Journal of Intelligent & Fuzzy Systems, vol. 4, no. 4, pp. 243-256, 1996

Selecting Input Variables for Fuzzy Models

Stephen L. Chiu

Rockwell Science Center
1049 Camino Dos Rios
Thousand Oaks, California 91360, USA

Abstract

We present an efficient method for selecting important input variables when building a fuzzy model from data. Past methods for input variable selection require generating different models while searching for the optimal combination of variables; our method requires generating only one model that employs all possible input variables. To determine the important variables, premises in the fuzzy rules of this initial model are systematically removed to search for the best simplified model without actually generating any new models. We also present an efficient method for generating the initial model that typically must incorporate a large number of input variables. These methods are illustrated through application to the benchmark Box and Jenkins gas furnace data; the results are compared with those of other fuzzy models found in literature.

1. Introduction

One of the great challenges in modeling nonlinear systems is selecting the important input variables from all possible input variables. For example, for an automobile painting process, variables such as booth temperature, humidity, pressure, paint temperature, paint nozzle pressure, surface temperature, and drying time may all affect the result, but to different degrees. From a modeling perspective, incorporating only the important variables into a model provides a simpler, more useful, and more reliable model; the model will also be more practical to apply because fewer variables need to be measured. From a control perspective, understanding the relative importance of variables allows the process control engineers to focus their efforts on the variables that matter, eliminating the time and cost involved in controlling and finding good set-points for unimportant variables.

For linear models, Akaike's information criterion (AIC) [Akaike 1974] is often used for selecting an optimal set of input variables. Application of AIC involves building models that incorporate different sets of input variables and selecting the model that minimizes this criterion. AIC balances the model's error with the model's complexity based on sound statistical principles. However, for nonlinear models, such as the polynomial models generated from the Group Method of Data Handling (GMDH) [Ivakhnenko et al. 1979] or fuzzy models, statistically rigorous criteria for model selection do not exist; one must rely on heuristic criteria.

Even when a good criterion exists for model selection, there is no guarantee that a model based on a given set of variables is optimal unless all possible combinations of variables have been explored. Hence, finding an optimal solution requires building a model for each possible combination of input variables, which becomes computationally prohibitive for problems involving even a moderate number of candidate input variables (the number of possible combinations is 2^N, where N is the number of candidate variables). The computational problem is exacerbated when each model itself requires significant effort to generate (e.g., fuzzy models or neural network models). This trade-off between optimality and computational complexity has long been recognized, and thus virtually all methods for input variable selection are suboptimal search methods that do not guarantee optimality of the solution. One notable exception is the *branch and bound* algorithm developed by Narendra and Fukunaga [1977] in the context of pattern classification; this algorithm can find an optimal solution without exhaustive search if a monotonicity condition is satisfied. The monotonicity condition requires the value of the model selection criterion to change monotonically with each level of nested sets of variables. For example, if V_1, V_2,...,V_k are different subsets of the input variables and $J(.)$ is a performance criterion used to evaluate the models associated with each subset of variables, then the monotonicity condition requires

$$V_1 \subset V_2 \subset ... \subset V_k \Rightarrow J(V_1) \geq J(V_2) \geq ... \geq J(V_k) \ .$$

The monotonicity condition basically legitimizes early backtracking during the search, because it guarantees that there will not be a better node downstream. A criterion that satisfies the monotonicity condition, for example, is the root mean square (RMS) output error of linear models evaluated with respect to the same training data used to generate the models. However, most model selection criteria used in building practical models do not satisfy the monotonicity condition (e.g., the RMS output error with respect to an independent set of checking data).

An often adopted suboptimal search method, called *forward selection*, involves systematically generating models with gradually increasing number of variables. A collection of one-input models is first generated based on each of the candidate input variables, and the input variable that provided the best model is selected as an important input. This input is then combined with every remaining candidate input to generate a collection of two-input models, and the inputs that provided the best two-input model are selected as important inputs. As this procedure continues, more inputs are added to the list of important inputs until the performance of the resultant model ceases to improve. An alternative search method, called *backward selection*, starts with a model that contains all candidate input variables and then removes one input variable at each stage by evaluating a collection of simpler models; this process continues until the performance of the model becomes unacceptable. The forward and backward selection methods originated in the 1960's in the context of feature selection for pattern classification [Marill and Green 1963]. Although forward and backward selection are much more efficient than exhaustive search, they are still computationally expensive, requiring many different models to be generated while exploring the different combinations of variables.

The forward selection method was used to determine input variables for fuzzy models in [Sugeno and Kang 1988] and [Sugeno and Yasukawa 1993]. In [Sugeno and Kang 1988], the input variable selection problem was folded into the fuzzy partitioning problem (i.e., determining how to divide the range of each input variable into more than one fuzzy subspaces or membership functions). In their approach, fuzzy models with increasingly complex partition patterns are generated while watching a model selection criterion; input variables that require no partitioning are the ones that can be removed from the model. In [Sugeno and Yasukawa 1993], the input variable selection problem was decoupled from the fuzzy partitioning problem, and forward selection was used directly for variable selection in the more familiar way. In [Hayashi et al. 1992], backward selection was used to determine input variables for a neural network-based fuzzy model. Tanaka et al. [1995] used backward selection with simple linear models to provide a first-round elimination of the input variables for a fuzzy model; a fuzzy model is then constructed using the remaining input variables through a fuzzy partitioning procedure similar to [Sugeno and Kang 1988], where additional variables may be eliminated. Forward selection is usually the preferred method over backward selection, because forward selection starts with simple models that are easy to generate and builds up complexity only as necessary.

An extremely fast method for variable selection was proposed in [Lin and Cunningham 1994]. 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 curve is relatively flat. Although this method is extremely fast, it can be derailed when the output surface is averaged out with respect to the hidden dimensions; furthermore, this method cannot recognize and eliminate redundant input variables (i.e., closely correlated variables of which only one needs to be included in the model).

Yen et al. [1993] proposed using principal component analysis, a statistical analysis technique, to reduce the number of inputs for fuzzy models. Principal components are eigenvectors of the input variables' covariance matrix; the first principal component corresponds to a linear combination of the input variables that produces maximum variance in the input value (i.e., maximum input excitation) [Anderson 84]. The number of inputs can be reduced by using the first few principal components as inputs, neglecting those linear combinations with small variance. An original input variable can be discarded if the linear coefficient associated with the variable is small in the chosen principal components. Although this method is a computationally efficient way to select input variables, the selection is based on variability in the input value, not based on whether the input actually affects the output. Conceivably an input variable with large variance may be completely unrelated to the output.

We present an efficient and reliable method for identifying important input variables for fuzzy models. Our method is based on generating only one fuzzy 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 is essentially a backward selection procedure that avoids the need to repeatedly generate new models to test each

combination of variables. Using an efficient clustering-based method for fuzzy model extraction, the effort required to generate the initial model is relatively small even when it involves a large number of input variables. We demonstrate this algorithm by applying it to the well-known Box and Jenkins gas furnace data [Box and Jenkins 1970]. We also examine the use of different model selection criteria in the context of this algorithm and discuss the strengths and deficiencies of each criterion.

2. Extracting the Initial Fuzzy Model

The input variable selection procedure begins by first extracting an initial fuzzy model that incorporates all possible input variables. In this section we describe briefly our method for extracting the initial fuzzy model.

Our method for extracting fuzzy model from data is based on using a cluster estimation method to determine the number of rules and initial rule parameters and then applying optimization algorithms to tune the rule parameters. Because a rule is generated only where there is a cluster of data, the resultant rules are scattered in the input space rather than placed according to grid-like partitions in the input space. This feature of the model extraction method is particularly important for models with large number of input variables; it avoids the typical combinatorial explosion of rules with increasing dimension of the input space. Also, because the clustering step provides good initial rule parameter values, the subsequent optimization process converges quickly and consistently to a good solution.

2.1 Estimating Clusters

The cluster estimation method for determining the number of rules and initial rule parameters is described in detail in [Chiu 1994]. For completeness, here we provide a brief description of the method.

Consider a collection of n data points $\{x_1,x_2,...,x_n\}$ in an M dimensional space that combines both input and output dimensions. Without loss of generality, we assume that the data points have been normalized in each dimension so that they are bounded by a unit hypercube. We consider each data point as a potential cluster center and define a measure of the potential of data point x_i to serve as a cluster center as

$$P_i = \sum_{j=1}^{n} e^{-\alpha \| x_i - x_j \|^2} \qquad (1)$$

where $\alpha = \frac{4}{r_a^2}$, (2)

$\|.\|$ denotes the Euclidean distance, and r_a is a positive constant. Thus, the measure of the potential for a data point is a function of its distances to all other data points. A data point with many neighboring data points will have a high potential value. The constant r_a is effectively the radius defining a neighborhood; data points outside this radius have little influence on the potential.

After the potential of every data point has been computed, we select the data point with the highest potential as the first cluster center. Let x_1^* be the location of the first cluster center and P_1^* be its potential value. We then revise the potential of each data point x_i by the formula

$$P_i \Leftarrow P_i - P_1^* e^{-\beta \| x_i - x_1^* \|^2} \qquad (3)$$

where $\beta = \frac{4}{r_b^2}$

and r_b is a positive constant. Thus, we subtract an amount of potential from each data point as a function of its distance from the first cluster center. The data points near the first cluster center will have greatly reduced potential,

and therefore will unlikely be selected as future cluster centers. The constant r_b is effectively the radius defining the neighborhood which will have measurable reductions in potential; we typically choose $r_b = 1.25\, r_a$.

When the potential of all data points have been revised according to Eq. (3), we select the data point with the highest remaining potential as the second cluster center. We then further reduce the potential of each data point according to their distance to the second cluster center. In general, after the *k*'th cluster center has been obtained, we revise the potential of each data point by the formula

$$P_i \Leftarrow P_i - P_k^* \, e^{-\beta \| x_i - x_k^* \|^2}$$

where x_k^* is the location of the *k*'th cluster center and P_k^* is its potential value.

The process of acquiring new cluster center and revising potentials repeats until the remaining potential of all data points are below some fraction of the potential of the first cluster center P_1^*; we typically use $P_k^* < 0.15 P_1^*$ as the stopping criterion. Some additional criteria for accepting and rejecting cluster centers are detailed in [Chiu 1994]. This cluster estimation method, called *subtractive clustering*, is an extension of the grid-based *mountain clustering* method developed by Yager and Filev [1994].

2.2 Obtaining the Initial Fuzzy Model

Each cluster center is in essence a prototypical data point that exemplifies a characteristic input/output behavior of the system we wish to model. Hence, each cluster center can be used as the basis of a rule that describes the system behavior.

Consider a set of *c* cluster centers $\{x_1^*, x_2^*, ..., x_c^*\}$ in an *M* dimensional space. Let the first *N* dimensions correspond to input variables and the last *M-N* dimensions correspond to output variables. We decompose each vector x_i^* into two component vectors y_i^* and z_i^*, where y_i^* is the location of the cluster center in input space and z_i^* is the location of the cluster center in output space. That is,

$$x_i^* = \begin{bmatrix} y_i^* \\ z_i^* \end{bmatrix}$$

where $y_i^* \in \Re^N$ and $z_i^* \in \Re^{M-N}$.

We consider each cluster center x_i^* as a fuzzy rule that describes the system behavior. Intuitively, cluster center x_i^* represents the rule "if input is near y_i^* then output is near z_i^*." Given an input vector *y*, the degree to which rule *i* is fulfilled is defined as

$$\mu_i = e^{-\alpha \| y - y_i^* \|^2}$$

where α is the constant defined by Eq. (2). We compute the output vector *z* via

$$z = \frac{\sum_{i=1}^{c} \mu_i \, z_i^*}{\sum_{i=1}^{c} \mu_i} .$$

We can view this computational model in terms of a fuzzy inference system employing traditional fuzzy if-then rules. Each rule has the following form:

if Y_1 is A_{i1} & Y_2 is A_{i2} &... then Z_1 is B_{i1} & Z_2 is B_{i2} ...

where Y_j is the j'th input variable and Z_j is the j'th output variable; A_{ij} is an exponential membership function in the i'th rule associated with the j'th input and B_{ij} is a singleton in the i'th rule associated with the j'th output. For the i'th rule, which is represented by cluster center x_i^* , A_{ij} and B_{ij} are given by

$$A_{ij}(Y_j) = \exp\left\{-\frac{1}{2}\left(\frac{Y_j - y_{ij}^*}{\sigma_{ii}}\right)^2\right\} \qquad (4)$$

$$B_{ij} = z_{ij}^* \qquad (5)$$

where y_{ij}^* is the j'th element of y_i^* , z_{ij}^* is the j'th element of z_i^* , and $\sigma_{ij}^2 = 1/(2\alpha)$. Our computational scheme is equivalent to an inference method that uses multiplication as the AND operator, weights the output of each rule by the rule's firing strength, and computes the output value as a weighted average of each rule's output. This type of fuzzy system can also be viewed as a radial basis function network [Jang and Sun 1993], with each rule representing a node in the network.

Although the number of rules (or clusters) is automatically determined by this method, we should note that the user specified parameter r_a (i.e., the cluster radius) strongly affects the number of rules/clusters that will be generated. A large r_a will result in fewer rules/clusters and hence a coarser model, while a small r_a can produce excessive numbers of rules/clusters and hence a model that does not generalize well. Therefore, r_a can be regarded as an approximate specification of the desired resolution of the model, which can be adjusted based on the resultant complexity and generalization ability of the model. Fortunately, our input variable selection method is not sensitive to the resolution of the initial model, as long as the initial model does not excessively over-fit the training data.

2.3 Optimizing the Fuzzy Model

There are several approaches to optimize the rules obtained from the cluster estimation method. One approach is to apply a gradient descent method to optimize the parameters y_{ij}^* , z_{ij}^* , and σ_{ij} in Eqs. (4) and (5) to minimize the RMS output error with respect to the training data. Backpropagation formulas to perform this optimization have been derived by Wang and Mendel [1992]. This is the approach adopted by Yager and Filev [1994] to optimize the rules obtained from the mountain clustering method.

Another approach is to let the consequent parameter z_{ij}^* be a linear function of the input variables, instead of a simple constant. That is, we let

$$z_{ij}^* = G_{ij}\, y + h_{ij}$$

where G_{ij} is an N-element vector of coefficients and h_{ij} is a scalar constant. The if-then rules then become the Takagi-Sugeno type. As shown by Takagi and Sugeno [1985], given a set of rules with fixed premise membership functions, optimizing G_{ij} and h_{ij} in all consequent equations to minimize the RMS output error is a simple linear least-squares estimation problem. Chiu [1994] adopted this approach to optimize the rules obtained from the subtractive clustering method; optimizing only the coefficients in the consequent equations allows a significant degree of model optimization to be performed without adding much computational complexity. This approach produced models that compared favorably with those obtained from more computationally intensive methods.

A third approach also involves using the Takagi-Sugeno rule format, but it combines optimizing the premise membership functions by gradient descent with optimizing the consequent equations by linear least squares estimation. This is the ANFIS (Adaptive Network-Based Fuzzy Inference System) methodology developed by Jang

[1993]. When the ANFIS approach is used to optimize the same rules, the resultant model is generally more accurate than that obtained from either of the two preceding approaches.

3. Selecting Input Variables

3.1 Testing Input Variable Combinations Efficiently

After an initial fuzzy model that incorporates all possible input variables has been generated by using the subtractive clustering method in conjunction with one of the aforementioned rule optimization methods, we can ascertain the importance of each input variable from this initial model. The fuzzy rule framework provides an easy mechanism to test the importance of each input variable without having to generate new models. The basic idea is to simply remove all antecedent clauses associated with a particular input variable from the rules and then evaluate the performance of the model. For example, suppose that the initial model has three inputs, with rules of the form:

if Y_1 is A_{i1} & Y_2 is A_{i2} & Y_3 is A_{i3} then Z_1 is B_{i1}.

We can test the importance of the Y_2 variable in the model by temporarily removing the antecedent clauses that involve Y_2, thus truncating the rules to the form:

if Y_1 is A_{i1} & Y_3 is A_{i3} then Z_1 is B_{i1}.

If the resultant model performance does not degrade with respect to some performance measure (e.g., by testing the model with respect to an independent set of checking data), then we can eliminate Y_2 from the list of possibly important variables. In practice, we need not actually remove antecedent clauses from the rules, but simply let the associated clauses (e.g., "Y_2 is A_{i2}") be assigned a truth value of 1.0 to achieve the effect.

The systematic procedure for input variable selection is as follows:

1. Evaluate the performance of the initial model with all candidate input variables in the model.
2. For each remaining input variable, evaluate the performance of the initial model with this variable temporarily removed.
3. Permanently remove the variable associated with the best model performance obtained in step 2. Record the resultant reduced variable set and the associated model performance.
4. If there are still variables remaining in the model, go back to step 2 to eliminate another variable. Otherwise go to step 5.
5. Choose the best variable set from among the sets recorded in step 3.

This is essentially a backward selection procedure that starts with all possible variables and reduces the number of variables by one at each stage. For example, starting with 4 input variables, this procedure first selectively removes one variable to arrive at the best 3 variable set; from among the 3 remaining variables, it then selectively removes one more variable to arrive at the best 2 variable set, and so on; among these "best" variable sets, the set that provided the best overall performance is chosen at the end, after we have reached a model with no input variable. The variable selection process for a four-input initial model is shown in Fig. 1. Because testing the different combinations of variables requires only truncating the rules in an initial full model and not generating any new models, the variable selection process is highly efficient.

While a backward selection process typically stops when the model's performance becomes unacceptable at some stage, we found it more useful to carry the process to the end until the performance of a model with no input variable is evaluated. It is important to let the human designer examine the model performance as a function of the number of variables removed and determine how to trade off accuracy for simplicity. The role of human judgment in this variable selection method will become clear as the method is illustrated through an example.

There are four conditions under which removing an input variable from the model will not degrade model performance: (1) the output does not vary with respect to the given input; (2) the output variation with respect to the given input is due to noise; (3) the given input is redundant such that the output variation can be predicted equally well by using other inputs; (4) the model is a highly over-fitted model. The first three conditions occur when an input variable is not important; the last condition occurs when the model is so over-fitted to noise that removing an important input variable can nonetheless improve the model's performance. Hence, we must be careful to avoid excessively complex initial models that over-fit the training data. The initial model does not have to be a highly accurate model because variable selection is based on comparing *relative* performances of the model as each variable is removed. In view of the problems associated with over-fitting the training data, it is generally preferable to use a coarse initial model for the purpose of variable selection.

After the input variables have been selected, we then reapply the clustering-based model extraction method to the training data to obtain the final model, this time using only the selected input variables in the training data. It is important to note that the truncated rules evaluated during variable selection are merely the mathematical mechanism for collapsing the initial full model's output surface to a lower dimension input space, and that these truncated rules usually bear no resemblance to the final rules that would be extracted using the same subset of input variables. This is evident when we consider that the number of truncated rules at any stage of the variable selection process will always remain the same as the number of rules in the initial full model, whereas the final model extracted using only the selected variables typically will have significantly fewer rules. Therefore, we choose not to re-optimize the truncated rules when comparing models during variable selection, since it is problematic that optimization will help a truncated-rule model to more accurately reflect the performance of the corresponding final model. In particular, if optimized, a truncated-rule model would tend to over-fit the training data due to the excessive number of tunable rules.

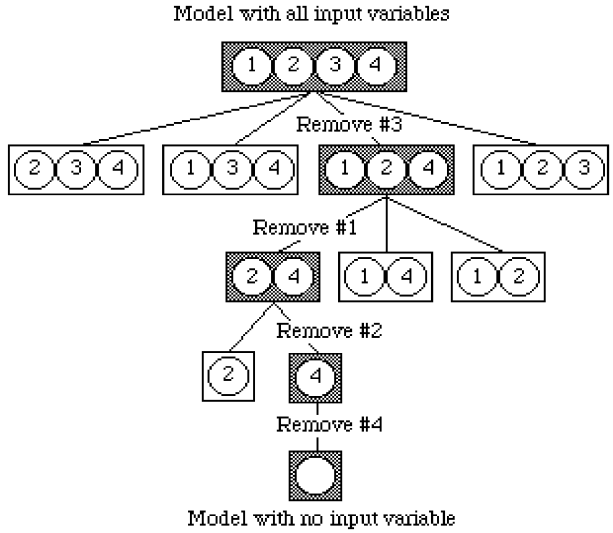

Fig. 1. Variable selection process for a four-input initial model. The different subsets of variables considered at each stage are shown; the shaded subsets are the ones that provided the best performance at each stage. After the variable removal process is carried to the end, we select from among the shaded subsets the subset that provides the best performance/complexity trade-off.

3.2 Model Selection Criteria

The variable selection procedure can be used with various performance criteria for model selection. The simplest performance criterion is the model's RMS output error with respect to an independent set of checking data. That is, we first divide the available data into two groups: A and B; we then generate the initial model by using the data in group A and call this model “model A”. The checking error criterion is then defined as:

$$J_C = \left[\sum_{i=1}^{nB} (z_i^B - z_i^{BA})^2 / n_B \right]^{1/2} \qquad (6)$$

where z_i^B is the *i*'th output data in group B; z_i^{BA} is the corresponding predicted output for group B obtained from model A; and n_B is the number of data points in group B. We repeatedly evaluate this criterion as each input variable is removed from model A and choose the variable subset that produced the smallest J_C at each stage. Although the checking error criterion favors model structures with high prediction accuracy, a drawback of this criterion is that the results are sensitive to the choice of training data and checking data.

Other possible performance criteria include the *unbiased criterion* and *regularity criterion* that originated from GMDH [Ivakhnenko et al. 1979]. These criteria are based on dividing the data into two groups and generating two separate models, one from group A and one from group B. The unbiased criterion is defined in [Sugeno and Kang 1988] as (there are other variations):

$$J_U = \left[\sum_{i=1}^{nA} (z_i^{AB} - z_i^{AA})^2 + \sum_{i=1}^{nB} (z_i^{BA} - z_i^{BB})^2 \right]^{1/2} \qquad (7)$$

The unbiased criterion is designed to make the chosen model insensitive to the data on which it was built. This criterion seeks to minimize only the difference between the output of the two separate models, without regard to the models' prediction accuracy; hence this criterion often does not result in selecting variables with the best prediction ability.

The regularity criterion is defined in [Sugeno and Yasukawa 1993] as:

$$J_R1 = \left[\sum_{i=1}^{nA} (z_i^A - z_i^{AB})^2 / n_A + \sum_{i=1}^{nB} (z_i^B - z_i^{BA})^2 / n_B \right] / 2 \qquad (8)$$

The regularity criterion is simply the mean square of the checking error criteria for the two separate models (there are other definitions of regularity criterion that involves only one model, see [Farlow 1984]). This criterion provides a good compromise between prediction accuracy and insensitivity to the choice of data. We prefer to use the square root of this criterion so that it becomes an RMS type of error, which gives the human designer a better feel for the error magnitude. Hence, we define the regularity criterion J_{R2} as:

$$J_{R2} = \sqrt{J_{R1}} \qquad (9)$$

In the case that either the unbiased criterion or the regularity criterion is used for variable selection, our procedure requires generating only two initial models instead of two models for each combination of variables to be tested.

The variable selection method can be applied to both conventional fuzzy models where the rule consequent is a singleton or membership function and Takagi-Sugeno models where the rule consequent is a linear equation in the input variables. However, application to conventional fuzzy models is straightforward, while application to Takagi-Sugeno models requires recomputation of some coefficients in the consequent equations as each variable is removed from the rules. For computational efficiency, and because the variable selection mechanism is not sensitive to the absolute accuracy of the initial model, we always choose the initial model to be a conventional fuzzy model (with singleton consequents) optimized by gradient descent. After the input variables have been selected from the initial model, we typically generate the final model as a Takagi-Sugeno model optimized by the ANFIS technique. This approach, which consists of generating a conventional fuzzy model from high dimension data for the variable selection phase and then generating a Takagi-Sugeno model from the reduced dimension data in the final phase, generally provides the best result in terms of both model accuracy and computation time.

4. Example

To illustrate the input variable selection method, we apply it to the well-known Box and Jenkins gas furnace data [Box and Jenkins 1970]. This is a time series data for a gas furnace process with gas flow rate $u(t)$ as the input and CO_2 concentration $x(t)$ as the output. We wish to extract a dynamic process model to predict $x(t)$ using the ten candidate input variables $\{x(t\text{-}1), x(t\text{-}2),...,x(t\text{-}4), u(t\text{-}1),u(t\text{-}2),...,u(t\text{-}6)\}$. The data set contains 296 $\{u(t),x(t)\}$ data pairs; after converting the data so that each training data point consists of $\{x(t),x(t\text{-}1),...,x(t\text{-}4),u(t\text{-}1),...,u(t\text{-}6)\}$, the number of effective data points reduces to 290. We use the first 145 data points as data set A and the last 145 data points as data set B. We will examine the results of using the different criteria for model selection and point out their strengths and deficiencies.

4.1 The Checking Error Criterion

An initial fuzzy model consisting of 11 rules was extracted from data set A by using cluster radius r_a = 0.5. The rules are of the conventional type where the consequent is a singleton. A comparison of the actual output values versus the model's predicted output values is shown in Fig. 2. The model has an RMS error of 0.139 with respect to the training data and an RMS error of 0.811 with respect to the checking data, indicating the model is over specialized. The time required to generate the initial model (cluster estimation plus gradient descent optimization) was 54 seconds using C code on a Macintosh computer with 68040 processor running at 25MHz.

Using this initial model and the checking error criterion (Eq. 6) as the selection criterion, we then applied the variable selection method outlined in the previous section. The checking error as a function of the number of variables remaining in the model is shown in Fig. 3, where the name of the variable removed at each stage is indicated next to each error point. We see that $x(t\text{-}1)$ is the last variable to be removed; hence it is the most important variable for predicting $x(t)$, as we would expect. Note that the error value is not too large even when all input variables are removed; this is because the model then outputs a constant value which is the simple average of the rules' individual output values (i.e., all rules have degree of fulfillment of 1.0); this constant output value is usually close to the average value of the output data.

Although the variable elimination process reached a minimum checking error when 5 variables remain (after $u(t\text{-}6)$ was eliminated), we see that it may be desirable to eliminate two or three additional variables from the model, since the increase in error due to removing these additional variables is relatively small. The figure shows that we can eliminate the variables up to $x(t\text{-}2)$, keeping only the top two variables $x(t\text{-}1)$ and $u(t\text{-}4)$, and still maintain a small checking error. This result agrees well with the commonly accepted relevant variables for this problem: $x(t\text{-}1)$ and $u(t\text{-}4)$. In practice, it is essential to display this type of plot to the human designer and allow the designer to decide how many variables to retain in the model. The computation time for the variable selection process, including the time to generate the initial full model, was 67 seconds on the Macintosh computer.

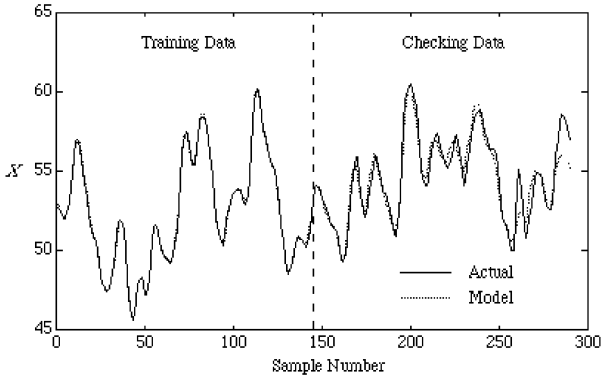

Fig. 2. Comparison of actual output values with the predicted output values for the 10-input initial model extracted from data set A.

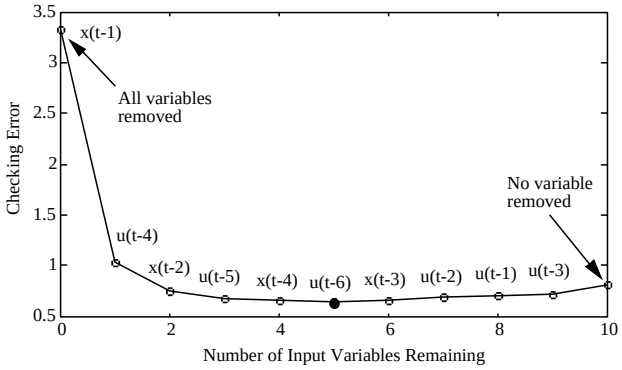

Fig. 3. Checking error as a function of number of input variables remaining in model A. The name of the variable that was removed at each stage is indicated next to the resultant error point. The point of minimum checking error is shown as a black circle.

To examine the sensitivity of the solution with respect to the choice of training and checking data, we next reversed the role of the data sets, using data set B as the training data and data set A as the checking data. An initial model consisting of 18 rules was extracted from data set B by using cluster radius r_a = 0.5. A comparison of the actual output versus the model's predicted output is shown in Fig. 4. The model has an RMS error of 0.190 with respect to the training data and an RMS error of 1.258 with respect to the checking data, again indicating the model is over specialized. The time required to generate this initial model (i.e., model B) was 1 minute 32 seconds on the Macintosh computer.

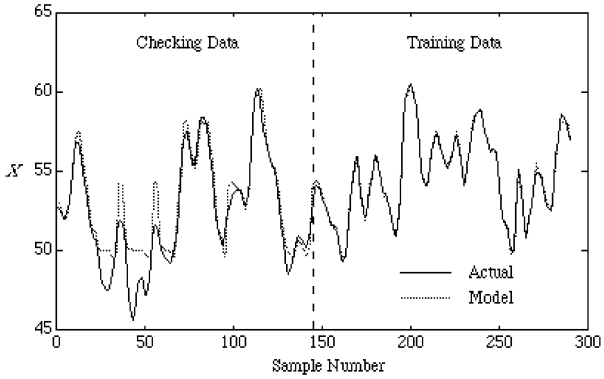

Fig. 4. Comparison of actual output values with the predicted output values for the 10-input initial model extracted from data set B.

Applying the variable selection method to this new initial model, we obtained the result shown in Fig. 5. Comparing Fig. 5 with Fig. 3, we see that the two solutions are different, but close. Both solutions agree that the dominant variable is $x(t\text{-}1)$, and the secondary and tertiary variables differ by one time step between the two solutions.

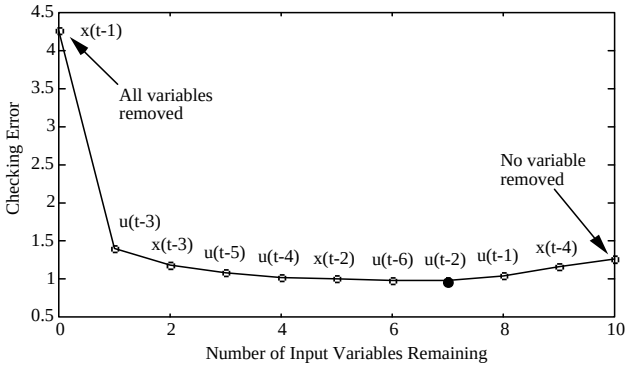

Fig. 5. Checking error as a function of number of input variables remaining in model B. The name of the variable that was removed at each stage is indicated next to the resultant error point. The point of minimum checking error is shown as a black circle.

The Box and Jenkins data is known to exhibit non-stationary behavior over the last forty or so samples [Young et al. 1971]; hence, a model built based on the last half of the data is expected to be more complex than a model built based on the first half of the data.

As shown in this example, the solution obtained by using the checking error criterion can easily change with the choice of data unless the training data and checking data are similar. However, carefully grouping the training data and checking data to ensure similarity defeats the purpose of having independent checking data to verify the model. In addition, the strong data dependency of the checking error criterion makes it a poor selection criterion when the data are noisy. Because of these drawbacks of the checking error criterion, many alternative criteria that are less sensitive to the choice of data have been proposed.

4.2 The Unbiased Criterion

The unbiased criterion (Eq. 7) selects the model structure whose models are least sensitive to changes in the training data. The unbiased criterion measures only the difference between the outputs of model A and model B and does not consider the prediction accuracy of the two models. Therefore, although its solution is least sensitive to the choice of data (and noisy data), the selected model structure often does not have high prediction ability.

Figure 6 shows the result of using the unbiased criterion for variable selection. We see that one of the commonly accepted important variables, $u(t\text{-}4)$, was the first variable eliminated; its surrogates, the closely correlated variables $u(t\text{-}2)$ and $u(t\text{-}3)$, were also eliminated before the criterion reached a minimum. This is because the unbiased criterion is reduced at each stage by removing the variable that best distinguishes the difference between the two models; unfortunately, such variable can be an important one.

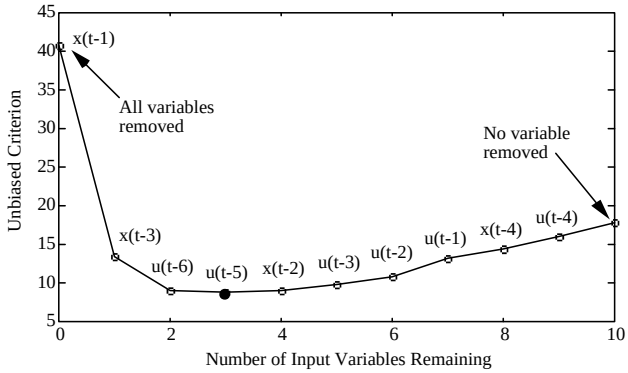

Fig. 6. Unbiased criterion as a function of number of input variables remaining in the models. The point at which the criterion reached a minimum is shown as a black circle.

4.3 The Regularity Criterion

The regularity criterion (Eq. 9) provides a good compromise between the strengths and deficiencies of the previous two criteria. The regularity criterion leads to a model structure with good prediction ability while maintaining a reasonable degree of insensitivity to the choice of data.

Fig. 7 shows the result of using the regularity criterion. As can be seen, this result is an amalgamation of the two different solutions produced by the checking error criterion. The computation time for variable selection using the regularity criterion, including the time to generate the two initial models, was exactly 3 minutes on the Macintosh computer.

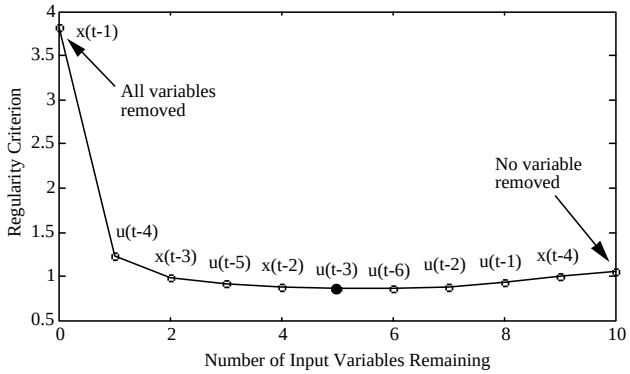

Fig. 7. Regularity criterion as a function of number of input variables remaining in the models. The point at which the criterion reached a minimum is shown as a black circle.

4.4 Comparison of Models

The different model structure selection criteria consistently identified $x(t\text{-}1)$ as the most important variable; however, they produced different answers for the ordering of the remaining variables. For each solution, we selected the top three variables and generated a Takagi-Sugeno type model that employs these three variables. To provide the same basis for comparison with other published results, each model was generated by using the entire set of 290 data points as training data. A cluster radius of 0.5 was used in all cases and the ANFIS approach was applied to optimize the rules; that is, the rule's premise membership functions were optimized by gradient descent while the consequent equations were optimized by linear least squares estimation. The rule optimization process stops when the change in RMS error after an epoch is less than 1.0E-5 times the previous RMS error. The performances of the resultant final models are compared in Table 1, where the model error shown is the mean square error with respect to the training data (i.e., all data), replicating the basis of other published performance results. The computation time for generating each model ranged from 22 seconds to 37 seconds on the Macintosh computer.

Variable Selection Criterion	Input Variables	Number of Rules	Model Error (mean square)
Checking Error Criterion (using Model A)	$x(t\text{-}1)$ $u(t\text{-}4)$ $x(t\text{-}2)$	3	0.0892
Checking Error Criterion (using Model B)	$x(t\text{-}1)$ $u(t\text{-}3)$ $x(t\text{-}3)$	3	0.0718
Unbiased Criterion	$x(t\text{-}1)$ $x(t\text{-}3)$ $u(t\text{-}6)$	3	0.1940
Regularity Criterion	$x(t\text{-}1)$ $u(t\text{-}4)$ $x(t\text{-}3)$	3	0.0942

Table 1. Comparison of models generated by using the top three input variables recommended by the different model structure selection criteria.

As shown in Table 1, the checking error criterion was superior in selecting variables with high prediction ability. The regularity criterion selected good variables, but we paid a small price for the increased insensitivity to training data. The model that used variables selected by the unbiased criterion fared the worst. Although the checking error criterion worked best for the Box and Jenkins data, we cannot extrapolate this result to other data. For noisy data and applications requiring long-range prediction of a time series, models that are less sensitive to the training data tend to provide better results [Farlow 1984].

We also generated two-input Takagi-Sugeno type models based on the top two input variables selected by each criterion. Again, each model was generated by using the entire set of 290 data points as training data and using a cluster radius of 0.5. Performances of the resultant models are compared in Table 2. As can be seen, the results are consistent with that of the three-input case.

Variable Selection Criterion	Input Variables	Number of Rules	Model Error (mean square)
Checking Error Criterion (using Model A)	$x(t\text{-}1)$ $u(t\text{-}4)$	3	0.156
Checking Error Criterion (using Model B)	$x(t\text{-}1)$ $u(t\text{-}3)$	3	0.146
Unbiased Criterion	$x(t\text{-}1)$ $x(t\text{-}3)$	3	0.235
Regularity Criterion	$x(t\text{-}1)$ $u(t\text{-}4)$	3	0.156

Table 2. Comparison of models generated by using the top two input variables recommended by the different model structure selection criteria.

The output predicted by the best three-input model, which employs $\{x(t\text{-}1),u(t\text{-}3),x(t\text{-}3)\}$, is shown in Fig. 8; and the output predicted by the best two-input model, which employs $\{x(t\text{-}1),u(t\text{-}3)\}$, is shown in Fig. 9. We see that both models have good prediction ability. The rules for these models are shown in Figures 10 and 11.

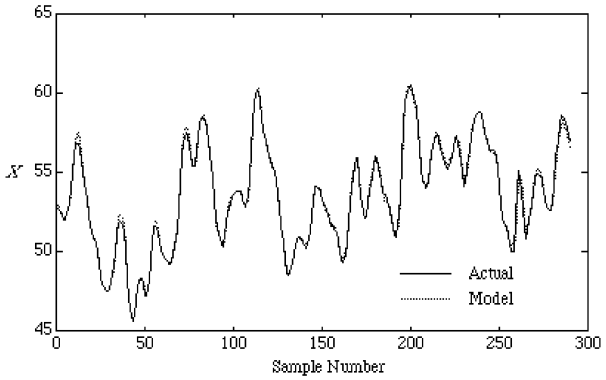

Fig. 8. Comparison of actual output with the output of the three-input model.

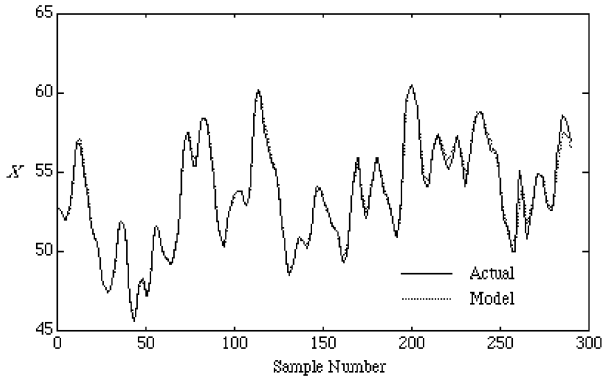

Fig. 9. Comparison of actual output with the output of the two-input model.

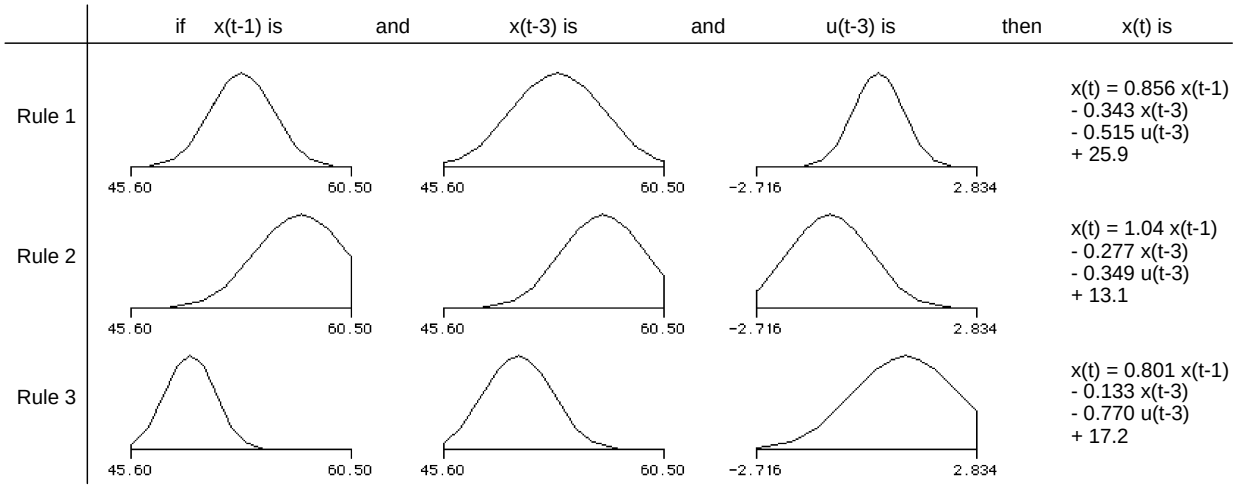

Fig. 10. Rules for the three-input model.

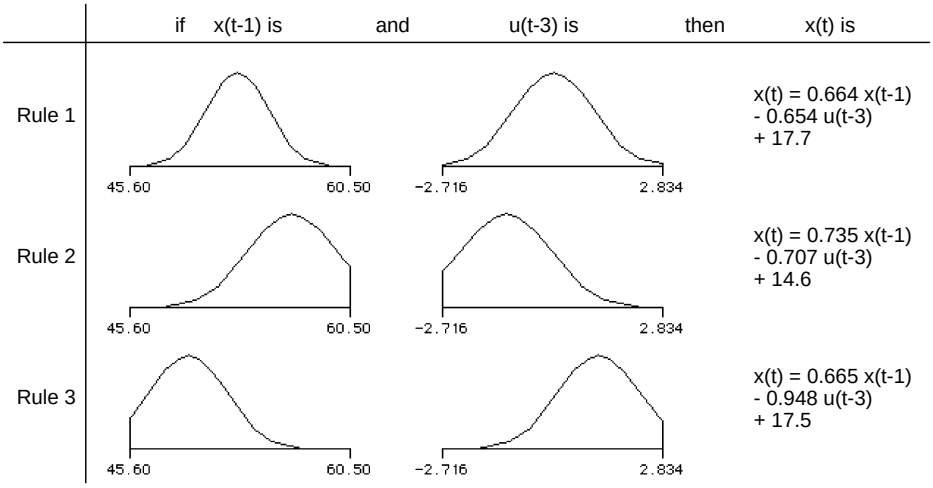

Fig. 11. Rules for the two-input model.

In Table 3, our best three-input and two-input models are compared with models obtained from other methods. It can be seen that our models provide performance comparable to the most accurate model but require much fewer input variables.

Model	Input Variables	Number of Rules	Model Error (mean square)
Tong's model [1980]	$x(t\text{-}1)$ $u(t\text{-}4)$	19	0.469
Pedrycz's model [1984]	$x(t\text{-}1)$ $u(t\text{-}4)$	81	0.320
Xu's model [Xu and Yong 1987]	$x(t\text{-}1)$ $u(t\text{-}4)$	25	0.328
Takagi-Sugeno model [Sugeno and Tanaka 1991]	$x(t\text{-}1)$ $x(t\text{-}2)$ $x(t\text{-}3)$ $u(t\text{-}1)$ $u(t\text{-}2)$ $u(t\text{-}3)$	2	0.068
Sugeno's position-gradient model [Sugeno and Yasukawa 1993]	$x(t\text{-}1)$ $u(t\text{-}4)$ $u(t\text{-}3)$	6	0.190
Our Takagi-Sugeno model [three-input]	$x(t\text{-}1)$ $u(t\text{-}3)$ $x(t\text{-}3)$	3	0.072
Our Takagi-Sugeno model [two-input]	$x(t\text{-}1)$ $u(t\text{-}3)$	3	0.146

Table 3. Comparison of various models derived for the Box and Jenkins gas furnace data. The first 5 rows were excerpted from a table in [Sugeno and Yasukawa 1993].

5. Conclusion

We presented an efficient method for selecting input variables when building a fuzzy model from data. This method overcomes a computational bottleneck: the need to generate a new model to test each combination of variables. By first generating an initial fuzzy model that incorporates all possible input variables, the fuzzy rule framework allows us to quickly test different simplified models without generating any new models. This methodology can be used in conjunction with different criteria for model structure selection.

We also presented a fast method for generating fuzzy models based on combining two existing techniques: subtractive clustering and ANFIS. Because the rules are found by clustering and not by dividing the input space into grid-like partitions, this method can generate good models that contain only a small set of rules irrespective of the dimension of the data space. This is particularly important when generating the initial fuzzy model from high dimensional data.

By using the Box and Jenkins gas furnace data as an example, we illustrated the input variable selection procedure as well as pointed out some issues regarding model selection criteria. No criterion is suited for all applications; the human designer must choose the appropriate criterion based on the characteristics of the data; it is also beneficial to explore the results of using different criteria.

The backward selection method employed here is a suboptimal search method that can be derailed in some cases (see [Cover and Campenhout 1977] for a discussion of how any suboptimal search method can potentially select the worst set of variables). Without resorting to exhaustive search, genetic algorithm holds promise as a more robust alternative search method [Siedelecki and Sklansky 1989]. Using genetic algorithm as the search mechanism is a topic for future work. Although the number of input variable combinations to be tested during search by genetic algorithm is significantly greater than during backward selection, our method for easily evaluating the different combinations will help to make the genetic algorithm approach practical.

References

H. Akaike (1974): “A new look at the statistical model identification,” IEEE Trans. Automatic Control, vol. 19, pp. 716-723.

T.W. Anderson (1984): *An Introduction to Multivariate Statistical Analysis*, 2nd edition, John Wiley & Sons, New York, NY.

G.E.P. Box and G.M. Jenkins (1970): Timer Series Analysis, Forecasting and Control, Holden Day, San Francisco, pp. 532-533.

S. Chiu (1994): “Fuzzy model identification based on cluster estimation,” J. of Intelligent and Fuzzy Systems, vol. 2, no. 3, pp. 267-278.

T.M. Cover and J.M. Van Campenhout (1977): “On the possible orderings in the measurement selection problem,” IEEE Trans. Systems, Man and Cybern., vol. 7, pp. 657-661

S.J. Farlow (1984): “The GMDH Algorithm,” in Self-Organizing Methods in Modeling: GMDH Type Algorithms, ed. S.J. Farlow, pp. 1-24, Marcel Dekker Inc., New York.

I. Hayashi, H. Nomura, H. Yamasaki, and N. Wakami (1992): “Construction of fuzzy inference rules by NDF and NDFL”, Int’l J. Approximate Reasoning, vol. 6, pp. 241-266.

A.G. Ivakhnenko, G.I. Krotov, and V.N. Visotsky (1979): “Identification of the mathematical model of a complex system by the self-organization method,” in Theoretical Systems Ecology: Advances and Case Studies, Chap. 13, E. Halfon (ed.), Academic Press, New York.

J.S.R. Jang (1993): “ANFIS: Adaptive-network-based fuzzy inference system,” IEEE Trans. on Systems, Man & Cybernetics, vol. 23, no. 3, pp. 665-685.

J.S.R. Jang and C.T. Sun (1993): "Functional equivalence between radial basis function networks and fuzzy inference systems," IEEE Trans. Neural Networks, vol. 4, no. 1, pp. 156-159.

Y. Lin and G.A. Cunningham III (1994): “A fuzzy approach to input variable identification,” Proc. 3rd IEEE Int'l Conf. on Fuzzy Systems, Orlando, FL, pp. 2031-2036.

T. Marill and D.M. Green (1963): “On the effectiveness of receptors in recognition systems,” IEEE Trans. Info. Theory, vol 9, pp. 11-17.

P.M. Narendra and K. Fukunaga (1977): “A branch and bound algorithm for feature subset selection,” IEEE Trans. Comp., vol. 26, pp. 917-922.

W. Pedrycz (1984): “An identification algorithm in fuzzy relational systems,” Fuzzy Sets and Systems, vol. 13, pp. 153-167.

W. Siedlecki and J. Sklansky (1989): “A note on genetic algorithms for large scale feature selection,” Pattern Recognition Letters, vol. 10, pp. 335-347.

M. Sugeno and G.T. Kang (1988): “Structure identification of fuzzy model,” Fuzzy Sets and Systems, no. 28, pp. 15-33.

M. Sugeno and K. Tanaka (1991): “Successive identification of a fuzzy model and its application to prediction of a complex system,” Fuzzy Sets and Systems, no. 42, pp. 315-334.

M. Sugeno and T. Yasukawa (1993): “A fuzzy-logic-based approach to qualitative modeling,” IEEE Trans. on Fuzzy Systems, vol. 1, no. 1.

T. Takagi and M. Sugeno (1985): “Fuzzy identification of systems and its application to modeling and control,” IEEE Trans. on Systems, Man & Cybernetics, vol. 15, pp. 116-132.

K. Tanaka, M. Sano, H. Watanabe (1995): “Modeling and control of carbon monoxide concentration using a neuro-fuzzy technique,” IEEE Trans. on Fuzzy Systems, vol. 3, no. 3, pp. 271-279.

R.M. Tong (1980): “The evaluation of fuzzy models derived from experimental data,” Fuzzy Sets and Systems, vol. 4, pp. 1-12.

L.X. Wang and J.M. Mendel (1992): “Back-propagation fuzzy system as nonlinear dynamic system identifiers,” Proc. 1st IEEE Int'l Conf. on Fuzzy Systems, San Diego, CA, pp. 1409-1418.

C.W. Xu and Z. Yong (1987): “Fuzzy model identification and self-learning for dynamic systems,” IEEE Trans. on Systems, Man & Cybernetics, vol. 17, no. 4, pp. 683-689.

R. Yager and D. Filev (1994): “Generation of fuzzy rules by Mountain Clustering,” J. of Intelligent and Fuzzy Systems, vol. 2, no. 3, pp. 209-219.

J. Yen, H. Wang and J. Liao (1993): “A method for automatic generation of a fuzzy model,” Proc. 3rd Int'l Conf. on Industrial Fuzzy Control and Intelligent Systems, Houston, TX, December 1993, pp. 88-92.

P.C. Young, S.H. Shellswell, and C.G. Neethling (1971): “A recursive approach to time series analysis,” report CUED/B-Control/TR16, Eng. Dept., Cambridge University.