Int. J. Appl. Math. and Comp. Sci., 1999, Vol.9, No.4, 899–919

ITERATIVE CONSTRUCTION AND OPTIMIZATION OF FUZZY MODELS

NOUREDDINE GOLÉA*, KHIER BENMAHAMMED**

In this paper, a constructive approach to the fuzzy model selection problem is developed. First, the selection of membership functions is decoupled from parameter calculations using an orthogonalization procedure. Since each membership function depends only on its own parameters, the selection of rules is performed in a sequential manner. At each learning step, a new membership function is created and its parameters are optimized. The resulting parameter calculation boils down to the solution of a triangular system. This approach reduces significantly the computational complexity, and allows for the derivation of a simple optimization algorithm. In addition, optimization of the membership functions is related to the approximation accuracy. Simulation results, when compared with the orthogonal least-squares algorithm, show that this approach is less sensitive to the size of the training data and converges rapidly.

Keywords: fuzzy approximation, decoupled learning, iterative optimization

1. Introduction

Fuzzy systems have recently found extensive applications in a wide variety of domains (see e.g. Lee, 1992 and references therein). This is essentially due to two main features: first, fuzzy systems allow for simple inclusion of qualitative knowledge and, second, they do not require the existence of an analytical model of the system. In most applications of fuzzy systems, the main design objective can be transformed to the search for a desired model from the input space to the output space. Thus, the problem of designing fuzzy models can be viewed as a function approximation one. Recently, the approximation capability has been proved for a large class of fuzzy systems (Kosko, 1992; Wang and Mendel, 1993; Ying, 1994), i.e. we have learnt that they can approximate any continuous function on a compact set to an arbitrary degree of accuracy. Although at first sight this result appears to be attractive, it does not provide much insight into the following practical problem: for a prescribed accuracy on a given compact space, which fuzzy model structure and parameters achieve the best approximation? In fact, the above result was obtained without any condition on the fuzzy model structure. Conceptually, a fuzzy model with few rules will be unable

* Electrical Engineering Institute, University of Oum-El-Bouaghi, 04000 Oum-El-Bouaghi, Algeria, e-mail: ngolea@mailcity.com

** Electronics Institute, University of Sétif, 19000 Sétif, Algeria

to capture the underlying function from which the data were generated, while a fuzzy model with too many rules will fit the noise in the data and again result in a poor approximation of the underlying function. To tackle this problem, various learning schemes have been proposed over the last years, which attempt generally to realize two major tasks:

i) Selection of the appropriate fuzzification for input/output spaces and the number of rules, i.e. structure selection.

ii) Selection of a set of free parameters that would maximize the fuzzy model performance for a given approximation problem.

A conventional approach to this problem is to cover uniformly the input space with membership functions and to use linear optimization techniques to find the missing parameters. However, the cost of using linear algorithms and unimodal error surfaces is the need for an exponentially increasing number of rules in high-dimensional spaces, and therefore this approach is only advised in lower-dimensional spaces. In (Horikawa *et al.*, 1992; Wang, 1995) nonlinear optimization algorithms are used to learn the parameters of a fixed *a-priori* fuzzy model structure. The empirically chosen structure may be suboptimal, and many trials may be necessary to find the appropriate one. Another possible approach is to realize a self-organizing choice of appropriate rules based on an unsupervised clustering algorithm (Lin and Cunningham, 1995; Euntai *et al.*, 1997; Juang and Lin, 1998). This allows the membership functions to be grouped in statistically important input/output data regions. The free parameters are then determined using a supervised learning algorithm. However, the membership functions found in an unsupervised manner are not always optimal, and the supervised learning algorithm cannot fully exploit the complete set of functions that the fuzzy model is capable of implementing. New approaches include techniques that select the parameters of the appropriate membership functions as a subset of the training samples. In (Wang and Mendel, 1992; Hohensohn and Mendel, 1996), the orthogonal least-squares (OLS) algorithm and an incremental index are used to select the appropriate rules. The orthogonality of the rules allows the contribution of each rule to the approximation to be calculated independently. However, the performance of the OLS algorithm is very sensitive to the training data size, and the set of selected rules cannot be larger than the training set. Other approaches use structure identification and clustering algorithms to select the rules. They either start with one rule and add new rules as needed, or start with a large number of rules and perform the selection by pruning unnecessary rules (e.g. Sugeno and Kang, 1988; Yoshinari *et al.*, 1993; Nozaki *et al.*, 1996). Recently, a good performance for a small-sized fuzzy model has been achieved by selecting the rules using genetic algorithms (Karr, 1991; Homaifar and McCormick, 1995).

In this work, the fuzzy model construction is formulated as a nonlinear least-squares problem depending on both the membership functions and resulting parameters. Because the parameters enter the squared error criterion in a quadratic manner, their calculation is decoupled from the membership function optimization using the generalized Gram-Schmidt orthogonalization. As each membership function depends

on its own parameters, the appropriate membership functions are learnt sequentially. At each step, a new membership function is added and its parameters are optimized, until a required level of accuracy is obtained. This approach provides two advantages: First, membership function creation and optimization are related to the approximation performance, i.e. the fuzzy model complexity is controlled by the specified approximation accuracy. Second, this approach provides a suitable solution to the nonlinear optimization problem, since the number of the parameters to be optimized at each step is independent of the fuzzy model size. The simulation results for the gas furnace data prediction and the control of a complex nonlinear system, reveal that this approach reduces significantly the time and storage requirements, and exhibits good performance for small training data size.

2. Problem Statement

2.1. Fuzzy Models

In the following we assume that fuzzy models are multi-input single-output systems: $X \longmapsto Y$, where $X = X_1 \times X_2 \times \cdots \times X_n \subset \mathbb{R}^n$ is the input space and $Y \subset \mathbb{R}$ is the output space. A multi-output model can always be separated into a set of single-output models.

Consider a fuzzy model (see Fig. 1) which consists of four principal components: fuzzifier, fuzzy rule base, fuzzy inference engine, and defuzzifier. The rule base consists of the fuzzy rules in the following form:

$$R_i\colon \mathit{If}\ u_1\ \mathit{is}\ A_{i1}\ \mathit{and}\ u_2\ \mathit{is}\ A_{i2}\ \mathit{and}\ \ldots\ \mathit{and}\ u_n\ \mathit{is}\ A_{in}\ \mathit{Then}\ v\ \mathit{is}\ B_i$$

or

$$R_i\colon \mathit{If}\ \boldsymbol{u}\ \mathit{is}\ A_i\ \mathit{Then}\ v\ \mathit{is}\ B_i, \tag{1}$$

where $\boldsymbol{u} = (u_1, u_2, \ldots, u_n) \in X$ and $v \in Y$ are linguistic variables. Their numerical values are $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in X$ and $y \in Y$, respectively. A_{ij} $(i = 1, \ldots, M;\ j = 1, \ldots, n)$ in X_j and B_i $(i = 1, \ldots, M)$ in Y are fuzzy sets characterized by the membership functions $A_{ij}(x_j)$ and $B_i(y)$, respectively. A_i is the fuzzy set in X describing the input vector $\boldsymbol{x}$ with the membership function

$$A_i(\boldsymbol{x}) = A_{i1}(x_1) \star A_{i2}(x_2) \star \cdots \star A_{in}(x_n), \tag{2}$$

where $\star$ is a T-norm. The most common T-norms are 'product' and 'min' (Lee, 1992). Each rule R_i can be viewed as a fuzzy implication (relation) $R_i : A_i \mapsto B_i$, which is a fuzzy set in $X \times Y$ with the membership function

$$R_i(\boldsymbol{x}, y) = A_i(\boldsymbol{x}) \star B_i(y). \tag{3}$$

The fuzzy inference engine is a decision-making logic that employs fuzzy rules from the rule base to determine a mapping from the fuzzy sets in the input space X

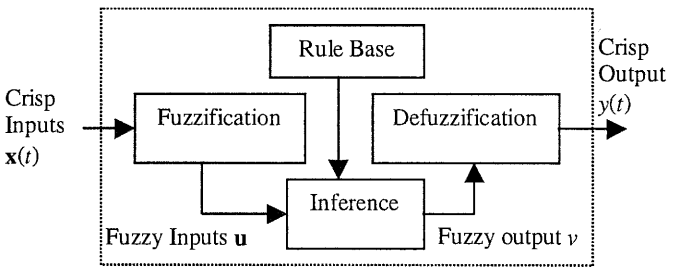


Fig. 1. Fuzzy system components.

to the fuzzy sets in the output space Y. Let A be an arbitrary fuzzy set in X. Then each rule R_i of (1) determines a fuzzy set $Y_{A \circ R_i}$ in Y based on sup-star composition

$$Y_{A \circ R_i}(y) = \sup_{x \in X} \left\{ A(x) \star R_i(x, y) \right\}. \tag{4}$$

If we choose $\star$ in(2) and (3) to be an algebraic product, (4) becomes

$$Y_{A \circ B_i}(y) = \sup_{x \in X} \left\{ A(x) A_i(x) B_i(y) \right\}. \tag{5}$$

The fuzzifier maps a crisp point x into a fuzzy set A in X. In general, there are two possible choices of this mapping, namely a singleton or a nonsingleton. In this work, we use the singleton fuzzifier mapping, i.e. $A(x') = 1$ if $x' = x$ and $A(x') = 0$ if $x' \neq x$.

The defuzzifier performs a mapping from the fuzzy sets in Y to crisp points in Y. Here, we choose the defuzzifier to be the weighted-sum defuzzifier, which maps the fuzzy set (5) to a crisp point

$$y = \sum_{i=1}^{M} w_i Y_{A \circ R_i}(w_i), \tag{6}$$

where w_i is the point in Y at which $B_i(y)$ achieves its maximum.

The choice of membership functions is quite subjective, but if their parameters are to be optimized, they must be differentiable. From the approximation theory point of view (Poggio and Girosi, 1990), a Gaussian membership function is a good choice. In this paper, we use the following Gaussian membership function:

$$A_{ij}(x) = \exp\left(- \rho_{ij}^2 \left(x_j(t) - c_{ij} \right)^2 \right). \tag{7}$$

The fuzzy system with the Gaussian membership function (7), the product inference (5), the singleton fuzzifier, and the weighted-sum defuzzifier (5) is of the following form:

$$y = \sum_{i=1}^{M} \exp\left(- \sum_{j=1}^{n} \rho_{ij}^2 \left(x_j(t) - c_{ij} \right)^2 \right) w_i. \tag{8}$$

2.2. Approximation Problem

According to the universal approximation property, a given set of inputs $\boldsymbol{x}(t) = (x_1 t), x_2(t), \ldots, x_n(t))$ and the corresponding desired output $d(t)$ can be approximated, to any chosen accuracy, by a 'correctly constructed' fuzzy model, such that

$$d(t) = \sum_{i=1}^{M} A_i\big(\boldsymbol{x}(t), \boldsymbol{\theta}_i\big) w_i + e(t), \tag{9}$$

where $\boldsymbol{\theta}_i = (c_{i1}, \ldots, c_{in}, \rho_{i1}, \ldots, \rho_{in})^T$ is the parameter vector controlling the shape and the location of the membership function $A_i(\cdot)$, and $e(t)$ is the modeling error at sample t.

For $t = 1, \ldots, N$, (9) can be written in the following matrix form:

$$\boldsymbol{e} = \boldsymbol{d} - \boldsymbol{\Phi}\,(\boldsymbol{\Theta})\,\boldsymbol{w}, \tag{10}$$

where

$$\boldsymbol{e} = \left[\; e(1) \quad e(2) \quad \cdots \quad e(N) \;\right]^T, \tag{11}$$

$$\boldsymbol{d} = \left[\; d(1) \quad d(2) \quad \cdots \quad d(N) \;\right]^T, \tag{12}$$

$$\boldsymbol{w} = \left[\; w_1 \quad w_2 \quad \cdots \quad w_M \;\right]^T, \tag{13}$$

$$\boldsymbol{\Phi}(\boldsymbol{\Theta}) = \left[\; \boldsymbol{\phi}_1(\boldsymbol{\theta}_1) \quad \boldsymbol{\phi}_2(\boldsymbol{\theta}_2) \quad \cdots \quad \boldsymbol{\phi}_M(\boldsymbol{\theta}_M) \;\right], \tag{14}$$

with

$$\boldsymbol{\phi}_i(\boldsymbol{\theta}_i) = \left[\; A_i\big(\boldsymbol{\theta}_i, \boldsymbol{x}(1)\big) \quad A_i\big(\boldsymbol{\theta}_i, \boldsymbol{x}(2)\big) \quad \cdots \quad A_i\big(\boldsymbol{\theta}_i, \boldsymbol{x}(N)\big) \;\right]^T \tag{15}$$

and

$$\boldsymbol{\Theta} = \big(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_M\big). \tag{16}$$

The squared-error criterion is given by

$$J(\boldsymbol{\Theta}, \boldsymbol{w}) = \big[\boldsymbol{d} - \boldsymbol{\Phi}(\boldsymbol{\Theta})\boldsymbol{w}\big]^T \big[\boldsymbol{d} - \boldsymbol{\Phi}(\boldsymbol{\Theta})\boldsymbol{w}\big]. \tag{17}$$

The objective is to minimize (17) by a proper choice of the fuzzy model structure and parameters. Since the resulting parameters $\boldsymbol{w}$ enter (17) in a quadratic form, their computation can be decoupled from the selection of the membership functions $A_i(\boldsymbol{\theta}_i)$ (represented by the vector $\boldsymbol{\phi}_i(\boldsymbol{\theta}_i)$). To perform such decoupling, the algorithm developed for nonlinear separation of variables (Golub and Pereyra, 1973) will be adopted.

3. Decoupling Algorithm

In order to decouple $J(\boldsymbol{\Theta}, \boldsymbol{w})$, the concept of orthogonal vectors is used (Schilling and Lee, 1988). A given vector $\boldsymbol{d} \in \mathbb{R}^N$ can be uniquely decomposed as

$$\boldsymbol{d} = \boldsymbol{d}_1 + \boldsymbol{d}_2, \tag{18}$$

where $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ are orthogonal vectors with the following inner-product property:

$$\boldsymbol{d}_1^T \boldsymbol{d}_2 = 0, \tag{19}$$

$\boldsymbol{d}_1$ being a unique vector contained in the range space spanned by $\boldsymbol{\Phi}(\boldsymbol{\Theta})$ and $\boldsymbol{d}_2$ being a unique vector that is orthogonal to it. The determination of the orthogonal vectors involves the computation of the $N \times N$ projection matrix $\boldsymbol{P}(\boldsymbol{\Theta})$ which spans the same space of $\boldsymbol{\Phi}(\boldsymbol{\Theta})$. It can be shown that $\boldsymbol{P}(\boldsymbol{\Theta})$ is given by the following expression:

$$\boldsymbol{P}(\boldsymbol{\Theta}) = \boldsymbol{\Phi}(\boldsymbol{\Theta})\left[\boldsymbol{\Phi}^T(\boldsymbol{\Theta})\boldsymbol{\Phi}(\boldsymbol{\Theta})\right]^{-1}\boldsymbol{\Phi}^T(\boldsymbol{\Theta}). \tag{20}$$

Then the orthogonal vectors in (18) are given by

$$\begin{aligned} \boldsymbol{d}_1 &= \boldsymbol{P}(\boldsymbol{\Theta})\boldsymbol{d}, \\ \boldsymbol{d}_2 &= \left[\boldsymbol{I}_N - \boldsymbol{P}(\boldsymbol{\Theta})\right]\boldsymbol{d}, \end{aligned} \tag{21}$$

and the decomposition of the vector $\boldsymbol{d}$ is given by

$$\boldsymbol{d} = \boldsymbol{P}(\boldsymbol{\Theta})\boldsymbol{d} + \left[\boldsymbol{I}_N - \boldsymbol{P}(\boldsymbol{\Theta})\right]\boldsymbol{d}. \tag{22}$$

A convenient form of the projection matrix $\boldsymbol{P}(\boldsymbol{\Theta})$ is computed using the generalized Gram-Schmidt orthogonalization (Cadzow, 1990), where the $N \times M$ matrix $\boldsymbol{\Phi}(\boldsymbol{\Theta})$ is decomposed as

$$\boldsymbol{\Phi}(\boldsymbol{\Theta}) = \boldsymbol{Q}(\boldsymbol{\Theta})\boldsymbol{R}(\boldsymbol{\Theta}), \tag{23}$$

where

$$\boldsymbol{Q}(\boldsymbol{\Theta}) = \left[\begin{array}{cccc} q_1(\theta_1) & q_2(\theta_1, \theta_2) & \ldots & q_M(\theta_1, \theta_2, \ldots, \theta_M) \end{array}\right] \tag{24}$$

is the $N \times M$ matrix with orthogonal columns, such that

$$\boldsymbol{Q}^T(\boldsymbol{\Theta})\boldsymbol{Q}(\boldsymbol{\Theta}) = \boldsymbol{I}_M \tag{25}$$

and $\boldsymbol{R}(\boldsymbol{\Theta})$ is an $M \times M$ upper triangular matrix.

The matrix $\boldsymbol{P}(\boldsymbol{\Theta})$ is obtained by substitution of (23) in (20), which gives

$$\boldsymbol{P}(\boldsymbol{\Theta}) = \boldsymbol{Q}(\boldsymbol{\Theta})\boldsymbol{Q}^T(\boldsymbol{\Theta}). \tag{26}$$

Substituting (22) in (17) and developing the result, we get

$$\begin{aligned} \boldsymbol{J}(\Theta, w) = &\Big([\boldsymbol{I}_N - \boldsymbol{P}(\Theta)]\boldsymbol{d}\Big)^T [\boldsymbol{I}_N - \boldsymbol{P}(\Theta)]\boldsymbol{d} \\ &+ [\boldsymbol{P}(\Theta)\boldsymbol{d} - \boldsymbol{\Phi}(\Theta)w]^T [\boldsymbol{P}(\Theta)\boldsymbol{d} - \boldsymbol{\Phi}(\Theta)w] \\ &+ [\boldsymbol{P}(\Theta)\boldsymbol{d} - \boldsymbol{\Phi}(\Theta)w]^T [\boldsymbol{I}_N - \boldsymbol{P}(\Theta)]\boldsymbol{d} \\ &+ \Big([\boldsymbol{I}_N - \boldsymbol{P}(\Theta)]\boldsymbol{d}\Big)^T [\boldsymbol{P}(\Theta)\boldsymbol{d} - \boldsymbol{\Phi}(\Theta)w] \end{aligned} \tag{27}$$

using the properties of the projection matrix (i.e. $\boldsymbol{P}(\Theta)^T = \boldsymbol{P}(\Theta)$, $\boldsymbol{P}(\Theta)^2 = \boldsymbol{P}(\Theta)$, and $\boldsymbol{P}(\Theta)\boldsymbol{\Phi}(\Theta) = \boldsymbol{\Phi}(\Theta)$), we can prove that the third and fourth terms in (27) are zero and (27) can be simplified to

$$J(\Theta, w) = \boldsymbol{d}^T [\boldsymbol{I}_N - \boldsymbol{P}(\Theta)]\boldsymbol{d} + [\boldsymbol{P}(\Theta)\boldsymbol{d} - \boldsymbol{\Phi}(\Theta)w]^T [\boldsymbol{P}(\Theta)\boldsymbol{d} - \boldsymbol{\Phi}(\Theta)w]. \tag{28}$$

The first term of (28) denoted by

$$J_{\boldsymbol{\Theta}}(\Theta) = \boldsymbol{d}^T [\boldsymbol{I}_N - \boldsymbol{P}(\Theta)]\boldsymbol{d} \tag{29}$$

is independent of the parameters $\boldsymbol{w}$ and can be minimized by selecting an appropriate set of membership functions. The second term of (28) is in the sequel denoted by

$$J_{\boldsymbol{w}}(w, \Theta) = [\boldsymbol{P}(\Theta)\boldsymbol{d} - \boldsymbol{\Phi}(\Theta)w]^T [\boldsymbol{P}(\Theta)\boldsymbol{d} - \boldsymbol{\Phi}(\Theta)w]. \tag{30}$$

For any selection of the parameters Θ^0, $J_{\boldsymbol{w}}(w, \Theta^0)$ can be made zero using the following selection of $\boldsymbol{w}$:

$$w^0 = \boldsymbol{R}^{-1}(\Theta^0)\boldsymbol{Q}^T(\Theta^0)\boldsymbol{d}. \tag{31}$$

The result (31) is very important, because it indicates that the modelling error depends only on the value of the criterion (29), i.e. on the choice of the membership functions. Thus, the criteria (29) and (30) provide a more convenient form to construct the fuzzy model that best fits the modelled data, by learning independently its appropriate set of membership functions and consequent parameters using a simple learning scheme.

4. Iterative Construction

The fuzzy model construction is achieved in an iterative manner, by decomposing the problem into a sequence of simpler ones which consist in calculating parameters of individual rules. Thus, at each step one membership function is introduced and its parameter vector $\boldsymbol{\theta}_i$ is adjusted to minimize (29). The computational steps of this approach are summarized in the following:

- In the first step, the first membership function $\phi_1(\boldsymbol{\theta}_1)$ (i.e. $A_1(\boldsymbol{\theta}_1)$) is introduced, the orthogonalization procedure provides the projection matrix $\boldsymbol{P}_1(\boldsymbol{\theta}_1)$, and the criterion (29) is given by

$$J_{\boldsymbol{\Theta}}(\boldsymbol{\theta}_1) = \boldsymbol{d}^T [\boldsymbol{I}_N - \boldsymbol{P}_1(\boldsymbol{\theta}_1)]\boldsymbol{d}. \tag{32}$$

- In the second step, the first membership function is fixed as $\phi_1(\theta_1^0)$, and the second membership function $\phi_2(\theta_2)$ is added, which gives $\boldsymbol{P}_2(\theta_2)$ (fixed parameters are omitted where it is convenient), so we get the following criterion:

$$J_{\Theta}(\theta_2) = \boldsymbol{d}^T[\boldsymbol{I}_N - \boldsymbol{P}_1^0]\boldsymbol{d} - \boldsymbol{d}^T\boldsymbol{P}_2(\theta_2)\boldsymbol{d} \tag{33}$$

which can be rewritten as

$$J_{\Theta}(\theta_2) = J_{\Theta 1}^0 - \boldsymbol{d}^T\boldsymbol{P}_2(\theta_2)\boldsymbol{d}. \tag{34}$$

- In the k-th step we introduce the membership function $\phi_k(\theta_k)$. We get $\boldsymbol{P}_k(\theta_k)$ and the following criterion:

$$J_{\Theta}(\theta_k) = J_{\Theta k-1}^0 - \boldsymbol{d}^T\boldsymbol{P}_k(\theta_k)\boldsymbol{d}. \tag{35}$$

Expression (35) provides an iterative form to minimize (29), by considering the optimization of the parameter vector of one membership function at each step. Since the number of the parameters to be optimized during each learning step is the same, the computational cost is relatively constant and does not scale with the fuzzy model size.

- The iterative construction is stopped at the M_a-th step whenever

$$\frac{J_{\Theta}(\theta_{M_a})}{\boldsymbol{d}^T\boldsymbol{d}} < \varepsilon \tag{36}$$

ε being a chosen tolerance parameter. The division by the data energy has a clear normalizing effect on (36), thus the normalized stopping criterion values always lie in the interval $[0, 1]$. Note that any other statistical criterion that provides a compromise between the 'model fit' and the 'model complexity' can be used as the stopping criterion.

5. Parameter Optimization

As the parameter vector θ_k enters (35) in a nonlinear fashion, nonlinear optimization techniques will be used to perform parameter adaptation. At the k-th learning step, the parameter vector θ_k is updated according to the following formula:

$$\theta_k(l+1) = \theta_k(l) - \gamma \boldsymbol{H}(\theta_k, l), \tag{37}$$

where l is the iteration step and γ is the learning step size chosen to guarantee the convergence of the iterative process. The mapping $\boldsymbol{H}(\cdot)$ indicates the search direction, depending on the optimization algorithm used. The steepest descent algorithm involves small computational requirements, but it converges slowly. Newton-type algorithms provide a rapid convergence rate with more computation complexity (Eykhoff, 1979).

If the gradient algorithm is to be used, then

$$\boldsymbol{H}(\boldsymbol{\theta}_k) = \nabla J_k(\boldsymbol{\theta}_k), \tag{38}$$

where $\nabla J_k(\boldsymbol{\theta}_k)$ is the gradient with respect to the parameter vector $\boldsymbol{\theta}_k$ given by

$$\nabla J_k(\boldsymbol{\theta}_k) = -\boldsymbol{d}^T \frac{\partial \boldsymbol{P}_k(\boldsymbol{\theta}_k)}{\partial \boldsymbol{\theta}_k} \boldsymbol{d}. \tag{39}$$

From (39) it is seen that $\boldsymbol{H}(\boldsymbol{\theta}_k)$ involves the computation of the partial derivatives of $\boldsymbol{P}_k(\boldsymbol{\theta}_k)$ with respect to the vector $\boldsymbol{\theta}_k$. By taking the derivative of the two sides of the identity

$$\boldsymbol{P}(\boldsymbol{\theta}_k)\boldsymbol{\Phi}(\boldsymbol{\theta}_k) = \boldsymbol{\Phi}(\boldsymbol{\theta}_k) \tag{40}$$

and arranging the resulting terms, we get

$$\frac{\partial \boldsymbol{P}(\boldsymbol{\theta}_k)}{\partial \boldsymbol{\theta}_k} \boldsymbol{\Phi}(\boldsymbol{\theta}_k) = \left[\boldsymbol{I}_N - \boldsymbol{P}(\boldsymbol{\theta}_k)\right] \frac{\partial \boldsymbol{\Phi}(\boldsymbol{\theta}_k)}{\partial \boldsymbol{\theta}_k}. \tag{41}$$

The general solution to (41) was developed by Golub and Pereyra (1973), but a more useful formula was derived by Kaufman (1975), which offers the same convergence characteristics, where the expression for the required partial derivatives is as follows:

$$\frac{\partial \boldsymbol{P}(\boldsymbol{\theta}_k)}{\partial \boldsymbol{\theta}_k} \approx \left[\boldsymbol{I}_N - \boldsymbol{P}(\boldsymbol{\theta}_k)\right] \frac{\partial \boldsymbol{\Phi}(\boldsymbol{\theta}_k)}{\partial \boldsymbol{\theta}_k} \boldsymbol{\Phi}^+(\boldsymbol{\theta}_k), \tag{42}$$

$\boldsymbol{\Phi}^+(\boldsymbol{\theta}_k) = (\boldsymbol{\Phi}^T(\boldsymbol{\theta}_k)\boldsymbol{\Phi}(\boldsymbol{\theta}_k))^{-1}\boldsymbol{\Phi}^T(\boldsymbol{\theta}_k)$ is the pseudo-inverse of $\boldsymbol{\Phi}(\boldsymbol{\theta}_k)$.

In the sequential learning context, a much simpler form of the required partial derivatives can be derived. For that purpose, identity (40) at the k-th learning step is decomposed as

$$\boldsymbol{P}(\boldsymbol{\theta}_k)\boldsymbol{\phi}_k(\boldsymbol{\theta}_k) = \boldsymbol{\phi}_k(\boldsymbol{\theta}_k) \tag{43}$$

and the projection matrix $\boldsymbol{P}(\boldsymbol{\theta}_k)$ is written in the following iterative form:

$$\boldsymbol{P}(\boldsymbol{\theta}_k) = \sum_{i=1}^{k-1} \boldsymbol{P}_i^0 + \boldsymbol{P}_k(\boldsymbol{\theta}_k). \tag{44}$$

Then the approximate partial derivatives expression is given by

$$\frac{\partial \boldsymbol{P}_k(\boldsymbol{\theta}_k)}{\partial \boldsymbol{\theta}_k} \approx \left[\boldsymbol{I}_N - \boldsymbol{P}(\boldsymbol{\theta}_k)\right] \frac{\partial \boldsymbol{\phi}_k(\boldsymbol{\theta}_k)}{\partial \boldsymbol{\theta}_k} \frac{\boldsymbol{\phi}_k^T(\boldsymbol{\theta}_k)}{\boldsymbol{\phi}_k^T(\boldsymbol{\theta}_k)\boldsymbol{\phi}_k(\boldsymbol{\theta}_k)}. \tag{45}$$

Expression (45) presents, when compared with those already derived, the advantage to be computationally simpler, and its computation is undertaken by using the elements already used in the orthogonalization procedure, i.e. there is no need for computing the inverse of $\boldsymbol{R}$ (Golub and Pereyra, 1973) or the pseudo-inverse of $\boldsymbol{\Phi}$ (Kaufman, 1975). Note that the expression derived for partial derivatives remains valid when the gradient algorithm is replaced by any other nonlinear optimization algorithm, e.g. Gauss-Newton or Marquardt algorithms.

6. Multivariable Case

The foregoing approach can be easily extended to the approximation problem for multivariable systems. Consider the set of inputs $\boldsymbol{x}(t) = (x_1(t), x_2(t), \ldots, x_n(t))$ and the corresponding desired outputs $(d_1(t), d_2(t), \ldots, d_p(t))$ which can be approximated by the following MIMO fuzzy model:

$$d_j(t) = \sum_{i=1}^{M} A_i\big(\boldsymbol{x}(t), \boldsymbol{\theta}_i\big) w_{ij} + e_j(t), \quad j = 1, \ldots, p. \tag{46}$$

Then the sum of squared errors is given by

$$J(\boldsymbol{\Theta}, \boldsymbol{W}) = \sum_{j=1}^{p} \big[\boldsymbol{d}_j - \boldsymbol{\Phi}(\boldsymbol{\Theta})\boldsymbol{w}_j\big]^T \big[\boldsymbol{d}_j - \boldsymbol{\Phi}(\boldsymbol{\Theta})\boldsymbol{w}_j\big], \tag{47}$$

where

$$\boldsymbol{d}_j = \begin{bmatrix} d_j(1) & d_j(2) & \ldots & d_j(N) \end{bmatrix}^T, \tag{48}$$

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{w}_1 & \boldsymbol{w}_2 & \ldots & \boldsymbol{w}_p \end{bmatrix}, \tag{49}$$

and

$$\boldsymbol{w}_j = \begin{bmatrix} w_1 & w_2 & \ldots & w_M \end{bmatrix}^T. \tag{50}$$

The application of the decoupling algorithm to (47) yields the criteria

$$J_{\boldsymbol{\Theta}}(\boldsymbol{\Theta}) = \sum_{j=1}^{p} \boldsymbol{d}_j^T \big[\boldsymbol{I}_N - \boldsymbol{P}(\boldsymbol{\Theta})\big] \boldsymbol{d}_j \tag{51}$$

and

$$J_{\boldsymbol{W}}(\boldsymbol{W}, \boldsymbol{\Theta}) = \sum_{j=1}^{p} \big[\boldsymbol{P}(\boldsymbol{\Theta})\boldsymbol{d}_j - \boldsymbol{\Phi}(\boldsymbol{\Theta})\boldsymbol{w}_j\big]^T \big[\boldsymbol{P}(\boldsymbol{\Theta})\boldsymbol{d}_j - \boldsymbol{\Phi}(\boldsymbol{\Theta})\boldsymbol{w}_j\big]. \tag{52}$$

The minimization of (51) is performed in a similar manner by iterative construction and optimization of the membership functions. At the k-th learning step, the criterion to be minimized is given by

$$J_{\boldsymbol{\Theta}}(\boldsymbol{\theta}_k) = J^0_{\boldsymbol{\Theta}_{k-1}} - \sum_{j=1}^{p} \boldsymbol{d}_j^T \boldsymbol{P}(\boldsymbol{\theta}_k) \boldsymbol{d}_j \tag{53}$$

and the construction of the fuzzy model is stopped when

$$\frac{J_{\boldsymbol{\Theta}}(\boldsymbol{\theta}_{Ma})}{\sum_{j=1}^{p} \boldsymbol{d}_j^T \boldsymbol{d}_j} < \varepsilon. \tag{54}$$

For the selected membership functions (i.e. $\boldsymbol{\Theta}^0$), the criterion (52) is made zero by choosing

$$\boldsymbol{w}_j^0 = \boldsymbol{R}^{-1}(\boldsymbol{\Theta}^0) \boldsymbol{Q}^T(\boldsymbol{\Theta}^0) \boldsymbol{d}_j, \quad j = 1, \ldots, p. \tag{55}$$

Thus, the proposed approach can be extended to perform the approximation of multivariable systems with essentially no major modifications.

7. Simulation

7.1. Modelling the Gas Furnace Data

In this subsection, Box and Jenkins gas furnace data are used (Box and Jenkins, 1970). The data consist of 296 input/output measurements of a gas furnace system: the input measurement $u(t)$ is the gas flow rate into the furnace and the output measurement is the CO_2 concentration in the outlet gas.

The objective is to contrast the performance of the constructive and OLS algorithms to build fuzzy models (FM) that can predict the gas furnace data, such that

$$\widehat{y}(t) = \mathrm{FM}\big[\boldsymbol{x}(t), \boldsymbol{\Theta}, \boldsymbol{w}\big], \tag{56}$$

where the input vector $\boldsymbol{x}(t)$ is

$$\boldsymbol{x}(t) = \big(y(t-1), y(t-2), y(t-3), u(t), u(t-1), u(t-2)\big). \tag{57}$$

Two fuzzy models of the same complexity ($M_a = 20$) were built using the constructive and OLS algorithms. The fuzzy models are of the form (8), i.e. with product inference, singleton fuzzifier and weighted-sum defuzzifier. To simplify the computation, the width is set to $\rho = 0.05$ for all the membership functions.

The steps of the constructive algorithm are as follows:

1. At the k-th step, introduce a membership function $\boldsymbol{\phi}_k(\boldsymbol{\theta}_k)$, where $\boldsymbol{\theta}_k$ is chosen randomly from the training data.

2. Using the orthogonalization procedure, compute the projection matrix $\boldsymbol{P}_k(\boldsymbol{\theta}_k)$.

3. Update the vector parameters $\boldsymbol{\theta}_k$ using (37) and (45), until $(J_{\boldsymbol{\Theta}}(\boldsymbol{\theta}_k(l)) - J_{\boldsymbol{\Theta}}(\boldsymbol{\theta}_k(l-1))) \leq \epsilon_1$.

4. If $(J_{\boldsymbol{\Theta}}(\boldsymbol{\theta}_k)/\boldsymbol{Y}^T\boldsymbol{Y}) \leq \epsilon_2$ and/or $k = M_a$, then stop, else set $k = k+1$ and go to Step 2.

5. Compute the consequent parameters using (31).

As regards the notation, ϵ_1 and ϵ_2 are chosen tolerance parameters, and $\boldsymbol{Y}$ is the vector of the training outputs.

The OLS algorithm consists of the following steps:

1. At the first step ($k = 1$), based on the training data, construct a set of M membership functions: $\boldsymbol{\Phi}(\boldsymbol{\Theta}) = [\boldsymbol{\phi}_1(\boldsymbol{\theta}_1) \;\; \boldsymbol{\phi}_2(\boldsymbol{\theta}_2) \;\; \ldots \;\; \boldsymbol{\phi}_M(\boldsymbol{\theta}_M)]$ such that $\mathrm{rank}(\boldsymbol{\Phi}(\boldsymbol{\Theta})) = M$.

2. Using the orthogonalization procedure, compute the projection matrix $\boldsymbol{P}(\boldsymbol{\Theta})$.

3. For the set of membership functions, compute the following index:

$$\text{Ind}_i = \frac{\boldsymbol{Y}^T \boldsymbol{P}_i \boldsymbol{Y}}{\boldsymbol{Y}^T \boldsymbol{Y}}, \quad i = 1, \ldots, M.$$

4. Select a membership function $\phi_{i_0}(\boldsymbol{\theta}_{i_0})$ with the largest index Ind_{i_0}.

5. Remove $\phi_{i_0}(\boldsymbol{\theta}_{i_0})$ from the matrix $\boldsymbol{\Phi}(\boldsymbol{\Theta})$ and set $M = M - 1$.

6. If $(J_{\boldsymbol{\Theta}}(\boldsymbol{\theta}_k)/\boldsymbol{Y}^T\boldsymbol{Y}) \leq \epsilon_2$ and/or the number of the selected membership functions is equal to M_a, or $M = 0$, then stop, else set $k = k + 1$ and go to Step 2.

7. Compute the consequent parameters using (31).

The OLS algorithm is derived here using the already developed tools. A different version can be found in (Wang and Mendel, 1993), but the ultimate performance is the same.

A standard measure of fit is given by the normalized mean-squared error (NMSE) defined as

$$\text{NMSE} = \frac{1}{\sigma^2 L} \sum_{t=1}^{L} \big[y(t) - \widehat{y}(t)\big], \tag{58}$$

where σ^2 is the variance of $y(t)$ over the test duration L.

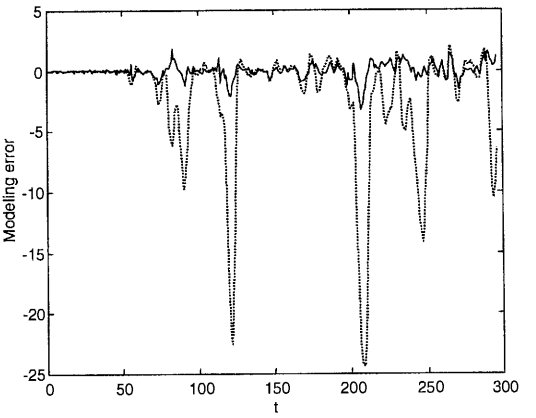
To test the effect of the training data size N on the performance of both the constructive and OLS algorithms, the fuzzy models were trained with the following sizes: $N = 50, 100, 150, 200, 250$, and 290. Figures 2(a)–(c) show that the constructive algorithm provides good performance with less sensitivity to the training data size when compared with the OLS algorithm. The NMSE values for different training data sizes illustrate numerically this superiority (see Fig. 3).

The multistep prediction performance of both the algorithms was examined using the fuzzy models trained with data size $N = 290$ and the following input vector:

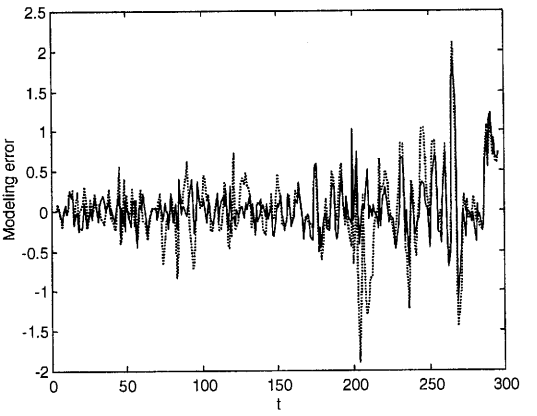
$$\boldsymbol{x}(t) = \Big[\widehat{y}(t-1), \widehat{y}(t-2), \widehat{y}(t-3), u(t), u(t-1), u(t-2)\Big]^T. \tag{59}$$

Figure 4 shows that the iterative output of the fuzzy model trained using the constructive algorithm tracks closely the true data over the testing set. Figure 5 depicts the numerical evaluation of the NMSE achieved by the two fuzzy models for multistep prediction. For values of N less than 290, the prediction performance of the OLS algorithm degrades rapidly and no comparison can be made.

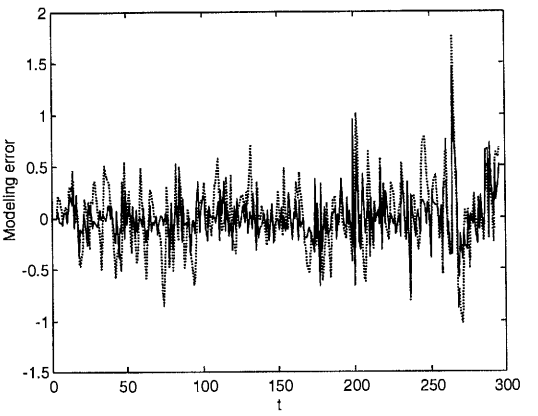
Figure 6 shows the two-dimensional plot of the membership functions selected by the OLS and constructive algorithms. The parameters of the constructed model ($N = 290$) are shown in Table 1. A comparison with other training algorithms tested on the gas furnace data is shown in Table 2.



(a)



(b)



(c)

Fig. 2. Modelling error for $N = 50$ (a), $N = 150$ (b), $N = 250$ (c) (solid line: constructive algorithm, dotted line: OLS algorithm).

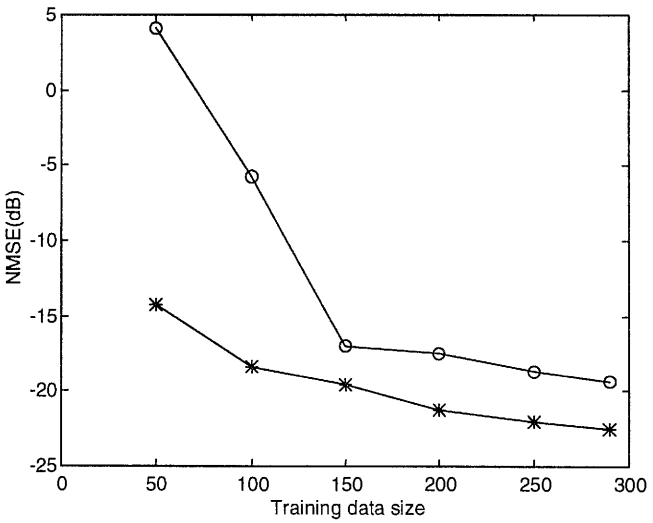


Fig. 3. NMSE values for different training data sizes (asterisks: constructive algorithm, open circles: OLS algorithm).

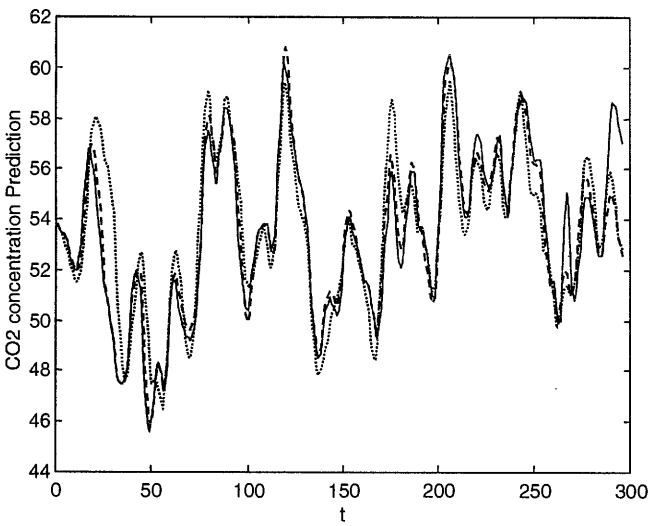


Fig. 4. Prediction of gas concentration (solid line: real data, dashed line: constructed algorithm, dotted line: OLS algorithm).

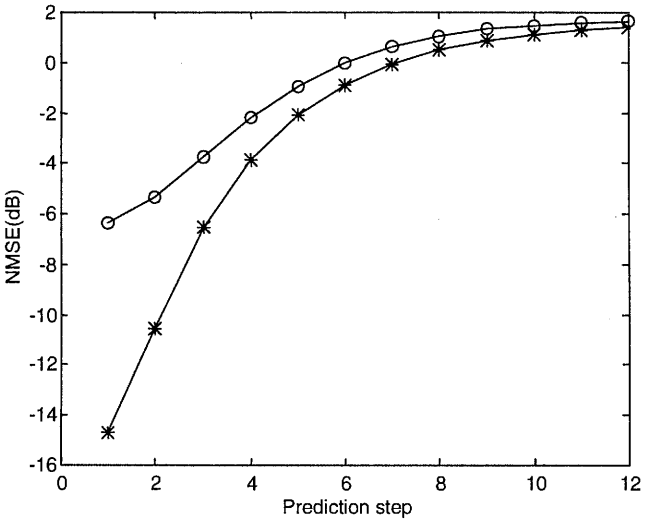


Fig. 5. NMSE evaluation of multisep prediction (asterisks: constructive algorithm, open circles: OLS algorithm).

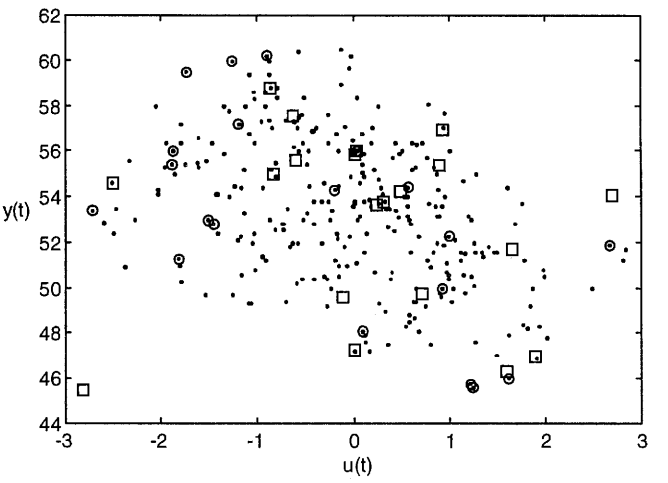


Fig. 6. Selected membership functions (two-dimensional plot, $N = 290$; open circles: OLS, open squares: constructive algorithm).

Table 1. Fuzzy model parameters ($N = 290$).

Rule No.	A_{1i}	A_{i2}	A_{i3}	A_{i4}	A_{i5}	A_{i6}	w_i
1	55.61	56.90	57.88	−0.608	−0.018	0.448	2027.6
2	45.50	48.48	52.18	−2.817	−2.525	−1.910	−161.9
3	54.05	53.67	52.75	2.698	3.088	3.091	−389.7
4	57.55	55.21	52.82	−0.636	−1.291	−1.733	192.5
5	51.71	53.25	54.51	1.656	0.757	−0.666	710.8
6	49.73	49.60	49.57	0.711	0.262	−0.430	−3671.5
7	49.57	49.63	50.05	−0.124	−1.027	−1.684	1387.7
8	55.04	56.08	56.97	−0.836	−0.519	−0.303	901.8
9	54.21	53.12	53.57	0.488	0.845	1.063	−2715.4
10	55.86	55.84	55.16	0.018	0.016	0.215	−5944.3
11	47.24	47.24	48.13	0.011	0.169	0.674	728.3
12	58.78	58.58	57.99	−0.860	−1.036	−1.111	660.2
13	46.94	47.84	48.23	1.896	2.018	1.810	1598.9
14	46.30	47.17	47.96	1.596	1.856	2.027	−1049.4
15	56.00	56.99	57.98	0.037	0.204	0.253	−1833.1
16	55.39	55.02	54.31	0.888	1.215	1.622	2567.9
17	53.63	51.86	51.62	0.245	0.284	0.032	2216.8
18	56.99	55.99	54.69	0.929	1.004	1.135	1310.1
19	53.79	53.80	53.75	0.316	0.229	0.116	1706.5
20	54.60	56.39	57.99	−2.502	−1.786	−1.343	−78.4

Table 2. Comparative results ($N = 290$).

Model	Number of rules	Number of parameters	NMSE (dB)
Box and Jenkins 1970	—	10	−13.89
Tong 1980	19	—	−6.57
Pedrycz 1984	81	—	−9.89
Xu and Lu 1987	25	—	−9.68
Lin And Cunnigham 1995	4	354	−22.97
OLS algorithm	20	140	−19.55
Constructive Algorithm	20	140	−23.75

7.2. Nonlinear System Control

In this subsection, we consider the application of the constructive algorithm to a nonlinear system control (Mukhopadhyay and Narandra, 1993). The nonlinear system considered here is represented mathematically as

$$y(t+1) = f\big(y(t), y(t-1), y(t-2), u(t-1)\big) + u(t), \tag{60}$$

where $u(t)$ and $y(t)$ are the input and the output of the system, respectively. The functional form of the nonlinear function $f(\cdot)$ is given by

$$f(\cdot) = \frac{y(t)\big[y(t-1) - 0.5u(t-1)\big]}{1 + y(t-1)^2} + \sin\big(y(t-1)\big) \exp\big(-y(t-2)^2\big). \tag{61}$$

The control problem can be stated as that of designing a control input that forces the nonlinear system (60) to follow the reference model given by

$$y_m(t+1) = 0.6y_m(t) + r(t) \tag{62}$$

with the reference input $r(t) = \sin(2\pi t/50) + \sin(2\pi t/20)$. If the nonlinear function $f(\cdot)$ is known, then it is an easy task to obtain the perfect following by using the control input

$$u(t) = y_m(t+1) - f(\cdot). \tag{63}$$

When this is not the case, the nonlinear function must be replaced by some approximation $\widehat{f}(\cdot)$. In the following, we investigate the performance of the constructive and OLS algorithms to construct fuzzy models that approximate the nonlinear function, such that

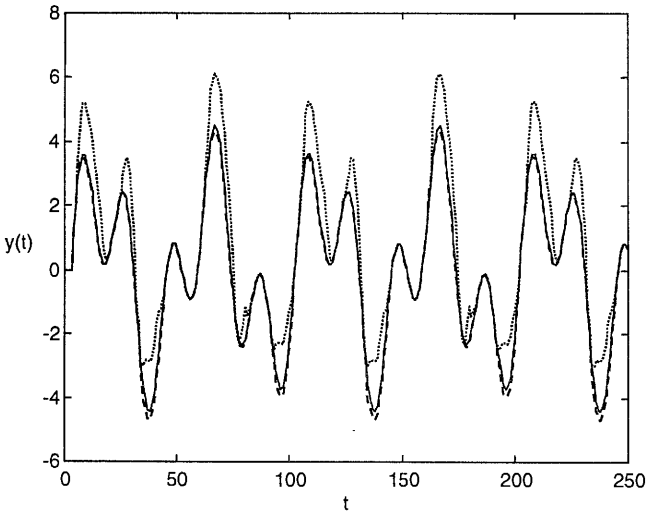
$$\widehat{f}(\cdot) = \mathrm{FM}\big[\boldsymbol{x}(t), \Theta, \boldsymbol{w}\big], \tag{64}$$

where the input vector $\boldsymbol{x}(t)$ is

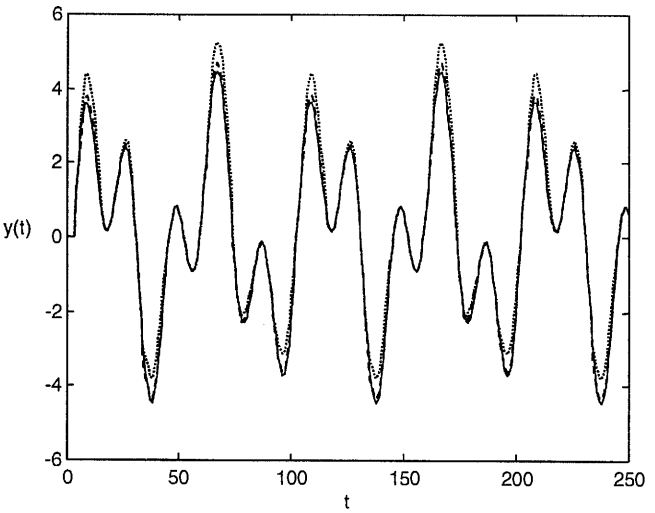
$$\boldsymbol{x}(t) = \big(y(t), y(t-1), y(t-2), u(t-1)\big). \tag{65}$$

To perform the nonlinear function identification, a training set $\{\boldsymbol{x}(t), y(t)\}$ of size N is generated from (60) by choosing $u(t)$ randomly with a uniform distribution in the range $[-1, 1]$. Two fuzzy models with the same size ($M_a = 100$) are built, as previously described, using the constructive and OLS algorithms. The identification was carried out for the training set sizes $N = 500$ and $N = 1000$.

Figures 7(a) and (b) depict the nonlinear system response when the control input (63) is computed with the nonlinear function $f(\cdot)$ replaced with one of the fuzzy models already constructed. It is clear that the constructive algorithm achieves a better tracking performance in all the cases. It is apparent from Figs. 8(a) and (b) that the OLS algorithm is remarkably sensible to the training data size, and the fuzzy model obtained by this algorithm does not approximate the nonlinear function well throughout the region of interest. The NMSE for the constructive algorithm is -23.62 dB and -25.14 dB for $N = 500$ and $N = 1000$, respectively, while the NMSE for the OLS algorithm is -7.49 dB and -15.05 dB for $N = 500$ and $N = 1000$, respectively. Note that the constructive algorithm learns faster than the OLS when the training data size tends to be large. The parameter optimization can be accelerated if qualitative knowledge is used to determine the initial values.

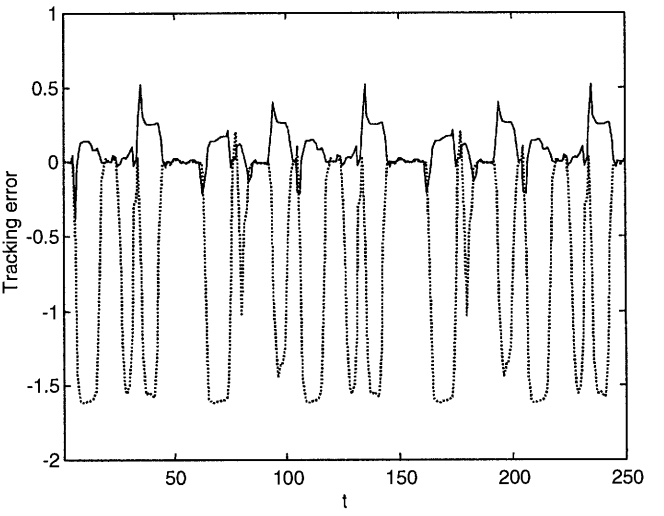


(a)

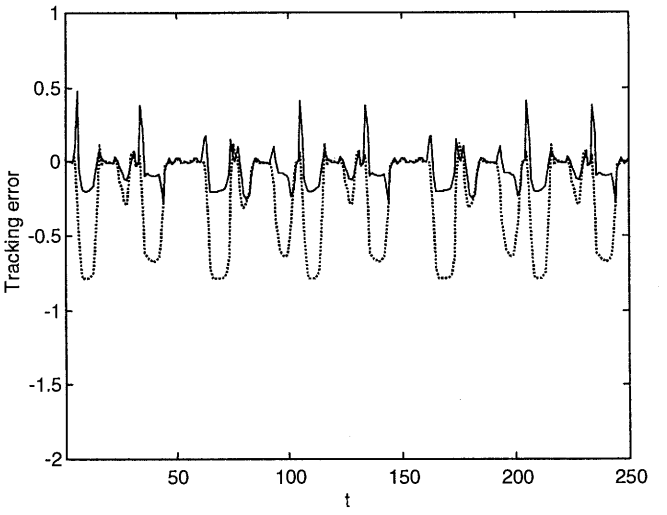


(b)

Fig. 7. Tracking performance: (a) $N = 500$, (b) $N = 1000$ (solid line: reference model, dashed line: constructive algorithm, dotted line: OLS algorithm).



(a)



(b)

Fig. 8. Tracking error: (a) $N = 500$, (b) $N = 1000$ (solid line: constructive algorithm, dotted line: OLS algorithm).

8. Conclusion

This paper has presented a systematic approach to the fuzzy model construction problem, due to which the appropriate structure and parameters can be found automatically. Both the structure and parameter identification schemes are executed simultaneously during the training process. This technique reveals two notable features: First, the rule selection task is decomposed into a number of sequential subtasks, which results in savings of computing time and space, and second, optimization of each membership function is related to the fuzzy model performance. The effectiveness of the proposed method was demonstrated with both real process data and a simulated nonlinear system. The simulation results have shown that the constructive algorithm is less sensitive to the training data size, which is an important feature required by many applications.

References

Box G.E.P. and Jenkins G.M. (1970): *Time Series Analysis, Forecasting and Control.* — San Francisco: Holden Day.

Cadzow J.A. (1990): *Signal processing via least squares error modeling.* — IEEE Signal Process. Mag., Vol.7, No.1, pp.12–32.

Eykhoff P. (1979): *System Identification.* — New York: Wiley.

Euntai K., Minkee P., Seunghwan J. and Migron P. (1997): *A new approach to fuzzy modeling.* — IEEE Trans. Fuzzy Syst., Vol.5, No.3, pp.328–337.

Golub G.H. and Pereyra V. (1973): *The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate.* — SIAM J. Num. Anal., Vol.10, pp.361–369.

Horikawa S.-I., Furuhashi T. and Uchikawa Y. (1992): *On fuzzy modelling using fuzzy neural networks with the back-propagation algorithm.* — IEEE Trans. Neural Networks, Vol.3, No.5, pp.801–806.

Homaifar A. and McCormick E. (1995): *Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms.* — IEEE Trans. Fuzzy Syst., Vol.3, No.2, pp.129–139.

Hohensohn J. and Mendel J.M. (1996): *Two-pass orthogonal least squares algorithm to train and reduce the complexity of fuzzy logic systems.* — J. Intell. Fuzzy Syst., Vol.4, pp.295–308.

Juang C.-F. and Lin C.-T. (1998): *An on-line self-constructing neural fuzzy inference network and its applications.* — IEEE Trans. Fuzzy Syst., Vol.6, No.1, pp.12–32.

Kaufman L. (1975): *A variable projection method for solving separable nonlinear least squares problems.* — BIT, Vol.15, pp.49–57.

Karr C. (1991): *Genetic Algorithms for fuzzy controllers.* — AI Expert, Vol.6, No.2, pp.26–33.

Kosko B. (1992): *Fuzzy systems as universal approximators.* — Proc. IEEE Int. Conf. *Fuzzy Syst.*, San Diego, CA, pp.1163–1170.

Lee C.C. (1992): *Fuzzy logic in control systems: Fuzzy logic controller: Parts* I & II. — IEEE Trans. Syst. Man Cybern., Vol.20, No.2, pp.404–435.

Lin Y. and Cunninghan G.A. (1995): *A new approach to fuzzy-neural system modeling.* — IEEE Trans. Fuzzy Syst., Vol.3, No.2, pp.190–197.

Mukhopadhyay S. and Narandra K.S. (1993): *Disturbance rejection in nonlinear systems using neural networks.* — IEEE Trans. Neural Networks, Vol.4, No.1, pp.63–72.

Nozaki K., Ishibuchi H. and Tanaka H. (1996): *Adaptive fuzzy rule-based classification systems.* — IEEE Trans. Fuzzy Syst., Vol.4, No.3, pp.238–250.

Pedrycz W. (1984): *An identification algorithm in fuzzy relational systems.* — Fuzzy Sets Syst., Vol.13, pp.153–167.

Poggio T. and Girosi F. (1990): *Networks for approximation and learning.* — Proc. IEEE, Vol.78, No.9, pp.1481–1499.

Sugeno M. and Kang G.T. (1988): *Structure identification of fuzzy model.* — Fuzzy Sets Syst., Vol.28, pp.15–33.

Schilling R.J and Lee H. (1988): *Engineering Analysis: A Vector Space Approach.* — New York: Wiley.

Tong R.M. (1980): *The evaluation of fuzzy models derived from experimental data.* — Fuzzy Sets Syst., Vol.4, pp.1–12.

Wang L.-X. and Mendel J.M. (1992): *Fuzzy basis functions, universal approximation, and orthogonal least-squares learning.* — IEEE Trans. Neural Networks, Vol.3, No.5, pp.807–813.

Wang L.-X. (1995): *Design and analysis of fuzzy identifiers of nonlinear dynamic systems.* — IEEE Trans. Automat. Contr., Vol.40, No.1, pp.11–23.

Xu C.W. and Lu Y.Z. (1987): *Fuzzy model identification and self-learning for dynamic systems.* — IEEE Trans. Syst. Man Cybern., Vol.17, pp.683–689.

Yoshinari Y., Pedrycz W. and Hirota H. (1993): *Construction of fuzzy models through clustering techniques.* — Fuzzy Sets Syst., Vol.54, pp.157–165.

Ying H. (1994): *Sufficient conditions on general fuzzy systems as function approximations.* — Automatica, Vol.30, No.3, pp.521–525.

Received: 11 January 1999
Revised: 22 June 1999