Input selection and partition validation for fuzzy modelling using neural network

D.A. Linkens, Min-You Chen

Department of Automatic Control and Systems Engineering, University of Sheffield, Mappin Street, Sheffield, S1 3JD, UK

Received August 1997; received in revised form October 1997

Abstract

A simple and effective method for selecting significant input variables and determining optimal number of fuzzy rules when building a fuzzy model from data is proposed. In contrast to the existing clustering-based methods, in this approach both input selecting and partition validating are determined on the basis of a class of sub-clusters created by a self-organising network instead of on the data. The important input variables which independently and significantly influence the system output can be extracted by a fuzzy neural network. On the other hand, the optimal number of fuzzy rules can be determined separately via the fuzzy c-means algorithm with a modified fuzzy entropy as the criterion of cluster validation. The simulation results show that the proposed method can provide good model structures for fuzzy modelling and has high computing efficiency. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Fuzzy modelling; Input selection; Fuzzy clustering; Fuzzy-neural networks

1. Introduction

Numeric data based fuzzy modelling, which was first explored systematically by Takagi and Sugeno [6,8] has found numerous applications in system identification, control, prediction and inference. According to Sugeno, fuzzy modelling comprises two procedures: structure identification and parameter estimation. There are two challenging problems in fuzzy structure identification: (1) extracting significant input variables among all possible input candidates, which is called input selection; (2) determining the number of rules needed, i.e. finding how many rules are necessary and sufficient to achieve the given mapping, which is called partition validation or cluster validation. Various methods have been proposed to deal with these problems either separately [2,3,5,10] or combinatorially [4,7]. Generally speaking, fuzzy structure identification involves three steps: (1) determine the optimal number of fuzzy rules and establish an initial fuzzy model. In most cases, fuzzy clustering is considered as an intuitive approach to generate the fuzzy models, (2) optimise the parameters of the initial fuzzy model; various approaches can be used to perform the parameter optimisation according to the different model formats; (3) extract the important input variables through evaluating performance (under a given performance index) of the initial model with different combinations of input variables in the model and iteratively removing the input variables which give no contribution...
to improving the model performance. To avoid exhaustive search of all combinations, suboptimal search such as forward search and backward search are adopted by most input selection methods.

Usually, structure identification of fuzzy models is time consuming and computationally expensive especially in systems with large numbers of training data and input candidates. Sugeno and Yasukawa [7] decoupled the input variable selection problem from the fuzzy partitioning problem and used a forward search method to determine input variables for fuzzy models. However, this method requires generating different fuzzy models for different combinations of input variables, thus increasing the computation burden. Chiu [2] proposed an efficient selection method based on generating only one model that employs all possible input variables and then systematically removing antecedent clauses in the fuzzy rules of this initial model to test the significance of each variable. This avoids the need to repeatedly generate new models to test each combination of variables. However, in this cluster estimation based approach, the number of rules depends on the user specified cluster radius. This heuristic parameter, obtained via trial-and-error, cannot be guaranteed to obtain the optimal number of rules.

A simple and fast method for variables selecting and fuzzy rule determination was proposed by Lin and Cunningham [4]. In this method, the training data are projected onto different input–output planes to obtain smoothed plots of the output value as a function of each input variable. The output is believed not to be dependent on a given input if the corresponding plotted fuzzy curve is relatively flat. Although the method is very fast, it cannot recognize and eliminate redundant input variables (i.e. closely correlated variables of which only one needs to be included in the model). Furthermore, the number of rules is decided by heuristics. Human interpretation is required and a different model structure may be obtained for the same system by different individuals.

In contrast to most existing fuzzy modelling methods which derive fuzzy models directly from raw data, our method provides a hierarchical clustering architecture based on a hybrid neural network for the fuzzy modelling procedure. Firstly, a fuzzy competitive neural network is exploited as a data pre-processor to extract a number of sub-clusters which can be viewed as an initial fuzzy model from raw data. This step is used to perform fuzzy classification with the objective of reducing the total number of training instances for the next stages. The number of clusters is determined by a classification phase which is based on the concept of maximum likelihood. Secondly, the problems of input selecting and fuzzy partition validating are dealt with separately on the basis of the initial fuzzy model. On one hand, a simplified fuzzy inference neural network is proposed to produce a fuzzy output for each input variable so that the importance of each input can be ranked and the significant input variables can be selected. On the other hand, we propose a modified fuzzy entropy measure as the fuzzy partition criterion based on the fuzzy c-means (FCM) clustering algorithm to decide the optimal number of fuzzy rules. The schematic block diagram of our method is shown in Fig. 1. Using this sub-cluster-based hierarchical clustering approach, the computation cost decreases greatly compared with data-based clustering because the

![Fig. 1. Hierarchical clustering based fuzzy modelling scheme.](image)
effort to generate the sub-clusters via a competitive cluster network is relatively small even when it involves a large number of input variables. It is particularly important when generating the initial fuzzy model from high-dimensional data. Finally, we demonstrate this approach by applying it to different types of non-linear system modelling.

2. Extracting the initial fuzzy model

Our method for extracting fuzzy model from data is based on using a self-organising network. Consider a collection of data points \( P = \{ P_1, P_2, \ldots, P_n \} \) in an \( s \) dimensional space that combines both input and output dimensions, where \( P_i = (x_{i1}, x_{i2}, \ldots, x_{im}, y_i) \), \( P \in \mathbb{R}^s \), \( i = 1, 2, \ldots, n \). Without loss of generality, we use a multi-input and single-output (MISO) model as a generic representation of fuzzy system because a system with multiple independent outputs can be presented as a collection of MISO systems. Modelling and inference is more straightforward for MISO fuzzy systems. Thus, the input–output data pair can be represented as \( P_i = (x_{i1}, x_{i2}, \ldots, x_{im}, y_i) \). For convenience of analysis, we firstly normalise the data points in the input dimensions so that they are bounded by a unit hypercube. The subsequent input data analysis can be processed in a unit hypercube.

2.1. Generating sub-clusters using a self-organising network

The data pre-processing can be considered as a clustering process which groups the data scattered in space \( \mathbb{R}^{m+1} \) into a collection of sub-clusters. The self-organising Kohonen network (as shown in Fig. 2) is introduced to produce the sub-clusters. The purpose of this stage is to classify the given training data into a small number, say \( p \ll n \), clusters using competitive learning. This step is used as a pre-processor to perform fuzzy classification with the objective of reducing the total number of training instances for the next stages, thus reducing the computing burden. The number of clusters is determined through a classification technique which is based on the concept of maximum likelihood. The unsupervised self-organising algorithm is presented as follows:

Step 1: Produce the first neuron in the competitive layer:
Consider there are \( n \) training patterns, \( P = \{ P_1, P_2, \ldots, P_n \} \), in \( (m + 1) \) dimensional space. Input the first input pattern \( P_1 = (x_{11}, x_{21}, \ldots, x_{m1}, y_1) \); set the iteration number \( l = 1 \). Let the first weight vector be \( W_1 = P_1 \), i.e. \( w_{1j} = x_{ji}, j = 1, 2, \ldots, m \).
Set the number of units \( n_1 = 1 \), the activation number of unit \( 1 N_{S1} = 1 \).

Step 2: For the \( l \)th input pattern (at the \( l \)th sampling instance):
Find the unit \( J \) which has the minimum distance to the current input pattern \( P_l \) by
\[
D(W_j, P_l) = \| W_j - P_l \| = \min_{i = 1, \ldots, n} \| W_i - P_l \|.
\]
The distance is defined as
\[
\| W_j - P_l \| = (W_j - P_l)(W_j - P_l)^T.
\]

Step 3: Determine the winner using the following rule:
If
\[
\begin{align*}
D(w_j, P_l) & \leq \delta, \quad \rightarrow \text{J is winner} \\
D(w_j, P_l) & > \delta, \quad \rightarrow \text{create a new unit}
\end{align*}
\]
If there is more than one winner, choose the one with the lowest index number as the winner.
If \( J \) is the winner, modify the weight vector of unit \( J \) by:
\[
W_j = W_j^{-1} + \alpha P_l - W_j, \quad \text{where} \ \alpha \ \text{is the learning rate which is determined by} \ \alpha = x_0/(N_{Sj} + 1), \ \text{where} \ x_0 \in [0, 1] \ \text{is the initial rate}; \ N_{Sj} = N_{Sj} + 1; \ l = l + 1.
\]
If creating a new unit, then the weight vector is given as $W_n^l = P_l$, $n = n_i + 1$.
If $l < n$, go to step 2; otherwise set $p = n_i$ and go to next step.

Step 4: Define the network output
The activation value of the output unit is defined as:
$$C_j = W_j; \quad j = 1, 2, \ldots, p$$
where $W_j = (w_{j1}, w_{j2}, \ldots, w_{jm+1})$ represents the prototype of the $j$th fuzzy cluster in input-output space.

When unsupervised learning is completed, a collection of $p$ fuzzy clusters $C = \{c_1, c_2, \ldots, c_p\}$ represented by the $p$ units in the competitive layer is produced.

2.2. Obtaining the initial fuzzy model

It can be seen that the produced $p$ units can be viewed as $p$ data clusters centred at $W = \{w_1, w_2, \ldots, w_p\}$. Each cluster centre $w_i = (w_{i1}, w_{i2}, \ldots, w_{im+1})$ is in essence a prototypical data point that exemplifies a characteristic input/output behaviour of the system we wish to model. Hence each cluster centre can be used as the basis of a rule that describes the system behaviour.

Consider a set of $p$ cluster centres $\{w_1, w_2, \ldots, w_p\}$ in an $(m+1)$-dimensional space. In the case of a MISO system, each vector $w_i$ can be decomposed into two component vectors $x_i^*$ and $y_i^*$. The cluster centre vector can be denoted as:
$$c_i = [x_i^*, y_i^*],$$
where
$$x_i^* = (x_{i1}^*, x_{i2}^*, \ldots, x_{im}^*) = (w_{i1}, w_{i2}, \ldots, w_{im}),$$
$$y_i^* = w_{im+1}, \quad y_i^* = (y_{i1}^*, y_{i2}^*, \ldots, y_{ip}^*).$$

We consider each cluster centre $c_i = (x_i^*, y_i^*)$ as a fuzzy rule that describes the system local behaviour. Intuitively, cluster centre $c_i$ represents the rule “If input is around $x_i^*$ then output is around $y_i^*$”. Given an input vector $x^*$, the degree to which rule $i$ is fulfilled is determined by the membership function $\mu_i(x_i)$. $\mu_i(x_i)$ can be defined as any type of membership functions including triangle, trapezoid and Gaussian. Here, we use the Gaussian function, i.e. $\mu_i = e^{-|x^* - x_i^*|^2}$, where $z$ is a constant.

The fuzzy output can be computed via the centre of gravity (COG) algorithm:
$$z = \frac{\sum_{i=1}^{p} \mu_i y_i^*}{\sum_{i=1}^{p} \mu_i}.$$
provides an easy mechanism to test the importance of each input variable without having to generate new models. The basic idea is simply to remove all antecedent clauses except one associated with a particular input variable from the rules and then compute the fuzzy output with respect to this input variable. Actually, we need not change the form of fuzzy rules, but simply let the associated clauses (e.g., \(X_{12}\) is \(A_{12}\)) be assigned a truth value of 1 to achieve the effect. Similarly to Lin and Cunningham’s approach [4], one could test the \(m\) input variables one by one via producing \(m\) input–output fuzzy curves based on the proposed initial fuzzy model. But here, we propose a simplified fuzzy inference neural network model which can generate in parallel all fuzzy outputs with respect to every individual input variable and test the importance of all \(m\) input variables simultaneously under the pre-defined index. The structure of this neural-fuzzy inference model is depicted in Fig. 3.

It can be seen that the model is composed of a three-layer feedforward network. Unlike common neural-fuzzy models, the input and output of each neuron are vectors. The prototypes of the \(p\) clusters \(\{c_1, c_2, \ldots, c_p\}\) generated by competitive learning are used as the input patterns of the network. The first layer is a fuzzification layer. The activation function of each neuron consists of a set of membership functions, i.e. \(\phi_i = (\phi_{i1}, \phi_{i2}, \ldots, \phi_{ip})\), where \(\phi_{ij}\) is the membership function of the \(j\)th fuzzy subset of the \(i\)th input variable, which is defined as

\[
\phi_{ij}(x_i^*) = \exp \left\{ - \left( \frac{x_i^* - x_{ij}^*}{\sigma_{ij}} \right)^2 \right\},
\]

\(i = 1, 2, \ldots, m; j = 1, 2, \ldots, p;\)

\(c_i = (x_{i1}^*, x_{i2}^*, \ldots, x_{im}^*, y_1^*)\).

Consider that all antecedent clauses are assigned the value 1 except for one dominant testing input variable, then fuzzy inference using multiplication as the AND operator and defuzzification using centre of gravity algorithm can be merged into one procedure which is implemented by the second layer of the network. In this layer, the output of the \(i\)th neuron \(z_i = (z_{i1}, z_{i2}, \ldots, z_{ip})\) denotes the fuzzy inference output corresponding to the contribution of the \(i\)th input variable. The output vector is computed by

\[
z_{ij} = \frac{\sum_{j=1}^{p} \phi_{ij}(x_i^*) y_j^*}{\sum_{j=1}^{p} \phi_{ij}(x_i^*)};
\]

\(i = 1, 2, \ldots, m; j = 1, 2, \ldots, p.\)

On the basis of \(m\) fuzzy output vectors, the importance of input variables can be recognised by calculating the change range of corresponding \(z_i\) which can be obtained in the output layer of the network by \(Rz_i = \max(z_i) - \min(z_i)\). The input selection is carried out according to the following steps:

(i) Define the importance factor of the \(i\)th input by

\[
F_i = \frac{Rz_i}{R_m}
\]

where \(R_m = \max\{Rz_1, Rz_2, \ldots, Rz_m\}\).

(ii) Rank the importance of all input variables according to their corresponding \(F_i\) values.

(iii) Remove all the input variables with respect to \(F_i < \lambda\), where \(\lambda \in (0, 1)\) is the pre-defined threshold.

Obviously, \(F_i = 1\) corresponds to the most important input variable, the large varying range of the fuzzy output \(R_i\) indicating the big influence of the corresponding input variable. A small value of \(F_i\) corresponds to a relatively unimportant input. When \(F_i\) is less than the threshold, i.e. \(F_i < \lambda\), the corresponding input variable is believed to be unimportant and can be removed. Assume that there are \(r\) inputs with the values of \(F_i > \lambda\), thus, a collection of \(r\) inputs are selected from \(m\) input variables.

(iv) Recognising the closely related input variables (independent input variable testing) calculate

![Fig. 3. Neural fuzzy model for input variable selection.](image-url)
the correlation functions, $\rho(x_i, x_j)$, between the selected input variables by
\[
\sigma(x_i, x_j) = \frac{1}{N} \sum_{k=1}^{N} (x_i - \bar{x}_i)(x_j - \bar{x}_j)
\]
where $\rho(x_i, x_j) \in [0, 1]$, $\bar{x}_i$, $\bar{x}_j$, $\phi_{x_i}$, $\phi_{x_j}$ are the means and variances of vector $x_i$ and $x_j$, respectively, $i, j = 1, 2, \ldots, r$; $r$ is the number of selected input variables. Owing to the normalisation of input variables, we can recognise how many independent variables there are among the $r$ selected inputs by the following rule: if $|\rho(x_i, x_j)| > \tau$, then $x_j$ is closely related with $x_i$, thus, removing the one which has a smaller value of importance factor from the list of selected significant input variables, where $\tau$ is the threshold.

Thus, the task of input selection is completed and a collection of $q$ ($q \leq r$) significant input variables are selected for the fuzzy model. It is easy to see that parameter optimisation of the initial model is not necessary.

\section{3. Validation of fuzzy partitions}

In this section, we discuss the method of determining the number of fuzzy rules which is equivalent to finding an optimal partition for a set of data or the number of clusters in fuzzy clustering. Based on the $p$ sub-clusters, a FCM clustering technique is used for partition validating. The simple and applicable support suggested by Sugeno and Yasukawa [7], which clusters only the output space applicable approach suggested by Sugeno and is used for partition validating. The simple and lent to minima of vector $x_i$ and $x_j$, respectively, $i, j = 1, 2, \ldots, r$; $r$ is the number of selected input variables. Owing to the normalisation of input variables, we can recognise how many independent variables there are among the $r$ selected inputs by the following rule: if $|\rho(x_i, x_j)| > \tau$, then $x_j$ is closely related with $x_i$, thus, removing the one which has a smaller value of importance factor from the list of selected significant input variables, where $\tau$ is the threshold.

Thus, the task of input selection is completed and a collection of $q$ ($q \leq r$) significant input variables are selected for the fuzzy model. It is easy to see that parameter optimisation of the initial model is not necessary.

\section{3. Validation of fuzzy partitions}

In this section, we discuss the method of determining the number of fuzzy rules which is equivalent to finding an optimal partition for a set of data or the number of clusters in fuzzy clustering. Based on the $p$ sub-clusters, a FCM clustering technique is used for partition validating. The simple and applicable support suggested by Sugeno and Yasukawa [7], which clusters only the output space applicable approach suggested by Sugeno and is used for partition validating. The simple and lent to minima of vector $x_i$ and $x_j$, respectively, $i, j = 1, 2, \ldots, r$; $r$ is the number of selected input variables. Owing to the normalisation of input variables, we can recognise how many independent variables there are among the $r$ selected inputs by the following rule: if $|\rho(x_i, x_j)| > \tau$, then $x_j$ is closely related with $x_i$, thus, removing the one which has a smaller value of importance factor from the list of selected significant input variables, where $\tau$ is the threshold.

Thus, the task of input selection is completed and a collection of $q$ ($q \leq r$) significant input variables are selected for the fuzzy model. It is easy to see that parameter optimisation of the initial model is not necessary.

\section{3. Validation of fuzzy partitions}

In this section, we discuss the method of determining the number of fuzzy rules which is equivalent to finding an optimal partition for a set of data or the number of clusters in fuzzy clustering. Based on the $p$ sub-clusters, a FCM clustering technique is used for partition validating. The simple and applicable support suggested by Sugeno and Yasukawa [7], which clusters only the output space applicable approach suggested by Sugeno and is used for partition validating. The simple and lent to minima of vector $x_i$ and $x_j$, respectively, $i, j = 1, 2, \ldots, r$; $r$ is the number of selected input variables. Owing to the normalisation of input variables, we can recognise how many independent variables there are among the $r$ selected inputs by the following rule: if $|\rho(x_i, x_j)| > \tau$, then $x_j$ is closely related with $x_i$, thus, removing the one which has a smaller value of importance factor from the list of selected significant input variables, where $\tau$ is the threshold.

Thus, the task of input selection is completed and a collection of $q$ ($q \leq r$) significant input variables are selected for the fuzzy model. It is easy to see that parameter optimisation of the initial model is not necessary.
Step 3: Take $t = 1, 2, \ldots, T$; calculate
\[
u_{ik,t} = \left[ \frac{1}{\sum_{j=1}^{c} \left( \frac{u_{ik,t} - u_{kj,t-1}}{u_{kj,t} - v_{kj,t-1}} \right)^{2(m-1)}} \right]^{1-1},
\tag{4}
\]
where
\[
v_{i,t} = \frac{\sum_{k=1}^{p} (u_{ik,t})^m y_k^m}{\sum_{k=1}^{p} (u_{ik,t})^m}.
\tag{5}
\]
If $\|V_t - V_{t-1}\| < \varepsilon$, go to next step, otherwise repeat Step 3.

Step 4: Calculate $H_m(c)$ by (3), if $H_m(c - 1) < H_m(c)$ and $H_m(c - 1) < H_m(c - 2)$, then stop, set the optimal cluster number $c = c - 1$; else repeat from step 2.

After the completion of partition validation, we can obtain both the number of rules and the prototypes of the output clusters $v = (v_1, v_2, \ldots, v_c)$. Let $a = (a_1, a_2, \ldots, a_c)$ denote the prototypes of fuzzy partition of the input space, where $a_i = (a_{i1}, a_{i2}, \ldots, a_{iq})$, $a_i = (a_{i1}, a_{i2}, \ldots, a_{iq})^T$ are the centre values of Gaussian membership functions in the antecedent of the $i$th rule. According to Sugeno’s approach, we can derive the input prototype $a$ from the $c$ cluster centre values of the output variable by projecting the output clusters onto the axes of input variables. The obtained $c$ prototypes can be used as the initial parameters of the fuzzy model.

So far, we have presented an efficient method for structure identification of fuzzy models. This method overcomes a computational bottleneck: the need to generate a new model to test each combination of input variables. Since the significant input variables can be decided by simply testing the importance rank of the input variables on the basis of the simplified neural-fuzzy inference model, optimisation of the initial fuzzy model is not necessary. Besides, as all computation related to FCM and fuzzy inference are based on the $p$ representative prototype points, ($p < n$), the computation cost is decreased drastically, especially in the case of a large number of training data.

4. Parameter optimisation

As mentioned, we have already completed the structure identification and obtained the initial model parameters. With these parameters, we can build the fuzzy model with $c$ rules as follows:

\[
R_i: \text{If } x_1 \text{ is } A_{i1} \text{ and } x_2 \text{ is } A_{i2} \ldots \text{ and } x_q \text{ is } A_{iq}
\]

Then $y$ is $v_i$

where $A_{ij}$ denotes the Gaussian membership function centred at $a_{ij}$, $a_{ij} \in a$; i.e.

\[
A_{ij} = \phi_{ij}(x_i) = \exp \left\{ - \left[ \frac{(x_j - a_{ij})^2}{\sigma_{ij}} \right] \right\};
\]

$i = 1, 2, \ldots, c; j = 1, 2, \ldots, q$.

To obtain satisfactory modelling accuracy, it is better to optimise the model parameters under a certain performance index. There are several methods for parameter optimisation. If the membership functions in the antecedent are fixed, the consequent parameters can be optimised simply by least-squares estimation. The antecedent parameters can be optimised by applying a gradient descent method. Here, we adopt the backpropagation-based approach, proposed by Wang and Mendel [9], to optimise the parameters $a_{ij}, \sigma_{ij}$ and $v_i$ combinatorially under the performance index of least-squares output error. After parameter learning, the final model is obtained.

5. Simulation results

In order to demonstrate the validity of our method, different kinds of non-linear modelling examples have been simulated. Three typical examples are given below:

Example 1. Non-linear system with two closely related inputs

\[
y = 0.2 + 0.8 \exp(-x_1) + 0.4 \sin(2\pi x_2),
\]

where the two input variables are closely related by the following mapping: $x_2 = 1 - x_1^2$.

The 100 input data points are randomly chosen within $[0, 1]$, i.e. $0 \leq x_1 \leq 1$, $x_2 = 1 - x_1^2$. In order to illustrate input variable selection, two dummy inputs $x_3$ and $x_4$ are added in the range $[0, 1]$. 

According to our method, the 100 input–output data points are firstly clustered into \( p = 34 \) sub-clusters by the self-organising network. Based on these sub-clusters, the ranges of the fuzzy outputs \( R_{z_1}, R_{z_2}, R_{z_3}, R_{z_4} \) with respect to each input computed by the neural network for input selection are 0.4137, 0.2972, 0.10420 and 0.1214, respectively. Obviously, \( x_1 \) and \( x_2 \) are selected as the significant input candidates. Through subsequent correlation analysis, the obtained value of correlation function between \( x_1 \) and \( x_2 \) is very high, i.e. \( \rho(x_1, x_2) = 0.948 \). Thus, only \( x_1 \) is selected as the important input variable of the given system.

Also, the fuzzy partition validation is progressed on the basis of \( p \) subclusters \( C = \{c_1, c_2, \ldots, c_p\} \). Using the proposed \( H_m \), the optimal number of fuzzy rules is obtained as \( n_r = 2 \). The fuzzy model produced by FCM clustering is shown in Fig. 4. The final model optimised by back-propagation learning after 50 iterations is illustrated in Fig. 5.

The comparison between model output and actual output is depicted in Fig. 6.

**Example 2.** Non-linear MISO system:

\[
y = (2 + x_1^{1.5} - 1.5 \sin(3x_2))^2.
\]

We randomly take 100 points from \( 0 \leq x_1, x_2 \leq 3 \) and obtain 100 input–output data. Again, we add random variables \( x_3 \) and \( x_4 \) in the range of \([0, 3]\) as the dummy inputs. 38 sub-clusters are created by the self-organising network. The range of fuzzy outputs corresponding to the 4 inputs are 36.9, 18.7, 5.16 and 9.01. Clearly \( x_1 \) and \( x_2 \) are chosen as the significant inputs. The input \( x_1 \) has little relation with \( x_2 \) because the correlation factor between \( x_1 \) and \( x_2 \) is \( \rho(x_1, x_2) = 0.1265 \). The optimal number of rules is chosen as \( n_r = 3 \). After 50 iterations parameter learning, the final fuzzy model is obtained.

![Fig. 4. Initial fuzzy model after FCM clustering.](image1)

![Fig. 5. The final fuzzy model.](image2)

![Fig. 6. Compared result of Example 1.](image3)

![Fig. 7. The final fuzzy model of Example 2.](image4)
and illustrated in Fig. 7. The model simulation result is shown in Fig. 8.

**Example 3.** Non-linear dynamic system:

\[
y(t + 1) = \frac{y(t)y(t - 1)(y(t) + 2.5)}{1 + y^2(t) + y^2(t - 1)} + u(t);
\]

\[
u(t) = \sin(0.04 \pi t)
\]

100 training points are generated by taking \( t = 1, 2, \ldots, 100 \). The system starts from zero initial state. The variables \( y(t), y(t - 1), y(t - 2), u(t) \) and \( u(t - 1) \) are considered as input candidates. The selected important inputs are \( y(t) \) and \( u(t) \). After 50 iterations of parameter training, the final fuzzy model with 3 rules and the corresponding simulation result are shown in Figs. 9 and 10, respectively. It is noted that the variable \( y(t - 1) \) is omitted in the final fuzzy model because it is related to \( y(t) \) closely and thus only \( y(t) \) is needed to be included in the model. Clearly, the final fuzzy model is simple and can approximate the dynamic system quite well.

### 6. Conclusions

We have proposed a fast and efficient fuzzy modelling method which is based on a self-organising network. The simulation results are highly encouraging, which support the following conclusions:

1. The proposed neural network based input selecting approach is simple and efficient. In this approach, neither fuzzy models for different input combinations nor parameter optimisation for an initial fuzzy model are needed. Through the network, the fuzzy outputs with respect to each input can be produced in parallel and analysed, thus the important input variables can be selected simultaneously.

2. Fuzzy cluster validation proceeds on the basis of the sub-clusters generated by the self-organising network instead of the whole data set. Thus, the computing load is decreased dramatically due to the much smaller number of sub-clusters, especially in the situation of a large number of input data. Compared to the computing effort for FCM, the effort for competitive learning in the data pre-processing is small. This hierarchical clustering method (i.e. fuzzy clustering is based on the produced
sub-clusters instead of on the whole data) provides a fast and efficient method for fuzzy clustering.

3. The whole procedure of input selection and partition validation is carried out automatically without human intervention (interpretation).

This methodology can be used in conjunction with different criteria for model structure selection. It is also a fast method for generating fuzzy models based on neural network and fuzzy clustering techniques. Since this method focuses on model simplicity and computing efficiency for a satisfactory modelling accuracy, the produced model structure (including the selected inputs and the number of rules) may not be optimal, but suboptimal instead. Further improvement of model structure optimisation for this method would be beneficial.

References


