.*, vol. 3, Tokyo, Japan, Aug. 1994, pp. 435–438.

[11] ——, "Fuzzy-neural network controllers using mean-value-based functional reasoning," *Neurocomput.*, vol. 9, pp. 39–61, 1995.

[12] K. Watanabe and S. G. Tzafestas, "Mean-value-based functional reasoning approach to neural-fuzzy control system design," in *Trends in Control Systems Technology*, C. Leondes, Ed. New York: Academic, 1997.

[13] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 7–31, Feb. 1993.

[14] S. Horikawa, T. Furuhashi, and Y. Uchikawa, "On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm," *IEEE Trans. Neural Networks*, vol. 3, pp. 801–806, Oct. 1992.

[15] C.-T. Sun, "Rule-base structure identification in an adaptive-network-based fuzzy inference system," *IEEE Trans. Fuzzy Syst.*, vol. 2, pp. 64–73, Feb. 1994.

[16] Y. Lin and G. A. Cuningham, III, "A new approach to fuzzy-neural system modeling," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 190–198, Apr. 1995.

[17] E. Kim *et al.*, "A new approach to fuzzy modeling," *IEEE Trans. Fuzzy Syst.*, vol. 5, pp. 328–337, June 1997.

[18] S. G. Tzafestas and K. C. Zikidis, "An on-line learning, neuro-fuzzy architecture, based on functional reasoning and fuzzy ARTMAP," in *Proc. ICSC Int. Symp. Fuzzy Logic Applicat.*, Zurich, Switzerland, 1997.

[19] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, pp. 759–771, 1991.

[20] S. G. Tzafestas, S. Raptis, and G. Stamou, "A flexible neurofuzzy cell structure for general fuzzy inference," *Math. Comp. Simul.*, vol. 41, no. 3/4, pp. 219–233, 1996.

[21] G. E. P. Box and G. M. Jenkins, *Time Series Analysis, Forecasting, and Control*. San Francisco: Holden Day, 1970.

[22] C.-T. Lin, C. -J. Lin, and C. S. G. Lee, "Fuzzy adaptive learning control network with on-line neural learning," *Fuzzy Sets Syst.*, vol. 71, pp. 25–45, 1995.

[23] C.-J. Lin and C.-T. Lin, "Reinforcement learning for an ART-based fuzzy adaptive learning control network," *IEEE Trans. Neural Networks*, vol. 7, pp. 709–731, June 1996.

[24] C.-J. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[25] G. A. Carpenter *et al.*, "Fuzzy ARTMAP: A neural architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. Neural Networks*, vol. 3, pp. 698–712, Oct. 1992.

[26] J. J. Buckley, "Sugeno type controllers are universal controllers," *Fuzzy Sets Syst.*, vol. 53, pp. 299–303, 1993.

[27] R. M. Tong, "The evaluation of fuzzy models derived from experimental data," *Fuzzy Sets Syst.*, vol. 4, pp. 1–12, 1980.

[28] W. Pedrycz, "An identification algorithm in fuzzy relational systems," *Fuzzy Sets Syst.*, vol. 13, pp. 153–167, 1984.

[29] C. Xu and Z. Yong, "Fuzzy model identification and self-learning for dynamic systems," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-17, pp. 683–689, Apr. 1987.

[30] L. Wang and R. Langari, "Building Sugeno-type models using fuzzy discretization and orthogonal parameter estimation techniques," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 454–458, Aug. 1995.

[31] K. C. Zikidis and A. V. Vasilakos, "A.S.A.F.ES.2: A novel, neuro-fuzzy architecture for fuzzy computing, based on functional reasoning," *Fuzzy Sets Syst.*, vol. 83, pp. 63–84, 1996.

[32] S. Barada and H. Singh, "Generating optimal adaptive fuzzy-neural models of dynamical systems with applications to control," *IEEE Trans. Syst., Man, Cybern. C*, vol. 28, pp. 371–391, Aug. 1998.

[33] J. Kim and N. Kasabov, "HyFIS: Adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems," *Neural Networks*, vol. 12, pp. 1301–1319, 1999.

H_∞ Control of Uncertain Dynamical Fuzzy Discrete-Time Systems

S. G. Cao, N. W. Rees, and G. Feng

***Abstract*—A new kind of dynamical fuzzy model is proposed to represent discrete-time complex systems which include both linguistic information and system uncertainties. A new stability analysis and control system design approach is then developed for this kind of dynamical fuzzy model. Furthermore, a constructive algorithm is developed to obtain the H_∞ feedback control law. An example is given to illustrate the application of the method.**

***Index Terms*—Control theory, fuzzy control system design, fuzzy systems.**

I. INTRODUCTION

Recently, there have been a number of applications of fuzzy systems theory in the control field. In most of these applications, the main design objective is to construct a fuzzy model to approximate a desired

Manuscript received October 7, 1995; revised May 31, 2001.

S. G. Cao and N. W. Rees are with the School of Electrical Engineering, University of New South Wales, Sydney, Australia.

G. Feng is with the Department of MEEM, City University of Hong Kong, Kowloon, Hong Kong (e-mail: megfeng@cityu.edu.hk).

Publisher Item Identifier S 1083-4419(01)08543-0.