Input Selection for Nonlinear Regression Models

Radek Šindelář and Robert Babuška

Abstract—**A simple and effective method for the selection of significant inputs in nonlinear regression models is proposed. Given a set of input–output data and an initial superset of potential inputs, the relevant inputs are selected by checking whether after deleting a particular input, the data set is still consistent with the basic property of a function. In order to be able to handle real-valued and noisy data in a sensible manner, fuzzy clustering is first applied. The obtained clusters are compared by using a similarity measure in order to find inconsistencies within the data. Several examples using simulated and real-world data sets are presented to demonstrate the effectiveness of the algorithm.**

Index Terms—**Fuzzy clustering, fuzzy modeling, input selection, regression models, similarity measures.**

I. Introduction

Real-world modeling problems in the fields like data mining or system identification involve a large number of potential inputs. The number of inputs actually used by the model must be reduced to the necessary minimum, especially when the model is nonlinear and contains many parameters (such as a fuzzy or a neuro-fuzzy model). Input selection is thus a crucial step with the aims of: 1) reducing the model's complexity, 2) removing inputs that depend on other inputs, 3) removing "noise" inputs that do not contribute to the output. For dynamic systems, the input-selection problem also includes the choice of the model's order (number of lagged inputs and outputs used as regressors) and the number of pure time delays.

Linear methods, like the analysis of correlations and auto-correlations, principal component analysis (PCA) and least-squares methods are well known tools in linear regression modeling. However, in real-world applications, which are almost always nonlinear, these tools often fail to discover the significant inputs. Hence, input selection methods for nonlinear systems have been studied. The methods found in the literature can generally be divided into two main groups.

- *Model-free methods*, which do not need to develop models in order to find significant inputs. These methods use the available data only and are based on statistical tests, properties of functions, etc. For instance, the method proposed in [1] exploits the continuity property of nonlinear functions. The so-called "Lipschitz coefficients" are computed in order to find the optimal *order* of input–output dynamic models. Another method, proposed in [2], is based on estimating the model performance directly from data. These methods are based on successively extending the regressor set by including additional lagged inputs and outputs. Hence, they cannot be applied to input selection in static problems or in dynamic problems in which some of the regressors are not expected to be included, due to transport delays, for instance.
- *Model-based methods*, which use a particular model in order to find the significant inputs. The Akaike's information criterion (AIC) [3] is often used for linear models. Models with different sets of input variables are compared and the model that minimizes the AIC is selected. Input selection for nonlinear systems is often based on heuristic criteria. A relatively simple and fast method was proposed in [4] for the ANFIS learning algorithm. Initially, a set of models whose inputs cover all potential candidates are generated. After one epoch of training, only models with the lowest root-mean-squared error (rmse) are developed in the next epoch because of their expected potential to achieve lower rmse after more epochs. Although this method was developed for the ANFIS fuzzy system, the same idea could possibly be used for other (neuro-) fuzzy systems. The authors of [5] proposed a method based on "fuzzy curves" that represent the sensitivity of the output with respect to the inputs. Also, methods described in [6]–[8] use the sensitivity of outputs with respect to inputs. In [8], the framework of neural networks is used and, in [7], the sensitivity analysis associated with the system's structure is exploited. The regularity criterion based on dividing a data set into two parts was used in [9]. An algorithm for finding relevant attributes is proposed in [10]. All the above-mentioned methods, except for [10], need to develop fuzzy models in order to compare them and select the relevant inputs. Such an approach is very time consuming and the result may be biased by the choice of the fuzzy model type and structure.

The method proposed in this article is model free. No specific model structure is assumed and the relevant inputs are found directly from input–output data by using geometric concepts and the basic property of a function. We apply fuzzy clustering to partition the data samples into fuzzy subsets and subsequently we use a similarity measure to quantify to what degree the data violate the property of a function. In this way, the potential inputs can be ordered according to their relative importance. The final interpretation of the result and the definitive selection of model inputs is done by the user. We found this technique to be less sensitive to noise and other disturbances in the data than the methods based on Lipschitz coefficients, which exploit the

Manuscript received November 21, 2001; revised February 21, 2003 and November 20, 2003. This work was supported in part by the Ministry of Education of the Czech Republic under Project LN00B096.

R. Šindelář is with the Center for Applied Cybernetics, Faculty of Electrical Engineering, Czech Technical University, 121 35 Prague 2, Czech Republic (e-mail: r.sindelar@c-a-k.cz).

R. Babuška is with the Delft Center for Systems and Control, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: r.babuska@dcsc.tudelft.nl).

Digital Object Identifier 10.1109/TFUZZ.2004.834810

continuity property of functions. In addition, an arbitrary subset of regressors can be analyzed by means of this technique. This is an advantage over methods based on Lipschitz coefficients [1] and false nearest neighbors [11], [12] which can only assess subsets of consecutive input and output lags.

This paper is organized as follows. First, the notation is introduced in Section II, which also gives a simple motivating example. Section III describes a fast clustering algorithm which is then used in the proposed input-selection technique. Section IV explains how the similarity of clusters is measured and also describes the algorithm. Section V gives a number of examples using both simulated and real-world data. Note that the problem of constructing fuzzy models is not addressed in this article.

II. Problem Statement

Many nonlinear static and dynamic processes can be represented by the following regression model [3]:

$$y(k) = f\left(\mathbf{x}(k)\right)$$

where the regression vector $\mathbf{x}(k)$ is given by

$$\mathbf{x}(k) = [u(k), \ldots, u(k-m+1), y(k-1), \ldots, y(k-n)].$$

The integer parameters m, n are related to the dynamic order of the system, $f\colon \mathbb{R}^r \to \mathbb{R}$ is a nonlinear function we wish to approximate. The total number of regressors is denoted by $r = m + n$. This model is known as the nonlinear auto-regressive model with exogenous input (NARX). The aforementioned single-input–single-output representation can be assumed without a loss of generality as the extension to multiple-input–single-output, multiple-input–multiple-output, and dead-time systems is straightforward.

Given the sequence of available input–output data pairs, $\mathcal{Z} = \{(u(i), y(i)) \mid i = 1, 2, \ldots, N\}$, we construct the *regressor matrix* and the *regressand vector* according to (1), as shown at the bottom of the page, where $l_m = \max(n, m) + 1$, $\mathbf{X} \in \mathbb{R}^{(N-l_m+1)\times r}$ and $\mathbf{Y} \in \mathbb{R}^{(N-l_m+1)\times 1}$. Further, we denote by $\mathbf{x}_i$ the ith row of $\mathbf{X}$, y_i the ith row (element) of $\mathbf{Y}$ and x_{ij} the jth element of $\mathbf{x}_i$. The aim of the modeling exercise is to approximate the unknown function f. It is assumed in this paper that the inputs used to obtain the data sequence $\mathcal{Z}$ are persistently exciting [3] up to the order of the dynamic system.

Define the matrix $\mathbf{Z} \in \mathbb{R}^{(N-l_m+1)\times(r+1)}$ as a concatenation of $\mathbf{X}$ and $\mathbf{Y}$

$$\mathbf{Z} = [\mathbf{X}\,\mathbf{Y}]. \tag{2}$$

For the moment, suppose that the relation between the inputs and the output is a function (i.e., there is no noise in the data). The main idea of the algorithm proposed in this paper stems from the following basic property of a function.

Property 1: If f is a function such that $y_i = f(\mathbf{x}_i)$, $y_j = f(\mathbf{x}_j)$, then $\mathbf{x}_i = \mathbf{x}_j \Rightarrow y_i = y_j, \forall i, j$

It is said that the matrix $\mathbf{Z}$ is *consistent* if Property 1 holds for all pairs of its rows. It is assumed that all inputs needed to construct the model are initially contained in $\mathbf{X}$. Besides these inputs, $\mathbf{X}$ also may contain variables that do not contribute to the output. The potential inputs can thus be divided into two subsets: *relevant inputs*, which are essential for describing the functional relationship between the input and output variables, and *redundant inputs*, which do not influence the output. The basic idea is to find the most relevant inputs by successively removing columns from $\mathbf{X}$ and checking whether the reduced data set is still consistent (i.e., Property 1 holds). If a relevant variable is removed, Property 1 can be violated and the data set is no longer consistent.

Denote by χ the number of distinct rows in $\mathbf{Z}$ that violate Property 1, i.e., rows for which it holds that:

$$\mathbf{x}_i = \mathbf{x}_j \wedge y_i \neq y_j, \qquad i \neq j.$$

Further, denote by M the number of distinct rows of $\mathbf{Z}$. This number indicates how many independent data samples are available to estimate f.

After removing column i from $\mathbf{Z}$ (the ith input variable is removed from the input space), a new matrix $\mathbf{Z}_i$ is obtained. The number of distinct rows in $\mathbf{Z}_i$ is denoted by M_i and the number of rows violating Property 1 is denoted by χ_i. One of the following three situations will occur.

a) $M_i = M$ and $\chi_i = \chi$. This means that the removed input i is redundant and can definitely be left out.
b) $M_i < M$ and $\chi_i = \chi$. This means that the removed input i is redundant and its removal leads to the reduction of the data set size (which is a more favorable situation than case a).
c) $M_i < M$ and $\chi_i > \chi$. This input is significant and may not be removed from the input space.

Now, we are ready to formulate the basic (naive) version of the input-selection algorithm. We call it naive because it only works for noise-free data and it assumes that the (in)equality of data vectors can be established. In the sequel, we extend the algorithm to the realistic situation with noisy real-valued data.

$$\mathbf{X} = \begin{bmatrix} u(l_m) & \ldots & u(l_m-m+1) & y(l_m-1) & \ldots & y(l_m-n) \\ u(l_m+1) & \ldots & u(l_m-m+2) & y(l_m-2) & \ldots & y(l_m-n-1) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ u(N) & \ldots & u(N-m+1) & y(N-1) & \ldots & y(N-n) \end{bmatrix}$$

$$\mathbf{Y} = \begin{bmatrix} y(l_m) \\ y(l_m+1) \\ \vdots \\ y(N) \end{bmatrix} \tag{1}$$

TABLE I
RESULTS OBTAINED IN THE INDIVIDUAL ITERATIONS OF THE ALGORITHM (EXAMPLE 1)

Iteration		x_1	x_2	x_3	x_4	x_5	$2x_2+x_1$	$2x_5$	x_{r1}	x_{r2}	x_{r3}
1	χ_i	0	0	12	13	0	0	0	0	0	0
	M_i	497	497	496	497	497	497	497	484	490	482
2	χ_i	0	0	35	41	0	0	0	0	0	×
	M_i	482	482	475	482	482	482	482	434	450	×
3	χ_i	0	0	78	96	0	0	0	×	0	×
	M_i	434	434	417	434	434	434	434	×	346	×
4	χ_i	0	0	121	159	0	0	0	×	×	×
	M_i	346	346	313	346	346	346	346	×	×	×
5	χ_i	0	0	121	159	154	0	×	×	×	×
	M_i	346	346	313	346	346	346	×	×	×	×
6	χ_i	129	129	121	159	154	×	×	×	×	×
	M_i	320	320	313	346	346	×	×	×	×	×

TABLE II
RESULTS OBTAINED IN THE INDIVIDUAL ITERATIONS OF THE ALGORITHM (EXAMPLE 1)

Iteration		$2x_2+x_1$	x_2	x_3	x_4	x_5	x_1	$2x_5$	x_{r1}	x_{r2}	x_{r3}
1	χ_i	0	0	12	13	0	0	0	0	0	0
	M_i	497	497	496	497	497	497	497	484	490	482
2	χ_i	0	0	35	41	0	0	0	0	0	×
	M_i	482	482	475	482	482	482	482	434	450	×
3	χ_i	0	0	210	255	0	0	0	×	0	×
	M_i	434	434	417	434	434	434	434	×	346	×
4	χ_i	0	0	121	159	0	0	0	×	×	×
	M_i	346	346	313	346	346	346	346	×	×	×
5	χ_i	0	0	121	159	154	0	×	×	×	×
	M_i	346	346	313	346	346	346	×	×	×	×
6	χ_i	129	48	121	159	154	×	×	×	×	×
	M_i	320	334	313	346	346	×	×	×	×	×

Basic Input-Selection Algorithm

1. Construct an initial data set $\mathbf{Z}$ such that $\mathbf{X}$ certainly contains the relevant inputs (and possibly many redundant inputs). For this data set, compute M and χ.
2. For $i = 1, 2, \ldots, r$, create $\mathbf{Z}_i$ from $\mathbf{Z}$ and compute the corresponding M_i and χ_i.
3. Find the index i for which it holds that $\chi_i = \chi$ and M_i is minimal. If there is no such i, stop, otherwise permanently remove the ith column from $\mathbf{Z}$, decrement r and remove from $\mathbf{Z}$ also the redundant rows. Go to step 2.

It is important to realize that the input variables must be removed iteratively one by one. If there are, for instance, two columns containing the same information (e.g., they are linearly dependent), in the first iteration both will seemingly appear as redundant. The importance of one of them will only become apparent after removing the other one.

Ties in step 3) of the previous algorithm are resolved by removing the column with the largest index. In this way, the user can include prior knowledge or impose certain preference by arranging the potential regressors $\mathbf{Z}$ in the order of decreasing importance. If there is no preference, ties can be resolved at random.

Furthermore, note that the data pairs causing the inconsistency do not necessarily have to be present in the data set. This will typically be the case when the number of inputs is large. In order to reduce this problem, the input with the lowest M_i (for which $\chi_i = \chi$) is permanently removed. This quickly reduces the size of the data set and the chance of discovering inconsistencies increases. Let us illustrate this algorithm with the help of two simple examples.

Example 1: Consider the following nonlinear static function:

$$y = 10 \sin\left(\frac{\pi}{10}x_1\right) x_2 + 2(x_3 - 0.5)^2 + 3x_4 + 5x_5.$$

The potential input variables are selected as follows:

$$\mathbf{x} = [\, x_1,\, x_2,\, x_3, x_4,\, x_5,\, x_1 + 2x_2,\, 2x_5,\, x_{r1},\, x_{r2},\, x_{r3}]$$

where x_{ri} are additional uniform random variables. The data set contains 500 uniformly randomly generated samples where $x_i \in \{0, 1, 2, 3\}$. Results obtained after the individual iterations are shown in Table I. The symbol "×" means that the input was removed from the data set in that particular iteration. Note that the algorithm indeed identified the true inputs of the model.

However, if we change the order of the input variables in $\mathbf{Z}$ in the following way:

$$\mathbf{x} = [\, x_1 + 2x_2,\, x_2,\, x_3,\, x_4,\, x_5,\, x_1,\, 2x_5,\, x_{r1},\, x_{r2},\, x_{r3}]$$

the results are different as one can see from Table II. Instead of input variables x_1 and x_2, their linear combination was chosen. This is because the data set is relatively small and this choice did

TABLE III
RESULTS OBTAINED IN THE INDIVIDUAL ITERATIONS OF THE ALGORITHM (EXAMPLE 2)

Iteration		$u(k)$	$u(k-1)$	$u(k-2)$	$u(k-3)$	$y(k-1)$	$y(k-2)$	$y(k-3)$
1	χ_i	317	0	253	0	0	0	0
	M_i	1072	1061	1145	1131	1544	1346	1272
2	χ_i	247	×	198	0	62	0	0
	M_i	641	×	685	522	819	790	746
3	χ_i	132	×	104	×	56	0	0
	M_i	278	×	295	×	302	310	265
4	χ_i	73	×	56	×	34	0	×
	M_i	132	×	131	×	131	124	×
5	χ_i	33	×	29	×	20	×	×
	M_i	57	×	52	×	56	×	×

not result in any conflicts in the data. A detailed discussion of the limitations of the proposed approach is given later in Section VI.

Example 2: Consider the following nonlinear dynamic system:

$$y(k) = \lceil 0.1\left(y(k-1)^2 + u(k)u(k-2)\right)\rceil \tag{3}$$

where $\lceil\cdot\rceil$ means rounding the argument toward the nearest integer. The inputs and outputs can only attain integer values from the following set: $u, y \in \{0,1,2,3,4,5\}$. The goal is to find the regressors in (3), given only the time series of input–output data pairs $\{(u(k), y(k)) \mid k = 1, 2, \ldots, 2000\}$ obtained by simulating the system. The initial matrix $\mathbf{Z}$ was constructed by using the regressors for $m = 4$ and $n = 3$. Table III shows the results obtained in five iterations of the algorithm. One can see that inputs $u(k)$, $u(k-2)$ and $y(k-1)$ were selected as the most significant ones. This result corresponds to the original system (3).

This basic algorithm has several limitations.

1. The data must be discrete in order to establish (in)equality of the individual data items. The discretization, although theoretically feasible, causes a number of problems. If we denote by d the number of discrete levels in the inputs, the number of possible points in the input space is d^r. To populate such a large space with data, a very good excitation of the system is necessary. For instance, in Example 1, the total number of discrete points is $6^9 \approx 10^7$. The data set that was used in this example had 500 samples, which is a tiny fraction of the total number. Such a data set will certainly not cover the function hypersurface with a sufficient density. In such a case, it can then happen that $\chi_i = \chi$ and $M_i = M$ for each i and no decision can thus be taken in step 2) of the basic input-selection algorithm.
2. The approach is time and memory consuming. The complexity rapidly increases with the number of candidate inputs and the number of discrete values. The number of all possible subsets of the inputs is 2^r.
3. Real data are often corrupted by noise. In such a case, Property 1 will be violated even if the removed input is not significant (actually it will be violated already in the initial data set).

To make the method practically useful, the basic algorithm must be modified. Rather than comparing discrete points in the product space we will use fuzzy clustering and similarity measures to assess the (approximate) equality of data samples.

III. FAST FUZZY CLUSTERING ALGORITHM

To cluster the data, a number of standard algorithms can be used. The most important criteria for the choice of a suitable algorithm are the ability to quickly cluster a large number of data samples (up to say tens of thousands) into a relatively large number of clusters (around 10% of the number of data samples). The number of clusters is not known a priori. In this sense, clustering algorithms based on the minimization of an objective function (such as the c-means algorithm) are not suitable.

We use an algorithm similar to the self-organizing Kohonen network [8]. The data set is normalized such that all variables are contained in the unit interval: $z_{jk} \in [0, 1]$. The clusters are represented by Gaussian membership functions of the following form:

$$\mu_{ik} = \exp\left(-\sum_{j=1}^{r}\left(\frac{m_{ji} - z_{jk}}{\sigma_{ji}}\right)^2\right) \tag{4}$$

where μ_{ik} is the membership degree of data point $\mathbf{z}_k$ in cluster i, $\mathbf{m}_i$ is the ith cluster center and σ_{ji} is the cluster width along the jth feature. It is assumed that the data samples are vectors in a metric space. If this is not the case, a suitable dissimilarity measure can be defined instead of the distance.

The algorithm processes the input patterns one by one. Initially, the first input sample defines the first cluster center and the initial width of this cluster is set to some default value (a user-defined parameter). For each pattern, the algorithm then checks whether the sample belongs to a sufficient degree to some existing cluster. If this is the case, the pattern is added to that cluster and the cluster center and volume are changed to account for this new input pattern. Otherwise, a new cluster is created.

```
**Fast Clustering Algorithm**
Set the parameters $\mu_{\min}$, $\alpha_c$, $\alpha_e$, $\sigma_{\min}$, $\sigma_{\max}$
and create the first cluster: $\mathbf{m}_1 := \mathbf{z}_1$,
$\boldsymbol{\sigma}_1 := \boldsymbol{\sigma}_{\text{init}}$, $N_1 := 1$, $c := 1$.
Repeat for $k = 2, \ldots, N$
```

TABLE IV
COMPARISON OF COMPUTATION TIME (IN SECONDS) FOR DIFFERENT NUMBERS OF CLUSTERS

Number of clusters	FCM	Fast algorithm
10	4.20	1.46
20	7.75	1.49
40	15.12	1.57
80	29.83	1.63
120	44.43	1.65
200	76.62	1.81
300	130.74	2.03

```
Compute $\mu_{ik}$, $\forall i$ by using (4) and find the closest cluster:

    $I = \arg \max_i \mu_{ik}$

if $\mu_{Ik} \geq \mu_{\min}$, add the input pattern to cluster $I$ and modify the cluster's parameters:
$\mathbf{m}_I := (\mathbf{m}_I N_I) + \mathbf{z}_k / N_I + 1 \quad N_I := N_I + 1$,
for $j = 1, \ldots, r$

    $\sigma_{jI} := \begin{cases} \min(\alpha_e \sigma_{jI}, \sigma_{\max}), & \text{if } \exp[-\frac{(m_{jk}-z_{jk})^2}{\sigma_{jI}^2}] > 0.5 \\ \max(\alpha_c \sigma_{jI}, \sigma_{\min}), & \text{otherwise} \end{cases}$

else create a new cluster:
$c := c + 1$, $\mathbf{m}_c := \mathbf{z}_k$, $\sigma_c := \sigma_{\text{init}}$, $N_c = 1$.
```

The threshold $\mu_{\min} \in (0, 1)$ is used to find out whether a given input pattern belongs to an existing cluster. Along with the maximal width $\sigma_{\max}$ and the contraction/expansion parameter α, it indirectly determines the number of clusters. Larger $\mu_{\min}$ and smaller $\sigma_{\max}$ result in more clusters. The default setting for σ_{init} is $0.75\sigma_{\max}$.

In Tables IV and V, the fast clustering algorithm and the fuzzy c-means (FCM) algorithm are compared on the data set from the heat-transfer process presented in Section V-D. This data set contains 1309 samples in eight dimensions. The standard fuzzy c-means algorithm was used with the fuzziness exponent $m = 2$, a random initial position of cluster centers and the algorithm stopped when the number of iterations $l \geq 100$ or when

$$||U^l - U^{l-1}|| < 0.01$$

where U^l is the partition matrix in the l^{th} iteration. Table IV shows the computational time demand (in MATLAB) as a function of the number of clusters. Table V presents the time demand for different numbers of data samples clustered in 70 clusters. One can see that the proposed algorithm is considerably more effective for large data sets and large numbers of clusters.

IV. INPUT SELECTION ALGORITHM

First, the initial data matrix $\mathbf{Z}$ given by (2) is clustered, using the algorithm described in the previous section. Note that the data space is the complete product space of all the regressors and the regressand. As a result, the clustering algorithm provides for each cluster its center, width, and the number of data samples N_i contained in that cluster. Now, instead of establishing the equality of the individual data items, the degree of similarity among the individual clusters is computed. To assess the similarity, we use the inclusion measure defined for two fuzzy sets A and B by

TABLE V
COMPARISON OF COMPUTATION TIME (IN SECONDS) FOR DIFFERENT NUMBERS OF SAMPLES

Number of samples	FCM	Fast algorithm
100	3.69	1.47
250	6.67	1.61
500	10.58	1.87
750	15.68	1.98
1000	21.39	2.23

$$I_a(A, B) = \frac{|A \cap B|}{|A|} \qquad I_b(A, B) = \frac{|A \cap B|}{|B|}.$$

The similarity of these two fuzzy sets is then given by

$$S = \max(I_a, I_b). \tag{5}$$

As it is computationally prohibitive to compare all pairs of clusters, clusters that are close to one another are first selected. The clustering algorithm (see Section III) defines the maximal cluster width and therefore it is easy to find the distance of cluster centers above which the clusters cannot not be sufficiently similar. These dissimilar clusters are not considered further.

The similarity in the input space and the similarity in the output space are computed separately. Similarity of clusters in one dimension is computed according to (5). The total similarity in the given space is

$$S = \prod_{j=1}^{r} S^j$$

where S^j is the similarity of two clusters in jth dimension and r is the dimension of the space. The product operator in the above equation is a natural choice given that the Gaussian membership functions are used (the total membership degree is the product of membership functions in the individual dimensions).

The basic algorithm described in Section II uses the variable M, which quantifies the amount of information that is available to discover the function relation between the input and output patterns. Here, the total similarity S is introduced that has a similar meaning as M:

$$S = S^x \, S^y. \tag{6}$$

Using the obtained clusters, the inconsistency in the data set is defined as follows.

Clusters that are similar in the input space and not similar in the output space are not consistent with Property 1.

As this definition is fuzzy, a fuzzy decision function is used (see Fig. 1). The three corners in which the decision surface reaches zero represent the following situations.

- Clusters are not similar in the input space and are not similar in the output space.

TABLE VI
INCONSISTENCY INDEX χ AND THE NUMBER OF CLUSTERS c FOR EXAMPLE V-A

Iteration	u(k)	u(k-1)	u(k-2)	u(k-3)	u(k-4)	y(k-1)	y(k-2)	y(k-3)	y(k-4)	c
1	0.7479	0.0202	0.2273	0	0.0046	0.7610	0.2350	0.0041	0	66

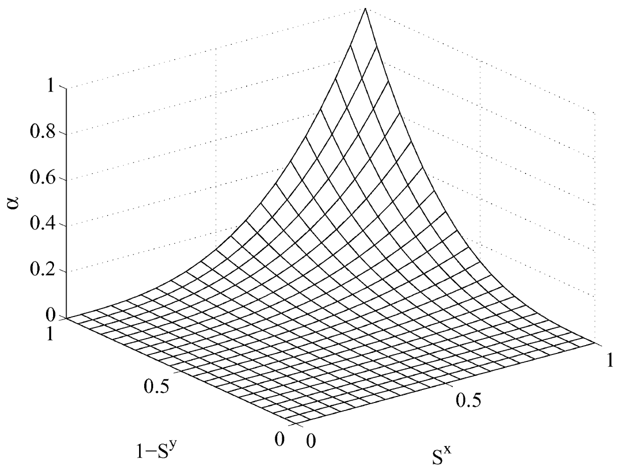

Fig. 1. Decision surface $(S^x(1-S^y))^\lambda$ for $\lambda = 3$.

- Clusters are not similar in the input space and are similar in the output space.
- Clusters are similar in the input space and are similar in the output space.

The fourth corner represents the situation when the clusters in the input space are very similar but they are very dissimilar in the output space, i.e., Property 1 does not hold. Mathematically, the decision function is defined as follows:

$$\alpha = (S^x \cdot (1 - S^y))^\lambda \cdot \min(N_A, N_B) \tag{7}$$

where S^x and S^y are the similarity in the input and output space, respectively, N_A and N_B are the number of data samples in clusters A and B, respectively, and λ is a parameter defining the curvature of the fuzzy decision surface such that inconsistencies are emphasized. The value of λ is usually set to 3 to 4. The weighting by $\min(N_A, N_B)$ is involved because the number of patterns in the individual clusters can significantly differ.

The choice of the relevant inputs is based on the ratio between the inconsistency given by α and the total similarity S, given by (6). Removing a significant input causes more inconsistency in the corresponding matrix $\mathbf{Z}_i$. On the other hand, the removal of a less significant input causes the increase of the similarity index S. The total inconsistency α_i for the given $\mathbf{Z}_i$ is computed by adding up the outcomes of (7) for all pairs of similar clusters. The (relative) *inconsistency index* is then defined as the following ratio:

$$\chi_i = \frac{\alpha_i - \alpha}{S_i - S}, \qquad i = 1, 2, \ldots, r. \tag{8}$$

The resulting χ_i are then normalized such that their sum equals to one. For dynamic input–output models, this normalization is done separately for the input and output regressors.

Now, we are able to formulate the clustering-based input selection algorithm.

Input-Selection Algorithm

1. Construct an initial data set $\mathbf{Z}$ such that $\mathbf{X}$ certainly contains the relevant inputs and possibly many redundant inputs.
2. Cluster the data in $\mathbf{Z}$ and compute χ. For $i = 1, 2, \ldots, r$, create $\mathbf{Z}_i$ from $\mathbf{Z}$ and compute the corresponding normalized χ_i.
3. Find the index i for which χ_i is minimal. Remove permanently the ith column from $\mathbf{Z}$ and decrement r. Go to step 2.

Note that this input selection algorithm does not include any termination condition. The interpretation of the relative differences in χ_i is left to the user. Generally, one stops removing regressors from the data when the distribution of χ_i over the regressors does no longer change significantly. The examples presented in the next section give an illustration of the typical results that one can obtain.

V. EXAMPLES

This section presents some typical results of an extensive experimental evaluation that was done for the proposed method. Data sets from two simulated and two real-world systems are used. The default settings of the clustering algorithm are $\alpha_e = 1$, $\alpha_c = 1$ (clusters cannot expand or contract), $\sigma_{\max} = 0.1$ and $\mu_{\min} = 0.1$. The cluster size parameters $\sigma_{\max}$ and $\sigma_{\min}$ may need to be adjusted for the problem at hand. The decision surface parameter λ has been set to 4 in all the examples. For the first example, we show a comparison with the method based on Lipschitz coefficients.

A. Simulated Nonlinear System

Consider the following nonlinear dynamic system:

$$y(k) = \sin(2.3u(k)u(k-2)) + e^{u(k)} - u(k-2)^{5/2} + y(k-1). \tag{9}$$

The data set was obtained by simulating the above system for an input sequence of 300 samples generated randomly according to the normal distribution with a zero mean and a unit variance. The default settings work fine with this data set, but as this is a simulated system with noise-free data, we may increase the maximum cluster size to, e.g., $\sigma_{\max} = 0.2$ in order to reduce the number of clusters. The following initial set of regressors was assumed:

$$\begin{aligned}\mathbf{x}(k) = [&u(k), u(k-1), u(k-2), u(k-3), u(k-4)\\ &y(k-1), y(k-2), y(k-3), y(k-4)]. \end{aligned} \tag{10}$$

The results obtained after the first iteration of the algorithm are given in Table VI. The last column shows the number of clusters. Regressors $u(k)$, $u(k-2)$ and $y(k-1)$ can clearly be selected as the three most important ones.

For a comparison, Table VII shows the Lipschitz coefficients computed by using the method proposed in [1]. The row and

TABLE VII
LIPSCHITZ COEFFICIENTS FOR EXAMPLE V-A

Regressors	u(k)	u(k-1)	u(k-2)	u(k-3)	u(k-4)	u(k-5)	u(k-6)	u(k-7)
y(k-1)	15.58	3.72	2.44	1.63	1.38	1.16	1.04	0.96
y(k-2)	5.80	2.52	1.88	1.43	1.25	1.08	0.99	0.95
y(k-3)	3.72	1.98	1.62	1.30	1.16	1.04	0.95	0.89
y(k-4)	2.75	1.66	1.42	1.20	1.10	0.98	0.91	0.86
y(k-5)	2.20	1.38	1.21	1.08	1.00	0.93	0.87	0.82
y(k-6)	1.87	1.27	1.14	1.03	0.96	0.90	0.84	0.80
y(k-7)	1.63	1.09	1.01	0.95	0.90	0.85	0.80	0.76
y(k-8)	1.48	1.04	0.97	0.92	0.86	0.82	0.77	0.73

TABLE VIII
INCONSISTENCY INDEX χ AND THE NUMBER OF CLUSTERS c FOR EXAMPLE V-B

Iteration	u(k)	u(k-1)	u(k-2)	u(k-3)	u(k-4)	y(k-1)	y(k-2)	y(k-3)	y(k-4)	c
1	0.0194	0.0449	0.7941	0.0989	0.0427	0.8024	0.0989	0.0563	0.0424	269
2	$\times$	0.0412	0.8816	0.0494	0.0279	0.7149	0.0861	0.1338	0.0651	189
3	$\times$	0.1007	0.7160	0.1349	0.0484	0.9000	0.0638	0.0362	$\times$	145

TABLE IX
INCONSISTENCY INDEX χ AND THE NUMBER OF CLUSTERS c FOR EXAMPLE V-C

Iteration	u(k)	u(k-1)	u(k-2)	u(k-3)	u(k-4)	y(k-1)	y(k-2)	y(k-3)	y(k-4)	c
1	0.1216	0.6048	0.1785	0.0127	0.0825	0.4627	0.2016	0.0981	0.2376	36
2	0.2207	0.4734	0.2117	$\times$	0.0942	0.5184	0.1781	1197	1837	30
3	0.1675	0.4848	0.3426	$\times$	0.0051	0.7623	0.2215	$\times$	0.0161	31

column headings in the table denote the individual model structures. For instance, the coefficient in the row labeled $y(k-1)$ and column labeled $u(k-2)$ is calculated for the model structure $y(k) = f(u(k), u(k-1), u(k-2), y(k-1))$. To determine the order of the system, one has to look for a sharp decrease in the coefficients. We can see that the decrease is quite gradual throughout the entire table. The largest gap is between the first-order model (15.58) and the second-order model $y(k) = f(y(k-1), y(k-1), u(k), u(k-1))$, but referring to (9) we see that the first-order model is not the correct structure. Another limitation of the Lipschitz coefficients method is also clear from the example: the method can possibly select the order of the system (the largest lag for the inputs and outputs), but not an arbitrary subset of regressors in which some lags are missing (such as $u(k-1)$ in this example).

B. Simulated Transport-Delay System With Sensor Noise

This example shows that the proposed algorithm is able to detect systems with a transport delay and noise. The nonlinear system given by (11) and (12) was simulated in MATLAB using 1000 randomly generated input samples (normal distribution with a zero mean and a unit variance).

$$y_0(k) = \frac{-2u(k-2)u(k-3)}{y(k-1)} + e^{(-u(k-2)y(k-1))} - u^2(k-3). \quad (11)$$

The output of the system $y(k)$ is restricted to the interval $(-1,3)$ and

$$y(k) = y_0(k) + \epsilon(t) \quad (12)$$

where $\epsilon(t)$ is a nonstationary output noise $\epsilon \sim N(0, \sigma)$, with $\sigma = 0.1|y|$. The initial regressors were the same as in (10). The results after three iterations of the algorithm are given in Table VIII. Although the elimination process could continue, here we suppose that the algorithm stops at this point. Regressors $u(k-2)$ and $y(k-1)$ are clearly the most important ones. Regressor $u(k-3)$ could be included in the model as well, but note that the difference between the χ values for $u(k-3)$ and $u(k-1)$ is rather small (which means that $u(k-1)$ should then be included as well). This is caused by the presence of noise, which makes the clusters fuzzier and therefore more similar to each other. This blurs the result as one can see by comparing Table VIII with Table VI, in which a much larger difference in the χ indices is observed. A similar phenomenon occurs with standard linear techniques based on eigenvalues of the data covariance matrix, for instance.

C. Pressure Dynamics in a Fermeter

In this example, a real data set obtained from a laboratory fermenter is used. The goal is to identify a nonlinear model for the pressure dynamics in the fermenter's head-space. The input of the model is the position of the outlet valve and the output is the pressure in the head-space. A more detailed description of the system can be found in [13], [14].

The data set contains 757 data samples. The initial regressors are again selected as in (10). The results of the input-selection algorithm are given in Table IX. Regressors $u(k-1)$ and $y(k-1)$ seem to be the most important ones after the first iteration. However, as the differences in χ for the inputs $u(k-i)$ are very small, additional iterations of the algorithm were executed. After the first iteration, regressor $u(k-3)$ was removed and after the second iteration, regressor $y(k-3)$ was removed. After the third iteration, regressors $u(k-1)$, $u(k-2)$ and $y(k-1)$ are the most important ones ($y(k-2)$ could possibly be included as well). The resulting model has the following structure:

$$y(k) = f(u(k-1), u(k-2), y(k-1)).$$

TABLE X
INCONSISTENCY INDEX χ AND THE NUMBER OF CLUSTERS c FOR EXAMPLE V-D

Iteration	u(k-9)	u(k-8)	u(k-7)	u(k-6)	u(k-5)	u(k-4)	u(k-3)	u(k-2)	u(k-1)	u(k)	y(k-1)	y(k-2)	y(k-3)	y(k-4)	c
1	0.107	0.091	0.162	0.177	0.227	0.069	0.018	0.024	0.026	0.100	0.329	0.267	0.219	0.185	592
2	0.107	0.095	0.165	0.166	0.245	0.076	×	0.022	0.028	0.094	0.318	0.267	0.224	0.191	555
3	0.105	0.097	0.177	0.164	0.234	0.059	×	0.029	0.031	0.100	0.372	0.328	0.303	×	549
4	0.100	0.094	0.179	0.222	0.213	0.070	×	×	0.041	0.097	0.355	0.332	0.313	×	520

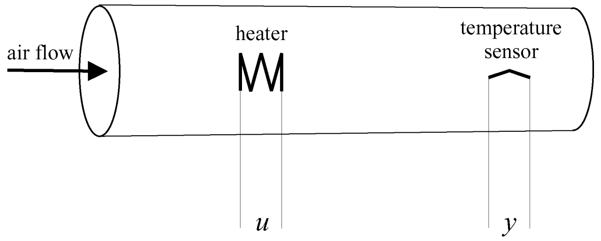

Fig. 2. Schematic diagram of the heat transfer process.

Simulations with a Takagi–Sugeno fuzzy model using these variables give a slightly better result than the first-order model $y(k) = f(u(k-1), y(k-1))$ used in [14].

D. Heat-Transfer Process

This example shows an application of the proposed algorithm to the laboratory heat-transfer system process (PT326, manufactured by Feedback), which is a typical example of a dynamic process with a significant transport delay. This process consists of a fan that blows air through a tube (Fig. 2). The air flow-rate can be manually adjusted by a damper. At the inlet of the tube, a heating resistor is mounted (similar to a hair dryer). The manipulated input to the system is the voltage applied to the power electronic circuit feeding the heating resistor. A temperature sensor is placed at the other end of the tube. The output of the system is the voltage given by this sensor. An identification data set was obtained by randomly manipulating the input in the interval 0 to 10 V. The total number of samples is $N = 1309$, with the sampling period of 1.1 s. For details, see [15] and [14].

The results obtained after the first iteration are shown in Table X. The most significant inputs are $u(k-5)$ and $y(k-1)$. The least significant input $u(k-3)$ is removed and the new data set is checked again. After the second iteration it is clear that the process contains a transport delay of five samples and third-order dynamics (which is a very realistic approximation of the system under study). According to this result, the model structure is

$$y(k)=f(u(k-5),u(k-6),u(k-7),y(k-1),y(k-2), y(k-3)).$$

VI. CONCLUSION

An effective method for selecting input variables has been presented. This method is model free, i.e., no specific model structure is assumed and no model has to be constructed during the selection procedure. The relevant inputs are found directly from the input–output data by using geometric concepts and the basic property of a function. A fast fuzzy clustering method and a similarity measure are used to assess to what degree the data samples are similar to one another. The use of fuzzy clustering is essential for this approach. A fast fuzzy clustering algorithm has been proposed which outperforms the standard FCM in terms of computational efficiency.

Rather than giving an exact selection of inputs that should be used to create model, the algorithm suggests a certain order of importance of the inputs. The final selection of inputs to be used in the model is left to the user. A full automation of the algorithm by using some threshold, for instance, would be difficult in general, as the level of importance depends on the type of nonlinearity and the excitation of the inputs (see, e.g., Example V-A). We found this technique to be less sensitive to noise and other disturbances in the data than the methods based on Lipschitz coefficients, which exploit the continuity property of functions. In addition, an arbitrary subset of regressors can be analyzed by means of this technique. This is an advantage over methods that only assess subsets of consecutive input and output lags.

Sometimes, it is not easy to decide what structure is optimal. For instance, in Example V-C, either of the models $y(k) = f(u(k-1), y(k-1))$ and $y(k) = f(u(k-1), u(k-2), y(k-1))$ can be chosen (in addition we could also consider $y(k) = f(u(k-1), u(k-2), y(k-1), y(k-2)))$. In such a case, several models can be constructed and the final model is chosen by considering the accuracy–complexity tradeoff. Note that with physical systems, there may not be any clear-cut solution, since the physical order may be infinite (distributed-parameter systems). Moreover, the NARX model structure does not require the number of input and output lags to be the same. The user may have preference for more input regressors (toward the nonlinear FIR model) or more output regressors (toward the nonlinear AR model).

Finally, note that mutual dependencies between the input candidates are often very strong. If there are many input candidates, the inconsistency levels may become very low. In such a case, more candidates can be removed from matrix $\mathbf{Z}$ at a given step and the result will generally depend on the order in which they are removed (see Section II for an example). A possible extension of the proposed algorithm could use a decision tree to optimize this choice.

ACKNOWLEDGMENT

This research was performed in part while the first author was staying at the Delft University of Technology, Delft, The Netherlands. The support of the Erasmus Program is gratefully acknowledged. The authors would also like to thank T. A. Johansen of the Norwegian Institute of Technology, Trondheim, Norway, for providing the data for the heat-transfer process example.

References

[1] X. He and H. Asada, "A new method for identifying orders of input-output models for nonlinear dynamic system," in *Proc. Amer. Control Conf.*, San Francisco, CA, 1993, pp. 2520–2523.

[2] A. Poncet and G. S. Moschytz, "Optimal order for signal and system modeling," in *Proc. IEEE Int. Symp. Circuits and System*, vol. 6, New York, 1994, pp. 221–224.

[3] L. Ljung, *System Identification: Theory for the User*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1999.

[4] J.-S. R. Jang, "Input selection for ANFIS learning," in *Proc. IEEE Int. Conf. Fuzzy System*, vol. 2, New York, 1996, pp. 1493–1499.

[5] Y. Lin, "A new approach to fuzzy-neural system modeling," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 190–198, Apr. 1993.

[6] S. L. Chiu, "Selecting input variables for fuzzy models," *J. Intell. Fuzzy Syst.*, vol. 4, pp. 243–256, 1996.

[7] D. Sáez and A. Cipriano, "Fuzzy modeling for a combined cycle power plant," in *Proc. IEEE Int. Fuzzy Syst. Conf. Proc.*, vol. 2, Seoul, Korea, 1999, pp. 1186–1190.

[8] D. A. Linkens and M.-Y. Chen, "Input selection and partition validation for fuzzy modeling using neural network," *Fuzzy Sets Syst.*, pp. 299–308, 1999.

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

[10] T. P. Hong and J. B. Chen, "Finding relevant attributes and membership functions," *Fuzzy Sets Syst.*, vol. 3, pp. 389–404, 1999.

[11] C. Rhodes and M. Morari, "Determining the model order of nonlinear input/output systems," *AIChE J.*, vol. 44, pp. 151–163, 1998.

[12] J. D. Bomberger and D. E. Seborg, "Determination of model order for NARX models directly from input–output data," *J. Process Control*, vol. 8, pp. 459–468, Oct.–Dec. 1998.

[13] R. Babuška, H. A. B. te Braake, A. J. Krijgsman, and H. B. Verbruggen, "Comparison of intelligent control schemes for real-time pressure control," *Control Eng. Practice*, vol. 4, no. 11, pp. 1585–1592, 1996.

[14] R. Babuška, *Fuzzy Modeling for Control*. Norwell, MA: Kluwer, 1998.

[15] T. A. Johansen, "Operating regime based process modeling and identification," Ph.D. dissertation, Univ. Trondheim, Norwegian Inst. Technol., Trondheim, Norway, 1994.

Radek Šindelář received the B.Sc., M.Sc., and Ph.D. degrees from the Czech Technical University (CTU), Prague, Czech Republic, in 1996, 1999, and 2004, respectively.

He is a Researcher at the Center for Applied Cybernetics, Prague, Czech Republic. His research focuses on fuzzy modeling and control of distribution networks.

Robert Babuška received the M.Sc. degree in control engineering from the Czech Technical University (CTU), Prague, Czech Republic, in 1990 and the Ph.D. degree from the Delft University of Technology, Delft, The Netherlands, in 1997.

Currently, he is a Professor with the Delft Center for Systems and Control, Faculty of Mechanical Engineering, Delft University of Technology. He has coauthored more than 50 journal papers and chapters in books, and has published a research monograph *Fuzzy Modeling for Control* (Norwell, MA: Kluwer, 1998). His research interests include the use of fuzzy set techniques and neural networks in nonlinear system identification and control.

Dr. Babuška is serving as an Associate Editor for the IEEE Transactions on Fuzzy Systems and *Engineering Applications of Artificial Intelligence*, and as an Area Editor of *Fuzzy Sets and Systems*.