Applied Soft Computing 11 (2011) 807–819

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Identification of the linear parts of nonlinear systems for fuzzy modeling

Mahmood Rezaei Sadrabadi [a,*], M. Hossein Fazel Zarandi [b]

[a] Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB, Eindhoven, The Netherlands
[b] Department of Industrial Engineering, Amirkabir University of Technology, Hafez Av., Tehran, Iran

ARTICLE INFO

Article history:
Received 3 November 2008
Received in revised form 23 April 2009
Accepted 31 December 2009
Available online 6 January 2010

Keywords:
Fuzzy clustering
Fuzzy modeling
Structure identification
Nonlinear functions

ABSTRACT

In direct approach to fuzzy modeling, structure identification is one of the most critical tasks. In modeling the nonlinear system, this fact is more crucial. In this paper, a new hybrid method is proposed to cluster the data located in the linear parts on the nonlinear systems. The proposed method can partition the input–output data in two groups: data located in the linear parts and data in the extrema. It is shown that the first group of data is suitable to be clustered by Fuzzy C-Regression Model (FCRM) clustering algorithm and the second group by Fuzzy C-Means (FCM). Then, based on the above findings, a new hybrid clustering algorithm is proposed. Finally, the proposed approach is tested and validated by several numerical examples of nonlinear functions.

1. Introduction

Fuzzy modeling is one of the most powerful techniques to estimate input–output relation in complex nonlinear systems. In general, there are two approaches to fuzzy modeling: (1) direct approach based on the expertise knowledge, and (2) indirect approach using input–output data of a system. In the first approach, the system is developed by negotiation with the experts, using some heuristics. The most common critic on this approach is that by increasing the number of input variables, the number of the rules increases exponentially, and it is not appropriate for complex systems with large number of input–output variables. Moreover, this approach is usually subjective and depends on the expert's knowledge that sometimes could be faulty [4]. Besides, knowledge acquisition is not a trivial task, experts are not always available, and when they are, their knowledge is not always consistent, systematic and complete, but often incomplete and episodic [14]. However, the main advantage of this approach is its interpretability, because when a fuzzy model is developed by using expertise knowledge, usually the model designer takes care that the model remains interpretable [9].

The second approach is extensively used by researchers and a wide variety of methods have been proposed in this domain [1,2,4,5,7,8,10,12,15–18,21–23]. Generally, fuzzy modeling methods in this approach comprise of two main phases: (1) structure identification (rough tuning), and (2) parameter identification (fine tuning). Structure identification is mostly associated with partitioning of the input space, whereas parameter identification concerns to estimating parameters of the fuzzy membership functions and the coefficients of the linear functions. In other words, the aim of structure identification is to construct an initial fuzzy model to describe the inherent structure of the given input–output data, whereas a procedure of parameter identification is applied to obtain a more precise fuzzy model regarding the identified structure [20]. Since antecedents and consequents in a fuzzy model are interdependent, parameter identification is a recursive task.

When there is a lot of input–output data of a system without any other information about it, determination of the structure of the fuzzy model becomes an important issue [20]. In such cases, fuzzy clustering is so efficient to construct the structure of the system [3]. The aim of a cluster analysis is to partition a given set of data or objects into partitions considering homogeneity within clusters and heterogeneity between clusters [6]. In the literature, different fuzzy clustering algorithms have been presented by researchers. From them, Fuzzy C-Varieties (FCV) [6], Fuzzy C-Regression Model (FCRM) [9], Fuzzy C-Means (FCM) [6,13,19,24], Gustafson-Kessel (GK) [6,19], and Fuzzy Sell Clustering (FSC) [6,24] are more prevalent.

The development of the FCM algorithm was the birth of all fuzzy clustering techniques in which spherical clouds of points are recognized [6]. The Euclidean distance is used to measure the distance between a data and cluster centers. By replacing the Euclidean distance by another metric induced by a positive definite and symmetric matrix in the FCM algorithm, ellipsoidal fuzzy clusters could also be recognized that leads to the GK algorithm [6]. Therefore, FCM and GK algorithms are suitable for detecting patches of data in hypersphere-shaped and hyperellipse-shaped clusters, respec-

* Corresponding author. Tel.: +31 40 247 5017.
E-mail addresses: m.rezaei.sadrabadi@student.tue.nl (M.R. Sadrabadi), zarandi@aut.ac.ir (M.H.F. Zarandi).

1568-4946/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2009.12.041

tively. Since GK algorithm is more adaptable to the size and the shape of the clusters, it results to more efficient clusters. FCM and GK algorithms are two of the most suitable ones which extensively have been used to identify the structure of Mamdani fuzzy models. They define fuzzy partitions based on the idea that the training data being close enough instead of having similar functions [2], which is proper to extract Mamdani fuzzy models and yet inappropriate to detect hyperplanes in TS fuzzy models.

While FCM and GK algorithms seek patches of data, the FCRM algorithm has been developed for the recognition of hyperplane-shaped clusters. It obtains the representative of each cluster by calculating a hyperplane-shaped fit of the affiliated data using Weighted Recursive Least Squared (WRLS) algorithm [9]. The idea behind the FCRM algorithm is to find a set of training data whose input–output relationship is somehow linear, and then, those training data can be clustered into one fuzzy subspace [9]. As a result, FCRM algorithm is proper to extract TS fuzzy models but inefficient to recognize the structure of Mamdani fuzzy models.

Even though TS fuzzy models are usually more precise, in contrast to Mamdani fuzzy models, but the gathered data of the system might not be appropriate for FCRM algorithm, i.e., they might form patches rather than hyperplanes. Therefore, an effective way to construct a precise fuzzy model is to detect the pattern of the data and to apply the proper clustering algorithms. Nevertheless, the pattern of the gathered data of a nonlinear system can be a mixture of the two mentioned ones: some data appropriate to regard them as the patches and some data proper to fit a hyperplane on. Generally, the data located in the linear parts of a nonlinear system is suitable for FCRM algorithm and the ones in the extrema is suitable for FCM (or GK). An effective way to enhance the results of clustering is separating data located in the linear parts and in the extrema. Then, applying FCRM clustering algorithm which complies with linear patterns on the first group and FCM (or GK) on the second group will help to increase the model's precision. In addition, such a data grouping facilitates specification of the proper number of partitions in each group. In this paper, we propose an algorithm by which the data can be classified into two patterns: the patches and the hyperplanes.

The remaining of this paper is organized as follows: in Section 2 the proposed method of identification of the linear parts of nonlinear systems is investigated. Section 3 presents the efficiency of the proposed method by some numerical examples. Finally, conclusions and future works appear in Section 4.

2. The proposed algorithm to identify the linear parts of nonlinear systems

A nonlinear function can be regarded as a set of local linear sub-functions. TS fuzzy models are constructed based upon this attitude. However, there is not a unique partitioning, but many different ones when partitioning is carried out subjectively (rather than algorithmically) by different individuals. For instance, consider Fig. 1, where a nonlinear function is partitioned to linear sub-functions subjectively by two individuals. Fig. 1(a) depicts partitioning of this nonlinear function to four linear sub-functions by the first person, whereas the second one prefers seven linear sub-functions to cover the nonlinear function, depicted in Fig. 1(b).

Though the more linear sub-functions results in a more precise estimation, one of the most desirable features of fuzzy models is their interpretability. The number of rules in a fuzzy model is a proportion of the number of partitions. In this sense, less partitions leads to more interpretable fuzzy models. By a more interpretable model, we mean a more explainable model rather than a more precise one. The proposed algorithm helps to determine the proper number of partitions.

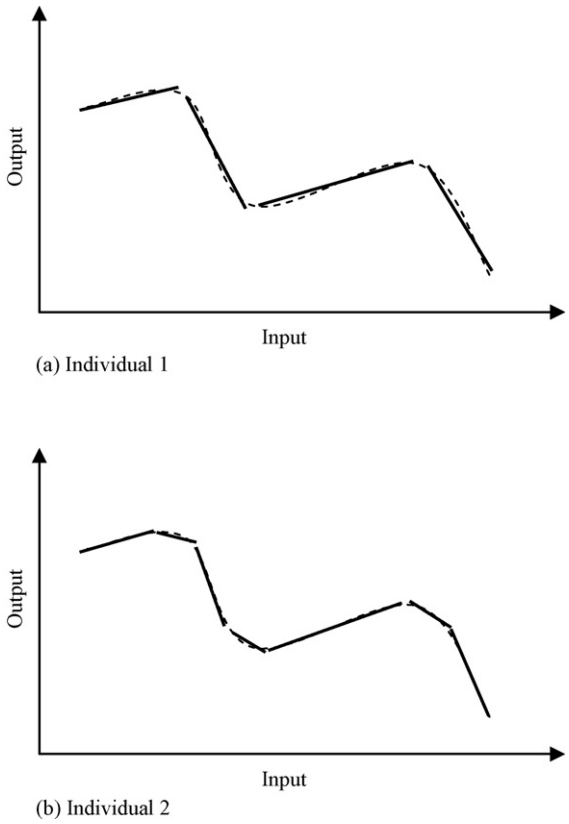

Fig. 1. Estimating a nonlinear function using linear sub-functions by two different individuals.

As mentioned in the previous section, FCRM is an appropriate algorithm to extract hyperplane-shaped clusters. However, as shown in Fig. 2, even if the hyperplanes are detected in the best manner, great errors in the extrema, in contrast to errors in the other points, are inevitable. This is usually true, regardless to the number of hyperplanes selected to estimate the main nonlinear function. Therefore, development of some method to decrease errors in the extrema is valuable.

In all nonlinear systems, sampled data can be divided into two main groups: data located in the linear parts of the system, and data located in the extrema. Applying FCRM clustering algorithm on the first group and FCM (or GK) clustering algorithm on the second group would significantly improve the results of fuzzy modeling. Fig. 3 shows the results of FCRM in partitioning of the sample data of linear parts of a nonlinear system. Similarly, Fig. 4 demonstrates the results of FCM in clustering the data in extrema of the nonlinear system.

This classification also decreases sensitivity of the specified criterion to select the best number of the clusters, because some desirable gaps are created among the data in both groups, i.e., the gaps among hyperplanes and the gaps among patches. These gaps help the FCM (or GK) algorithm to obtain clusters with more compactness inside clusters and more distinction among them. Similarly, the gaps between data selected to apply FCRM algorithm, help the algorithm to determine hyperplanes with more fitness on the data. Fig. 5 represents the final clustering result. This figure shows the identification of the linear parts of the nonlinear system. Here, the division of the sampled data into two groups not only retains the precision in the linear parts of the system, but also diminishes the errors in the extrema.

In this section, we propose a method to classify the sampled data into two main groups. The first group consists of data located in the linear parts of the nonlinear system, i.e., the data on which

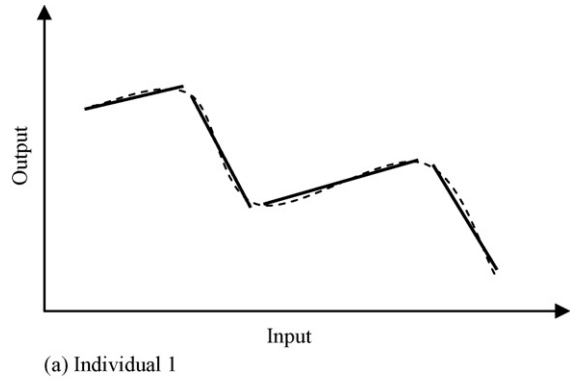

Fig. 2. Applying FCRM clustering algorithm on all data.

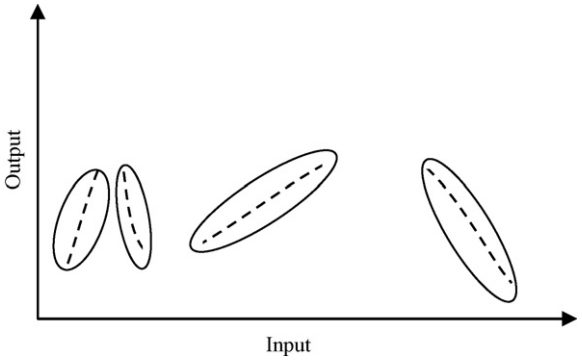

Fig. 3. Suitable data for applying FCRM clustering algorithm on.

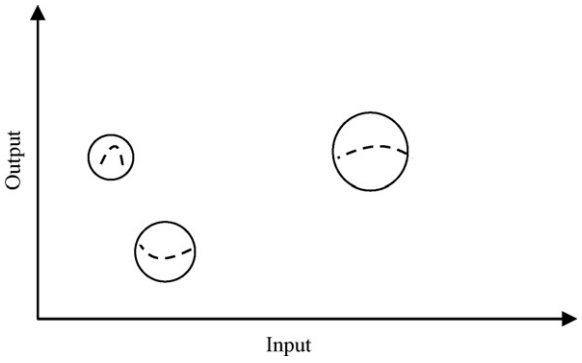

Fig. 4. Suitable data for applying FCM (or GK) clustering algorithm on.

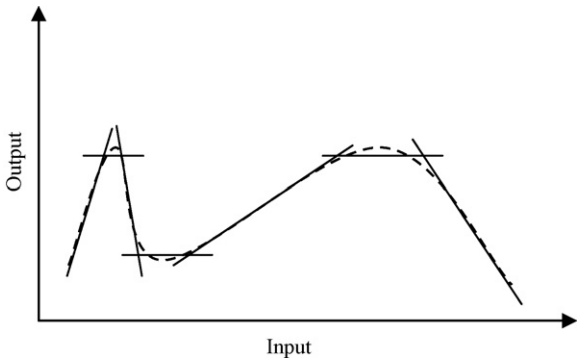

Fig. 5. The estimated function by applying FCRM and FCM clustering algorithms separately on the suitable data.

hyperplanes are fitted with high confidence level and extensive domain. In contrast, the second group consists of the data located in the extrema of the nonlinear system. FCRM clustering algorithm is appropriate for the first group, and the second group, which contains patches of data in the extrema, is suitable to be clustered by FCM (or GK). We expect that the rules generated by FCRM cover separate parts of the universe of discourse and the remaining of the universe of discourse is covered by the rules generated by FCM (or GK).

Accordingly, the consequents of the ultimate fuzzy model consists of linear functions which are the resultant of applying FCRM clustering algorithm on the first group as well as singletons which are the resultant of applying FCM (or GK) clustering algorithm on the second group. Eventually, we would have a TS fuzzy model in which consequents of some rules are linear functions and consequents of the other rules are singletons, which is a special case of linear functions. Finally, a parameter identification method can be used for fine tuning of the fuzzy model.

2.1. Theoretical aspects

This section presents some theoretical aspects related to the proposed algorithm. We first concentrate on linear functions in a 2-dimentional space and then will expand it to a $(m+1)$-dimensional space. Let $y(x) = ax + b$ be a linear function defined in the universe of discourse $U = [x', x'']$. For each input $x = x_0$, exact value of the output is calculated by $y(x_0) = ax_0 + b$. On the other hand, we are able to estimate the output of each input $x = x_0$ using the values of the outputs of other points, i.e., $\hat{y}(x_0)$ is obtainable from the values of $y(x)$; $\forall x \neq x_0$. Then, $y(x_0)$ is estimated by the weighted average of $y(x)$; $\forall x \neq x_0$, where the closer x to x_0 is assigned the bigger weight, and this weight is reduced exponentially when x distances x_0.

Although the Euclidean distance is proper to measure the distance between x and x_0, we use squared Euclidean distance for the sake of reduction of the computational effort to proof the proceeding theorems. Let $d(x)$ represents the distance between x and x_0, i.e.:

$$d(x) = (x - x_0)^2 \tag{1}$$

So, we can calculate the weight of $y(x)$ in estimating $\hat{y}(x_0)$ as follows:

$$w(x) = \frac{\exp(-\beta(x - x_0)^2)}{\int_{x'}^{x''} \exp(-\beta(x - x_0)^2)\, dx} \tag{2}$$

where β is the coefficient which indicates sensitivity towards the distance. The bigger the value of β, the more stress on the points

near to x_0. Thus, $\hat{y}(x_0)$ can be calculated as follows:

$$\hat{y}(x_0) = \frac{\int_{x'}^{x''} w(x)y(x)\ dx}{\int_{x'}^{x''} w(x)\,dx} \tag{3}$$

By considering the fact that $\int_{x'}^{x''} w(x)\ dx = 1$, we have:

$$\hat{y}(x_0) = \int_{x'}^{x''} w(x)y(x)\ dx \tag{4}$$

Theorem 2.1 (.). Let $y(x) = ax + b$ be a linear function with the universe of discourse $U = [x', x'']$. If we estimate each output by the weighted average of the other outputs according to (2) and (4), the point $x_0 = (x' + x'')/2$ would have the least squared error, that is $(y(x_0) - \hat{y}(x_0))^2 = 0$.

Proof (.). See Appendix A.

Theorem 2.2 (.). According to Theorem 2.1, the squared error of the point x with distance value Δx from the center of the line, $x_0 = (x' + x'')/2$, is calculated as:

$$(y(x_0 + \Delta x) - \hat{y}(x_0 + \Delta x))^2 = \frac{a^2L^2\beta}{\pi e^{\beta L^2/2}(2\Phi(L\sqrt{\beta/2}) - 1)^2}(\Delta x)^2 \tag{5}$$

where $L = x'' - x'$ and $\Phi(x) = \int_{-\infty}^{x} (1/\sqrt{2\pi})e^{-z^2}\,dz$.

Proof (.). See Appendix B.

As can be observed from Theorem 2.2, the squared error of the output of $x = x_0 + \Delta x$ is proportional to the squared Euclidean distance of x from $x_0 = (x' + x'')/2$. This implies that the maximum error occurs in the boundary points $x = x'$ and $x = x''$. Moreover, by increasing the interval L, the squared error for all points approaches to zero, because the numerator grows linearly by L^2 but the denominator grows exponentially by L^2. We can control the intensity of the growth of squared error by adjusting β. By increasing the value of β, the squared error is reduced because while the numerator grows linearly by β, the denominator grows exponentially by it. This complies with the initial role of β, inasmuch as by increasing the value of β, the nearer points to x are assigned bigger weights. In addition, if we use the point in a smaller interval to estimate the output, the related squared error would be smaller, regardless to its location. Since differentiation of a linear function is a fix value, Theorem 2.2 shows the exact value of the squared error for all points. Now, Theorems 2.1 and 2.2 are generalized to a $(m+1)$-dimensional space.

Theorem 2.3 (.). Let $y(X) = a_0 + \sum_{j=1}^{m} a_j x_j$ be a hyperplane in a $(m+1)$-dimensional space and $U_j = [x_j', x_j'']$ be the universe of discourse in the jth dimension. Similar to the previous discussion, the output of the point $X_0 = (x_{01}, x_{02}, \ldots, x_{0m})$ is estimated using outputs of the other points $X = (x_1, x_2, \ldots, x_m)$, where the weight of each output has a negative exponential relation corresponding to the squared Euclidean distance of X from X_0. In other words, the distance can be defined as:

$$d(X) = ||X - X_0||^2 = \sum_{j=1}^{m} (x_j - x_{0j})^2 \tag{6}$$

and the estimated output for the point X_0 is calculated as:

$$\hat{y}(X_0) = \int_{x_m'}^{x_m''} \cdots \int_{x_2'}^{x_2''} \int_{x_1'}^{x_1''} (w(X)y(X))\,dx_1\,dx_2 \ldots dx_m \tag{7}$$

where

$$w(X) = \frac{\exp\left(-\sum_{j=1}^{m} \beta_j (x_j - x_{0j})^2\right)}{\int_{x_m'}^{x_m''} \cdots \int_{x_2'}^{x_2''} \int_{x_1'}^{x_1''} \exp\left(-\sum_{j=1}^{m} \beta_j (x_j - x_{0j})^2\right)\ dx_1\,dx_2 \ldots dx_m} \tag{8}$$

Proof (.). See Appendix C.

By using this method, $X_0 = (x_{01}, x_{02}, \ldots, x_{0m})$ has the least squared error $(y(X_0) - \hat{y}(X_0))^2 = 0$ if $x_{0j} = (x_j' + x_j'')/2$; $\forall j = 1, 2, \ldots, m$.

Theorem 2.4 (.). If the outputs of a hyperplane are estimated according to Theorem 2.3, the squared error of the point X which has distance $\Delta X = (\Delta x_1, \Delta x_2, \ldots, \Delta x_m)$ from the center of the hyperplane, X_0, is:

$$(y(X_0 + \Delta X) - \hat{y}(X_0 + \Delta X))^2 = \left(\sum_{j=1}^{m} \Omega_j\right)^2 \tag{9}$$

where $L_j = x_j'' - x_j'$; $j = 1, 2, \ldots, m$ and

$$\Omega_j = \frac{a_j L_j \sqrt{\beta_j}}{\sqrt{\pi} e^{\beta_j L_j^2/4}(2\Phi(L_j\sqrt{\beta_j/2}) - 1)}\ \Delta x_j \tag{10}$$

Proof (.). See Appendix D.

As a matter of fact, (10) shows that the sign of Ω_j depends on the signs of a_j and Δx_j which both can be negative or positive. Consequently, squared error for each point consists of the propositions that can be negative or positive, and their total sum does not have direct proportion to the amounts of Δx_j's. Unfortunately, we cannot conclude that all boundary points in the hyperplane have the biggest squared error, because by increasing each $|\Delta x_j|$ the related total squared error does not necessarily increase. However, it is proved that all points which have the minimum amounts of squared error are concentrated around the center of the hyperplane. Theorem 2.5 states this pivotal matter.

Theorem 2.5 (.). If we estimate the outputs of a hyperplane according to Theorem 2.3, then all points which have the squared errors less than $(\Delta y)^2$ are located in a hyperellipse which its center is the centroid of the hyperplane and thus, they are concentrated around the center of the hyperplane.

Proof (.). See Appendix E.

According to Theorem 2.5, we expect that by distancing the center of the hyperplane, the squared error of the points increase. It should be noted that in the dimensions with bigger partial differentiation, i.e., bigger amount of $|a_j|$, the squared error increases more rapidly and in the dimensions with smaller partial differentiation, it increases slower.

2.2. Some additional remarks

Now, let us consider some necessary assumptions to the above mentioned theorems in order to attain a real nonlinear system. Consider a system in which the relation between inputs and output is expressed by several hyperplanes, where each hyperplane covers a part of the universe of discourse. We first deal with such a system in a 2-dimensional space. Let L_i be the length of the part of universe of discourse covered by the ith line. If we estimate the output of each point on the X axis using the outputs of the points of the same line, then the center of each interval L_i would have the least squared error equal to zero, and the extrema would have the

biggest squared errors according to Theorem 2.2, i.e.:

$$(y(x_0+\Delta x)-\hat{y}(x_0+\Delta x))^2=\frac{a_i^2L_i^2\beta}{\pi e^{\beta L_i^2/2}(2\Phi(L_i\sqrt{\beta/2})-1)^2}\left(\frac{1}{2}L_i\right)^2 \tag{11}$$

where i refers to the ith line.

Note that even if we do not adjust the weight of each output corresponding to the distance between x and x_0, we would again have a squared error with the amount of zero in the center of each interval, even though the other points would have bigger squared errors.

In a $(m+1)$-dimensional space, let L_{ij} be the length of the part of the universe of discourse covered by the ith hyperplane in the jth dimension. Similar to the 2-dimensional space, if we estimate the output of each point in a $(m+1)$-dimensional space using the outputs of the points of the same hyperplane, then the center of each hyperplane would have the least squared error with the amount of zero; i.e., the point $X_i=(x_{i1}, x_{i2}, \ldots, x_{im})$ where x_{ij} is in the center of L_{ij}. By moving from the center of each hyperplane, $X_i=(x_{i1}, x_{i2}, \ldots, x_{im})$, towards its boundaries into two opposite directions along the jth axis, the squared errors will increase. In this case, in the two boundary points, there is $\Delta x_j=\pm(1/2)L_{ij}$ for a particular j and $\Delta x_j=0$ for the other j's. Hence, the squared error is:

$$(y(X_0+\Delta X)-\hat{y}(X_0+\Delta X))^2=\left(\frac{a_{ij}L_{ij}\sqrt{\beta_j}}{\sqrt{\pi}e^{\beta_jL_{ij}^2/4}(2\Phi(L_{ij}\sqrt{\beta_j/2})-1)}\left(\pm\frac{1}{2}L_{ij}\right)\right)^2 \tag{12}$$

Like the 2-dimensional space, if we do not adjust the weight of each output corresponding to the distance between X and X_0, again the least squared error with the amount of zero in the center of each hyperplane is resulted. The other points, however, are expected to have bigger squared errors.

The above discussion shows that if the squared errors with the amount of zero are desired in the center of each hyperplane, each output must be estimated by the outputs of the points located in a narrower distance from it rather than all point in the universe of discourse. This results in squared errors with the amount of zero not only in the center of each hyperplane, but also in the points near to the centers. In 2-dimensional space, we define L_0 as:

$$L_0=\min\{L_i;\ \ i=1,2,\ldots,c\} \tag{13}$$

When just the outputs on a particular line are used to estimate an output on the same line, the points in the distance at most $L_i/2$ from the center of the interval L_i are used, and this leads to the squared error with the amount of zero just for the center of the line and positive squared errors for the other points. Now, if the points located in the distance less than $L_0/2$ from the center of each line are used to estimate the outputs, then all points in the distance less than $(L_i/2-L_0/2)$ from the center of the ith line would have squared errors with the amount of zero.

Similarly, in a $(m+1)$-dimensional space, L_{0j} is defined as:

$$L_{0j}=\min\{L_{ij};\ \ i=1,2,\ldots,c\};\ \ j=1,2,\ldots,m \tag{14}$$

If the points in the distance $L_{0j}/2$; $\forall j=1,2,\ldots,m$ from the center of each hyperplane are used, then all points located in the distance less than $(L_{ij}/2-L_{0j}/2)$; $\forall j=1,2,\ldots,m$ from the center of each hyperplane would have squared error with the amount of zero. In this paper, we use $L_{0j}=0.1L_j$, where L_j is the total length of the jth axis and 0.1 is an empirical coefficient specified by solving different numerical examples.

According to the discussions explained so far, we are able to detect the centers of hyperplanes and the points around them using the squared error as a criterion. There are, however, some crucial points that by considering them we must not expect to attain squared errors with the amount of zero, even in the most linear parts of the system. These are as follows:

(a) *There is a set of data rather than a set of functions*. Obviously, we access just to a finite number of input–output data of the system not to equations of hyperplanes. In other words, when we are faced to a fuzzy modeling problem, a set of input–output data of the system, rather than a set of linear functions are in hand. This implies that we do not know the amounts of L_j's. In order to solve this problem, the amount of L_j can be estimated as:

$$L_j=x_j^{\max}-x_j^{\min} \tag{15}$$

where $x_j^{\max}=\max\{x_{kj};\ \ k=1,2,\ldots,n\}$ and $x_j^{\min}=\min\{x_{kj};\ \ k=1,2,\ldots,n\}$.

Moreover, (2), (4), (8) and (7) can be transformed to (16)–(19) as follows:

$$w(x_k)=\frac{\exp(-\beta(x_k-x_0)^2)}{\sum_{k=1}^{n}\exp(-\beta(x_k-x_0)^2)} \tag{16}$$

$$\hat{y}(x_0)=\sum_{k=1}^{n}w(x_k)y(x_k) \tag{17}$$

$$w(X_k)=\frac{\exp\left(-\sum_{j=1}^{m}\beta_j(x_{kj}-x_{0j})^2\right)}{\sum_{k=1}^{n}\exp\left(-\sum_{j=1}^{m}\beta_j(x_{kj}-x_{0j})^2\right)} \tag{18}$$

$$\hat{y}(X_0)=\sum_{k=1}^{n}(w(X_k)y(X_k)) \tag{19}$$

(b) *Data are random*. If all the gathered data of the system have a deterministic uniform distribution, it can still be expected that the center of each hypothetical hyperplane and the points around them have squared errors with the amount of zero. In the real situations, even though the input data have usually a uniform distribution but this distribution is random. It implies that we must not expect squared errors exactly equal to zero for the centers and the points around them. Still, the error of these points would have a normal distribution with the mean zero, and thus their squared error would be a positive random variable with the mean near to zero. To conclude, in order to discover the centers of hyperplanes and the points around them, we must transform the condition *squared errors with the amount of zero* to the condition *squared errors near to zero*.

(c) *The system is inherently nonlinear*. Generally, the relation between inputs and output of the nonlinear systems is not a set of local hyperplanes but is a $(m+1)$-dimensional hypersurface on which we have supposed local hyperplanes so far. This implies that the squared errors again increase. In order to control this fact, the assigned weight to the points is decreased exponentially corresponding to their distances from the estimated point. This leads to the fact that each output is estimated relying more on the points around it. Therefore, in nonlinear systems, we try to estimate each output using the points around it. Hence, the previous theorems would remain quite valid. This, furthermore, causes that the squared error of all points, consisting the points around the centers, do not increase much and so we seek still the points with squared errors near to zero.

There are two main conflicting issues which handling them leads to a more efficient algorithm. In one hand, considering more limited points to estimate the output of each point leads to smaller squared errors that ultimately makes difficult detecting the linear parts of the systems. So, we must use the points

in a wider interval to estimate each output. This causes increasing the difference between squared error in the center of each hyperplane and squared errors of the other points and thus the centers and the points around them are recognized more easily. On the other hand, using many points to estimate the output of each point results in bigger squared errors and again makes difficult to detect the linear parts of the system, because the difference between squared error in the center of each hyperplane and squared errors of the other points increases. Investigating different nonlinear functions shows that using the points which are in the distance less than 10% of the universe of discourse along each dimension is proper, i.e., we should estimate the output of each point $X_0 = (x_{01}, x_{02}, \ldots, x_{0m})$ using the outputs of the points $X_k = (x_{k1}, x_{k2}, \ldots, x_{km})$ which satisfy (20):

$$|x_{0j} - x_{kj}| \le 0.1L_j; \quad \forall j = 1, 2, \ldots, m \tag{20}$$

2.3. The proposed algorithm

In the proposed algorithm, the data are classified into two separate groups; the first group consists of the data located in the linear parts of the nonlinear system, and the second group consists of the data located in the extrema of the nonlinear system. The proposed algorithm is presented in Fig. 6.

3. Numerical examples

In this section, the proposed method to identify the linear parts of nonlinear systems is validated using some numerical examples. In the first three examples, three nonlinear functions are considered. From the real function, some uniformly distributed data are generated and then the proposed algorithm is implemented on these data. In the last example, some data of an unknown nonlinear function are investigated, and efficiency of the algorithm to identify linear parts is demonstrated using these data.

Step 1) calculate the output of each point $X_0 = (x_{01}, x_{02}, \ldots, x_{0m})$ as:

$$\hat{y}(X_0) = \sum_{X_k \in S}^{n_0} w_k . y(X_k) \tag{21}$$

where,

$$S = \left\{ X_k \,\middle|\, |x_{0j} - x_{kj}| \le 0.1L_j \quad ; \forall j = 1,2,\ldots,m \right\} \tag{22}$$

$$w_k = \frac{\exp(-(\frac{x_{kj} - x_{0j}}{L_j})^2)}{\sum_{k=1}^{n_0} \exp(-(\frac{x_{kj} - x_{0j}}{L_j})^2)} \tag{23}$$

$$L_j = \max\{x_{kj}; k = 1,2,\ldots,n\} - \min\{x_{kj}; k = 1,2,\ldots,n\} \tag{24}$$

and n_0 is the number of elements in the set S.

Step 2) calculate the squared error for each point as:

$$(\Delta y_k)^2 = (y(X_k) - \hat{y}(X_k))^2 \quad ; k = 1,2,\ldots,n \tag{25}$$

Step 3) calculate the mean and the variance of the squared errors and form the sets S_0 and M_0 according to (26)-(29):

$$\mu_0 = E((\Delta y)^2) = \frac{\sum_{k=1}^{n} (\Delta y_k)^2}{n} \tag{26}$$

$$\sigma_0^{\,2} = Var((\Delta y)^2) = \frac{\sum_{k=1}^{n} (\Delta y_k - \mu_0)^2}{n} \tag{27}$$

$$S_0 = \left\{ X_k \,\middle|\, (\Delta y_k)^2 \le \mu_0 + 3\sigma_0 \right\} \tag{28}$$

$$M_0 = \left\{ X_k \,\middle|\, (\Delta y_k)^2 > \mu_0 + 3\sigma_0 \right\} \tag{29}$$

Step 4) calculate the new mean for the elements of S_0 and form the sets S_1 and M_1 using (30)-(32):

$$\mu_1 = \frac{\sum_{X_k \in S_0}^{n} (\Delta y_k)^2}{n_{S_0}} \tag{30}$$

$$S_1 = \{X_k \mid (\Delta y_k)^2 \le \mu_1\} \tag{31}$$

$$M_1 = \{X_k \mid (\Delta y_k)^2 > \mu_1\} \tag{32}$$

Step 5) apply FCRM algorithm on the data in S_1 and FCM (or GK) on the data in M_1.

Fig. 6. The proposed algorithm.

Table 1
Some numerical results of example 1.

# of data	$\mu_0+3\sigma_0$	μ_1	$\lvert S_1\rvert$	$\lvert M_1\rvert$
201	0.0195	0.0022	128	73

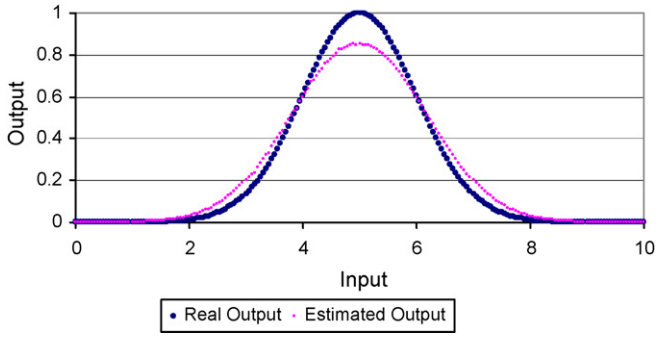

Fig. 7. The real and the estimated function of example 1.

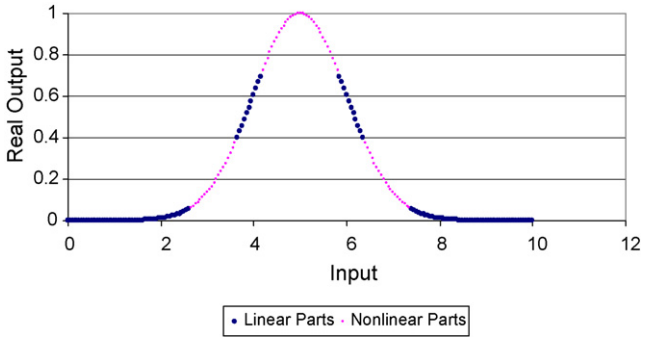

Fig. 9. The categorized data of example 1.

Table 2
Some numerical results of example 2.

# of data	$\mu_0+3\sigma_0$	μ_1	$\lvert S_1\rvert$	$\lvert M_1\rvert$
100	0.5332	0.1722	51	49

3.1. Example 1

For the first example, let us consider a *gaussian* function of the form:

$$y = \exp\left(-\frac{(x-5)^2}{2}\right) \tag{33}$$

We consider interval [0, 10] as the universe of discourse. So, $L=10$ and the data which are in the distance with the amount of at most $0.1L=0.1(10)=1$ from the estimated point must be taken into account. Based on 201 uniformly distributed data, the outputs are estimated according to the proposed algorithm. Table 1 shows a summary of some numerical results of implementation of the algorithm on the mentioned data. In this table, $|S_1|$ and $|M_1|$ indicate the cardinality of the sets S_1 and M_1, respectively. In other words, $|S_1|$ indicates the number of data which the algorithm categorizes as the data located in the linear parts. Equivalently, $|M_1|$ shows the number of data which the algorithm categorizes as the data located in the extrema.

The following figures show presentations of implementation of the algorithm. Fig. 7 shows the considered points of the real function as well as the estimated points. One can immediately understand that in the linear parts the estimation error is lower than in the nonlinear parts. This is, indeed, the main idea to develop the proposed algorithm.

In order to obtain the final category of the linear and nonlinear points, we use the squared errors shown in Fig. 8. All points with squared error less than $\mu_1=0.0022$ are assigned to set S_1 and the others are categorized in set M_1.

Fig. 9 once again shows the initial data but within two categories. Now, we have obtained a figure like Figs. 3 and 4, but using a mathematical algorithm rather than a subjective method. FCRM can perform much better on the data located in the linear parts, and the remaining ones can be clustered using FCM (or GK) more efficiently.

Note that these categories are very close to the way that we subjectively categorize linear and nonlinear parts of a *gaussian* function.

3.2. Example 2

In this example, a *Sin* function is used, i.e.:

$$y = \mathrm{Sin}(x) \tag{34}$$

We consider interval [0, 20] as the universe of discourse for this function. So, we have $L=20$ and the data in the vicinity of at most $0.1L=0.1(20)=2$ units from the estimated point must be considered. Based on 100 uniformly generated data, the outputs are estimated according to the proposed algorithm. Table 2 shows a summary of some numerical results of implementation of the algorithm on the mentioned data.

Interpretations of Figs. 10–12 are equivalent to those of Figs. 7–9. So, to avoid redundancy, we do not repeat those explanations.

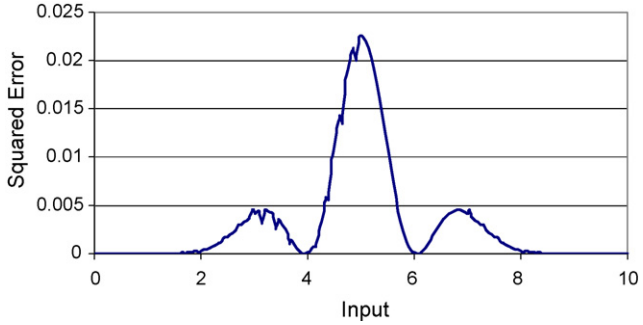

Fig. 8. The squared errors of data of example 1.

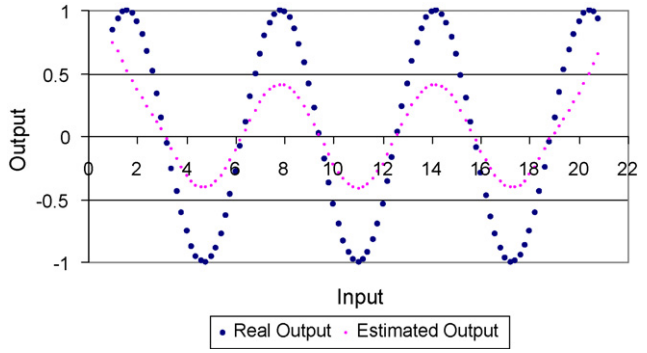

Fig. 10. The real and the estimated function of example 2.

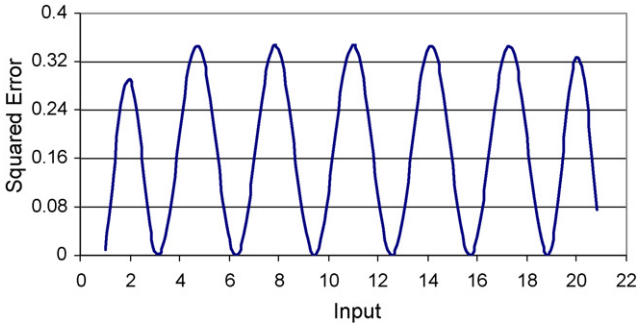

Fig. 11. The squared errors of data of example 2.

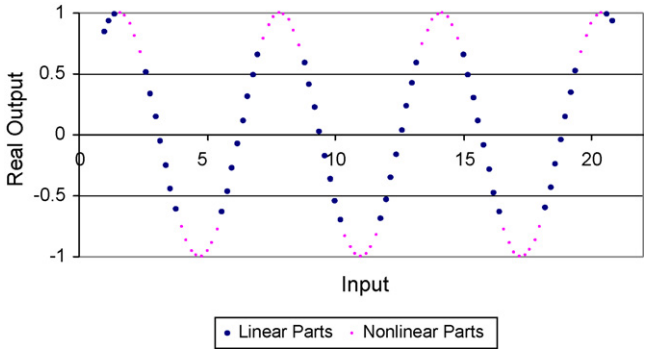

Fig. 12. The categorized data of example 2.

Table 3
Some numerical results of example 3.

# of data	$\mu_0+3\sigma_0$	μ_1	$\lvert S_1\rvert$	$\lvert M_1\rvert$
250	0.0107	0.0017	184	66

3.3. Example 3

Here, we investigate a saturated function, *log sigmoid*. This function is presented as:

$$y = \frac{1}{1+\exp(-x)} \tag{35}$$

Interval [−15, 15] is considered as its universe of discourse. So, we have $L=30$ and the data with the distance at most $0.1L=0.1(30)=3$ from the estimated point must be considered. Based on 250 uniform data in interval [−15, 15], the algorithm is implemented. Table 3 shows a summary of some numerical results, and Figs. 13–15 show the graphical presentations.

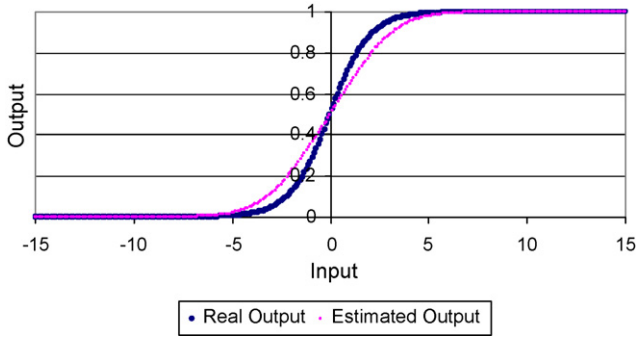

Fig. 13. The real and the estimated function of example 3.

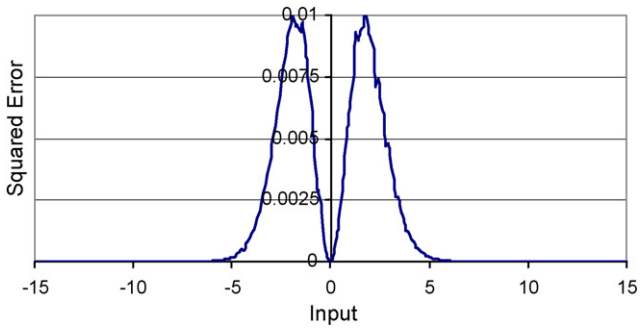

Fig. 14. The squared errors of data of example 3.

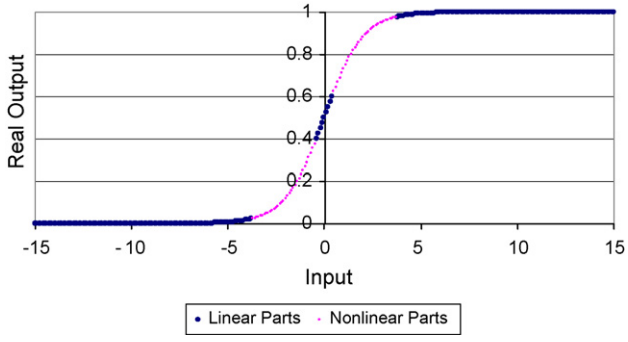

Fig. 15. The categorized data of example 3.

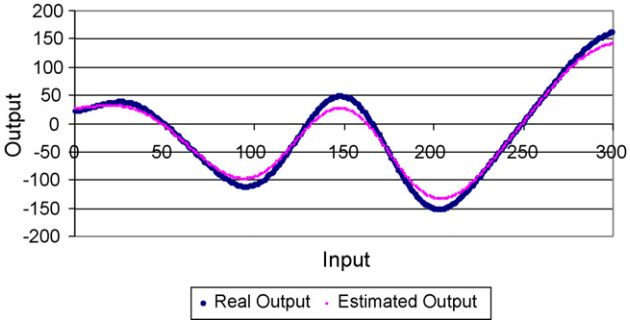

Fig. 16. The real and the estimated function of example 4.

3.4. Example 4

In the last example, we do not use any function. Instead, we use some data of a complicated unknown nonlinear function presented in Fig. 16.

We consider interval [0, 300] as the universe of discourse. So, $L=300$ and the data in the vicinity of at most $0.1L=0.1(300)=30$ units from the estimated point must be taken into account. Based on 300 uniformly distributed data, the outputs are estimated according to the proposed algorithm. Table 4 shows a summary of some numerical results of implementation of the algorithm on the mentioned data, and Figs. 16–18 show the visual presentations of the results.

Solving several numerical examples revealed the fact that the algorithm performs better for nonlinear functions the linear parts

Table 4
Some numerical results of example 4.

# of data	$\mu_0+3\sigma_0$	μ_1	$\lvert S_1\rvert$	$\lvert M_1\rvert$
300	488.94	108.58	193	107

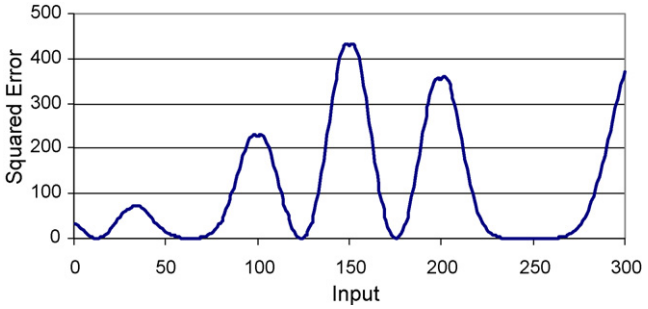

Fig. 17. The squared errors of data of example 4.

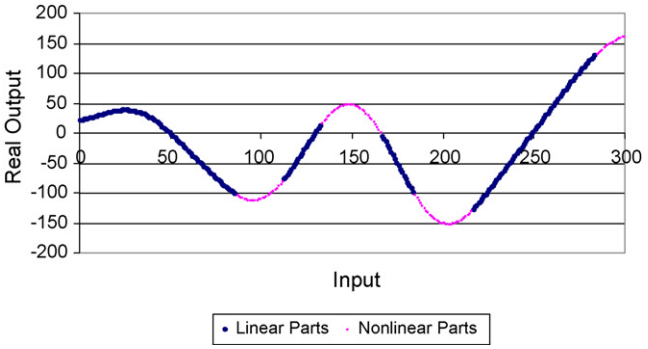

Fig. 18. The categorized data of example 4.

of which cover a higher ratio of the universe of discourse in comparison to the nonlinear parts.

4. Conclusion and future works

A novel hybrid method based on some mathematical theorems to identify the linear parts of nonlinear systems has been proposed in this paper. The proposed method divides the sample data into two groups: data located in the linear parts and data located in the extrema. Such a data grouping causes some desirable gaps among data in each group that contributes FCRM and FCM (or GK) algorithms to specify the correct number of clusters in each group. The proposed algorithm can be used to extract not only a more precise TS fuzzy model, but also to extract a more precise Mamdani fuzzy model by identifying the extrema, since an optimal Mamdani fuzzy model is obtained when rule patches cover the extrema of the approximated function [11]. Efficiency of the proposed method has been demonstrated by variant nonlinear data. The algorithm uses the points which are in the distance less than 10% of the universe of discourse along each dimension. This value has been adopted empirically by solving several numerical examples of nonlinear functions and plays a crucial role. However, more empirical and mathematical research is needed to adjust it more precisely in order to fortify the results of the proposed algorithm. Likewise, the bounds $\mu_0 + 3\sigma_0$ and μ_1 to assign the points to sets S_0 and S_1 have been selected empirically that needs more research.

Appendix A. Proof of Theorem 2.1

Considering the function $y(x) = ax + b$, the real output for $x = x_0$ is $y(x_0) = ax_0 + b$. According to the proposed algorithm, we estimate each output as:

$$\hat{y}(x_0) = \int_{x'}^{x''} w(x)y(x)\ dx$$

where $w(x) = e^{-\beta(x-x_0)^2} / \int_{x'}^{x''} e^{-\beta(x-x_0)^2}\ dx$ and β is the coefficient which indicates sensitivity towards the distance. So, we can calculate $\hat{y}(x_0)$ as follows:

$$\hat{y}(x_0) = \int_{x'}^{x''} w(x)y(x)\ dx$$

$$\hat{y}(x_0) = \frac{1}{\int_{x'}^{x''} e^{-\beta(x-x_0)^2}\ dx} \int_{x'}^{x''} e^{-\beta(x-x_0)^2}(ax+b)\ dx$$

$$\hat{y}(x_0) = \frac{1}{\int_{x'}^{x''} e^{-\beta(x-x_0)^2}\ dx} \left(a\int_{x'}^{x''} xe^{-\beta(x-x_0)^2}\ dx + b\int_{x'}^{x''} e^{-\beta(x-x_0)^2}\ dx \right)$$

where $\int_{x'}^{x''} e^{-\beta(x-x_0)^2}\ dx = (1/\sqrt{2\beta})\sqrt{2\pi} \int_{x'}^{x''} (1/(1/\sqrt{2\beta})\sqrt{2\pi})$ $e^{-(1/2)((x-x_0)/(1/\sqrt{2\beta}))^2}\ dx$.

The last integral indicates the area below diagram of a normal distribution with mean x_0 and variance $1/(2\beta)$. Therefore,

$$\int_{x'}^{x''} e^{-\beta(x-x_0)^2}\ dx = \frac{1}{\sqrt{2\beta}}\sqrt{2\pi}(\Pr(X \le x'') - \Pr(X \le x'))$$

where $X \sim N(x_0, 1/(2\beta))$.

Transforming X to a standard normal distribution results in:

$$\int_{x'}^{x''} e^{-\beta(x-x_0)^2}\ dx = \sqrt{\frac{\pi}{\beta}}(\Phi(\sqrt{2\beta}(x'' - x_0)) - \Phi(\sqrt{2\beta}(x' - x_0)))$$

where $Z \sim N(0, 1)$ and $\Phi(z) = \Pr(Z \le z)$.

Let $k = \sqrt{\pi/\beta}(\Phi(\sqrt{2\beta}(x'' - x_0)) - \Phi(\sqrt{2\beta}(x' - x_0)))$, so:

$$\hat{y}(x_0) = \frac{1}{k}\left(a\int_{x'}^{x''} xe^{-\beta(x-x_0)^2}\ dx + bk \right)$$

$$\hat{y}(x_0) = (ax_0 + b) - \frac{a}{2\beta k}(e^{-\beta(x''-x_0)^2} - e^{-\beta(x'-x_0)^2})$$

Now, let $L = x'' - x'$, so we have $x_0 = (x' + x'')/2$ when $x' - x_0 \to L/2$. Therefore,

$$\lim_{x_0 - x' \to L/2} k = \lim_{x_0 - x' \to L/2} \sqrt{\frac{\pi}{\beta}}(\Phi(\sqrt{2\beta}(x'' - x_0)) - \Phi(\sqrt{2\beta}(x' - x_0)))$$

$$\lim_{x_0 - x' \to L/2} k = \sqrt{\frac{\pi}{\beta}}\left(2\Phi\left(L\sqrt{\frac{\beta}{2}} \right) - 1 \right)$$

Accordingly,

$$\lim_{x_0 - x' \to L/2} \hat{y}(x_0) = (ax_0 + b) - \frac{a}{2\beta}\frac{\sqrt{\beta}}{\sqrt{\pi}(2\Phi(L\sqrt{\beta/2}) - 1)}(e^{-\beta(L/2)^2} - e^{-\beta(-L/2)^2})$$

$$\lim_{x_0 - x' \to L/2} \hat{y}(x_0) = (ax_0 + b) = y(x_0)$$

Thus, Theorem 2.1 is proved. □

Appendix B. Proof of Theorem 2.2

In one hand, we have:

$$y(x_0 + \Delta x) = y(x_0) + \frac{dy(x_0)}{dx_0}\Delta x = (ax_0 + b) + a\ \Delta x = a(x + \Delta x) + b$$

On the other hand, according to Theorem 2.1 we have:

$$\hat{y}(x_0) = (ax_0 + b) - \frac{a}{2\beta k}(e^{-\beta(x''-x_0)^2} - e^{-\beta(x'-x_0)^2})$$

Partially differentiating the above equation towards x_0 leads to:

$$\frac{d\hat{y}(x_0)}{dx_0} = a - \frac{a}{2\beta k}(2\beta(x''-x_0)e^{-\beta(x''-x_0)^2} - 2\beta(x'-x_0)e^{-\beta(x'-x_0)^2})$$

$$\lim_{x_0-x'\to L/2}\frac{d\hat{y}(x_0)}{dx_0} = a - \frac{a}{2\beta}\frac{\sqrt{\beta}}{\sqrt{\pi}(2\Phi(L\sqrt{\beta/2})-1)} \times \left(2\beta\frac{L}{2}e^{-\beta(L/2)^2} - 2\beta\left(-\frac{L}{2}\right)e^{-\beta(-L/2)^2}\right)$$

$$\lim_{x_0-x'\to L/2}\frac{d\hat{y}(x_0)}{dx_0} = a - \frac{aL\sqrt{\beta}}{\sqrt{\pi}e^{\beta L^2/4}(2\Phi(L\sqrt{\beta/2})-1)}$$

Therefore,

$$\hat{y}(x_0+\Delta x) = \hat{y}(x_0) + \frac{d\hat{y}(x_0)}{dx_0}\ \Delta x$$

$$\hat{y}(x_0+\Delta x) = (ax_0+b) + a\ \Delta x - \frac{aL\sqrt{\beta}}{\sqrt{\pi}e^{\beta L^2/4}(2\Phi(L\sqrt{\beta/2})-1)}\ \Delta x$$

$$\hat{y}(x_0+\Delta x) = y(x_0+\Delta x) - \frac{aL\sqrt{\beta}}{\sqrt{\pi}e^{\beta L^2/4}(2\Phi(L\sqrt{\beta/2})-1)}\ \Delta x$$

Accordingly,

$$(y(x_0+\Delta x) - \hat{y}(x_0+\Delta x))^2 = \frac{a^2L^2\beta}{\pi e^{\beta L^2/2}(2\Phi(L\sqrt{\beta/2})-1)^2}(\Delta x)^2$$

So, Theorem 2.2 is proved. □

Appendix C. Proof of Theorem 2.3

In a $(m+1)$-dimensional space, real output of the point $X_0 = (x_{01}, x_{02}, \ldots, x_{0m})$ is:

$$y(X_0) = a_0 + \sum_{j=1}^{m} a_j x_{0j}$$

and squared Euclidean distance of $X_0 = (x_{01}, x_{02}, \ldots, x_{0m})$ from $X = (x_1, x_2, \ldots, x_m)$ is:

$$d(X) = ||X - X_0||^2 = \sum_{j=1}^{m}(x_j - x_{0j})^2$$

Also, the weight of X in estimating X_0 is:

$$w(X) = \frac{e^{-\sum_{j=1}^{m}\beta_j(x_j-x_{0j})^2}}{\int_{x'_m}^{x''_m}\cdots\int_{x'_2}^{x''_2}\int_{x'_1}^{x''_1} e^{-\sum_{j=1}^{m}\beta_j(x_j-x_{0j})^2}\ dx_1\,dx_2\ldots dx_m}$$

where β_j is the coefficient which represents sensitivity towards the distance in the jth dimension. Therefore, $y(X_0)$ is calculated as follows:

$$\hat{y}(X_0) = \frac{\int_{x'_m}^{x''_m}\cdots\int_{x'_2}^{x''_2}\int_{x'_1}^{x''_1}(w(X)y(X))\,dx_1\,dx_2\ldots dx_m}{\int_{x'_m}^{x''_m}\cdots\int_{x'_2}^{x''_2}\int_{x'_1}^{x''_1} w(X)\ dx_1\,dx_2\ldots dx_m}$$

Since $\int_{x'_m}^{x''_m}\cdots\int_{x'_2}^{x''_2}\int_{x'_1}^{x''_1} w(X)\ dx_1\,dx_2\ldots dx_m = 1$, we have:

$$\hat{y}(X_0) = \int_{x'_m}^{x''_m}\cdots\int_{x'_2}^{x''_2}\int_{x'_1}^{x''_1}(w(X)y(X))\,dx_1\,dx_2\ldots dx_m$$

$$\hat{y}(X_0) = \frac{1}{\int_{x'_m}^{x''_m}\cdots\int_{x'_2}^{x''_2}\int_{x'_1}^{x''_1} e^{-\sum_{j=1}^{m}\beta_j(x_j-x_{0j})^2}\ dx_1\,dx_2\ldots dx_m} \times \int_{x'_m}^{x''_m}\cdots\int_{x'_2}^{x''_2}\int_{x'_1}^{x''_1} e^{-\sum_{j=1}^{m}\beta_j(x_j-x_{0j})^2}\left(a_0+\sum_{j=1}^{m}a_jx_j\right) dx_1\,dx_2\ldots dx_m$$

We can write:

$$\int_{x'_m}^{x''_m}\cdots\int_{x'_2}^{x''_2}\int_{x'_1}^{x''_1} e^{-\sum_{j=1}^{m}\beta_j(x_j-x_{0j})^2}\ dx_1\,dx_2\ldots dx_m = \prod_{j=1}^{m}\int_{x'_j}^{x''_j} e^{-\beta_j(x_j-x_{0j})^2}\ dx_j = \prod_{j=1}^{m} q_j$$

where

$$q_j = \int_{x'_j}^{x''_j} e^{-\beta_j(x_j-x_{0j})^2}\ dx_j$$

$$q_j = \sqrt{\frac{\pi}{\beta_j}}(\Phi(\sqrt{2\beta_j}(x''_j - x_{0j})) - \Phi(\sqrt{2\beta_j}(x'_j - x_{0j})))$$

$Z \sim N(0,1)$ and $\Phi(z) = \Pr(Z \le z)$.

Therefore,

$$\hat{y}(X_0) = \int_{x'_m}^{x''_m}\cdots\int_{x'_2}^{x''_2}\int_{x'_1}^{x''_1}(w(X)y(X))\,dx_1\,dx_2\ldots dx_m$$

where $w(X) = e^{-\sum_{j=1}^{m}\beta_j(x_j-x_{0j})^2}/\prod_{j=1}^{m}q_j$.

By replacing $w(X)$ in $\hat{y}(X_0)$ we would have:

$$\hat{y}(X_0) = \frac{1}{\prod_{j=1}^{m}q_j}\int_{x'_m}^{x''_m}\cdots\int_{x'_2}^{x''_2}\int_{x'_1}^{x''_1}\left(a_0+\sum_{j=1}^{m}a_jx_j\right) e^{-\sum_{j=1}^{m}\beta_j(x_j-x_{0j})^2}\ dx_1\,dx_2\ldots dx_m$$

$$\hat{y}(X_0) = \frac{1}{\prod_{j=1}^{m}q_j}\int_{x'_m}^{x''_m}\cdots\int_{x'_2}^{x''_2} e^{-\sum_{j=2}^{m}\beta_j(x_j-x_{0j})^2}R_1\ dx_2\ldots dx_m$$

where

$$R_1 = \int_{x'_1}^{x''_1}\left(a_0 + a_1x_1 + \sum_{j=2}^{m}a_jx_j\right)e^{-\beta_1(x_1-x_{01})^2}\ dx_1$$

$$R_1 = \left(a_0+\sum_{j=2}^{m}a_jx_j\right)\int_{x'_1}^{x''_1} e^{-\beta_1(x_1-x_{01})^2}\ dx_1 + a_1\int_{x'_1}^{x''_1} x_1e^{-\beta_j(x_1-x_{01})^2}\ dx_1$$

$$R_1 = (a_0 + a_1x_{01})q_1 + q_1\sum_{j=2}^{m} a_jx_j - \frac{a_1}{2\beta_1}(e^{-\beta_1(x_1''-x_{01})^2} - e^{-\beta_1(x_1'-x_{01})^2})$$

Thus,

$$\hat{y}(X_0) = \frac{1}{\prod_{j=1}^{m} q_j}\int_{x_m'}^{x_m''}\cdots\int_{x_3'}^{x_3''} e^{-\sum_{j=3}^{m}\beta_j(x_j-x_{0j})^2} R_2\ dx_3\ldots dx_m$$

where

$$R_2 = \int_{x_2'}^{x_2''} e^{-\beta_2(x_2-x_{02})^2}\left((a_0 + a_1x_{01})q_1 + q_1\sum_{j=3}^{m} a_jx_j - \frac{a_1}{2\beta_1}(e^{-\beta_1(x_1''-x_{01})^2} - e^{-\beta_1(x_1'-x_{01})^2}) + q_1a_2x_2\right) dx_2$$

$$R_2 = \left((a_0 + a_1x_{01})q_1 + q_1\sum_{j=3}^{m} a_jx_j - \frac{a_1}{2\beta_1}(e^{-\beta_1(x_1''-x_{01})^2} - e^{-\beta_1(x_1'-x_{01})^2})\right) q_2 + q_1a_2\left(\frac{-1}{2\beta_2}(e^{-\beta_2(x_2''-x_{02})^2} - e^{-\beta_2(x_2'-x_{02})^2}) + x_{02}q_2\right)$$

$$R_2 = \left((a_0 + a_1x_{01} + a_2x_{02})q_1q_2 + q_1q_2\sum_{j=3}^{m} a_jx_j\right) - \frac{a_1q_1q_2}{2\beta_1q_1}(e^{-\beta_1(x_1''-x_{01})^2} - e^{-\beta_1(x_1'-x_{01})^2}) - \frac{a_2q_1q_2}{2\beta_2q_2}(e^{-\beta_2(x_2''-x_{02})^2} - e^{-\beta_2(x_2'-x_{02})^2})$$

By doing this procedure successively we would have:

$$R_m = \left(\left(a_0 + \sum_{j=1}^{m} a_jx_{0j}\right) + \sum_{j=m+1}^{m} a_jx_j - \sum_{j=1}^{m}\frac{a_j}{2\beta_jq_j}(e^{-\beta_j(x_j''-x_{0j})^2} - e^{-\beta_j(x_j'-x_{0j})^2})\right)\prod_{j=1}^{m} q_j$$

$$R_m = \left(\left(a_0 + \sum_{j=1}^{m} a_jx_{0j}\right) - \sum_{j=1}^{m}\frac{a_j}{2\beta_jq_j}(e^{-\beta_j(x_j''-x_{0j})^2} - e^{-\beta_j(x_j'-x_{0j})^2})\right)\prod_{j=1}^{m} q_j$$

and so,

$$\hat{y}(X_0) = \frac{1}{\prod_{j=1}^{m} q_j}R_m = \left(a_0 + \sum_{j=1}^{m} a_jx_{0j}\right) - \sum_{j=1}^{m}\frac{a_j}{2\beta_jq_j}(e^{-\beta_j(x_j''-x_{0j})^2} - e^{-\beta_j(x_j'-x_{0j})^2})$$

Now, let $L_j = x_j'' - x_j'$; $j = 1, 2, \ldots, m$

$$\lim_{x_{0j}-x_j'\to L_j/2} q_j = \sqrt{\frac{\pi}{\beta_j}}\left(\Phi\left(\sqrt{2\beta_j}\frac{L_j}{2}\right) - \Phi\left(-\sqrt{2\beta_j}\frac{L_j}{2}\right)\right)$$

$$\lim_{x_{0j}-x_j'\to L_j/2} q_j = \sqrt{\frac{\pi}{\beta_j}}\left(2\Phi\left(L_j\sqrt{\frac{\beta_j}{2}}\right) - 1\right)$$

and ultimately,

$$\lim_{\substack{x_{0j}-x_j'\to L_j/2\\ j=1,2,\ldots,m}} \hat{y}(X_0) = \left(a_0 + \sum_{j=1}^{m} a_jx_{0j}\right) - \sum_{j=1}^{m}\frac{a_j}{2\beta_jq_j}(e^{-\beta_j(L_j/2)^2} - e^{-\beta_j(-L_j/2)^2})$$

$$\lim_{\substack{x_{0j}-x_j'\to L_j/2\\ j=1,2,\ldots,m}} \hat{y}(X_0) = a_0 + \sum_{j=1}^{m} a_jx_{0j} = y(X_0)$$

Therefore, the proof of Theorem 2.3 is completed. □

Appendix D. Proof of Theorem 2.4

In one hand we have:

$$\Delta X = X_0 + (\Delta x_1, \Delta x_2, \ldots, \Delta x_m)$$

$$y(X_0 + \Delta X) = y(X_0) + \sum_{j=1}^{m}\frac{\partial y(X_0)}{\partial x_{0j}}\ \Delta x_j = \left(a_0 + \sum_{j=1}^{m} a_jx_j\right) + \sum_{j=1}^{m} a_j\ \Delta x_j$$

On the other hand according to Theorem 2.3:

$$\hat{y}(X_0) = \left(a_0 + \sum_{j=1}^{m} a_jx_{0j}\right) - \sum_{j=1}^{m}\frac{a_j}{2\beta_jq_j}(e^{-\beta_j(x_j''-x_{0j})^2} - e^{-\beta_j(x_j'-x_{0j})^2})$$

Therefore,

$$\frac{\partial\hat{y}(X_0)}{\partial x_{0j}} = a_j - \frac{a_j}{2\beta_jq_j}(2\beta_j(x_j''-x_{0j})e^{-\beta(x_j''-x_{0j})^2} - 2\beta_j(x_j'-x_{0j})e^{-\beta_j(x_j'-x_{0j})^2})$$

Thus,

$$\lim_{x_{0j}-x_j'\to L_j/2}\frac{\partial\hat{y}(X_0)}{\partial x_{0j}} = a_j - \frac{a_j}{2\beta_jq_j}\left(2\beta_j\frac{L_j}{2}e^{-\beta(L_j/2)^2} - 2\beta_j\left(\frac{-L_j}{2}\right)e^{-\beta_j(-L_j/2)^2}\right)$$

$$\lim_{x_{0j}-x_j'\to L_j/2}\frac{\partial\hat{y}(X_0)}{\partial x_{0j}} = a_j - \frac{a_jL_j\sqrt{\beta_j}}{\sqrt{\pi}e^{\beta_jL_j^2/4}(2\Phi(L_j\sqrt{\beta_j/2}) - 1)}$$

Therefore,

$$\hat{y}(X_0+\Delta X)=\hat{y}(X_0)+\sum_{j=1}^{m}\frac{\partial\hat{y}(X_0)}{\partial x_{0j}}\ \Delta x_j$$

$$\hat{y}(X_0+\Delta X)=\left(a_0+\sum_{j=1}^{m}a_jx_j\right)+\sum_{j=1}^{m}a_j\ \Delta x_j - \sum_{j=1}^{m}\frac{a_jL_j\sqrt{\beta_j}}{\sqrt{\pi}e^{\beta_jL_j^2/4}(2\Phi(L_j\sqrt{\beta_j/2})-1)}\ \Delta x_j$$

$$\hat{y}(X_0+\Delta X)=y(X_0+\Delta X)-\sum_{j=1}^{m}\frac{a_jL_j\sqrt{\beta_j}}{\sqrt{\pi}e^{\beta_jL_j^2/4}(2\Phi(L_j\sqrt{\beta_j/2})-1)}\ \Delta x_j$$

Accordingly,

$$(y(X_0+\Delta X)-\hat{y}(X_0+\Delta X))^2=\left(\sum_{j=1}^{m}\frac{a_jL_j\sqrt{\beta_j}}{\sqrt{\pi}e^{\beta_jL_j^2/4}(2\Phi(L_j\sqrt{\beta_j/2})-1)}\ \Delta x_j\right)^2$$

Therefore, Theorem 2.4 is proved. □

Appendix E. Proof of Theorem 2.5

Consider (9) that represents the squared error and set $\beta_j=1/L_j^2$. We have:

$$(\Delta y)^2=\left(\sum_{j=1}^{m}\frac{a_j(x_j-x_{0j})}{\sqrt{\pi}e^{1/4}(2\Phi(\sqrt{2}/2)-1)}\right)^2$$

$$(\Delta y)^2=\frac{1}{(\sqrt{\pi}e^{1/4}(2\Phi(\sqrt{2}/2)-1))^2}\left(\sum_{j=1}^{m}a_j(x_j-x_{0j})\right)^2$$

According to the polynomial inequality:

$$\left(\sum_{j=1}^{m}a_j\right)^2\le\sum_{j=1}^{m}a_j^2$$

we have:

$$\left(\sum_{j=1}^{m}a_j(x_j-x_{0j})\right)^2\le\sum_{j=1}^{m}a_j^2(x_j-x_{0j})^2$$

Thus,

$$\frac{1}{(\sqrt{\pi}e^{1/4}(2\Phi(\sqrt{2}/2)-1))^2}\sum_{j=1}^{m}a_j^2(x_j-x_{0j})^2\ge(\Delta y)^2$$

$$\sum_{j=1}^{m}\frac{(x_j-x_{0j})^2}{b_j^2}\ge(\Delta y)^2$$

where, $b_j=(\sqrt{\pi}e^{1/4}(2\Phi(\sqrt{2}/2)-1))/a_j$.

Now, consider the below equation:

$$\sum_{j=1}^{m}\frac{(x_j-x_{0j})^2}{b_j^2}=(\Delta y)^2$$

which represents equation of a hyperellipse where $\sum_{j=1}^{m}(x_j-x_{0j})^2/b_j^2$ is the upper bound of the squared error for each point. Now, consider a special value such as $(\Delta y)^2$ for squared error. All points which the upper bound of their squared error, $\sum_{j=1}^{m}(x_j-x_{0j})^2/b_j^2$, is more than $(\Delta y)^2$ satisfy the below inequality:

$$\sum_{j=1}^{m}\frac{(x_j-x_{0j})^2}{b_j^2}\ge(\Delta y)^2$$

These are the points located outside of a hyperellipse. Similarly, all points which their upper bound of squared error, $\sum_{j=1}^{m}(x_j-x_{0j})^2/b_j^2$, is less than $(\Delta y)^2$ satisfy the below inequality:

$$\sum_{j=1}^{m}\frac{(x_j-x_{0j})^2}{b_j^2}\le(\Delta y)^2$$

These are the points inside a hyperellipse. When the upper bound of squared error for a point is less than (Δy^2) implies that the exact value of its squared error is less than $(\Delta y)^2$. Therefore, all points which their exact value of squared errors are less than $(\Delta y)^2$ are located inside a hyperellipse.

So, the proof of Theorem 2.5 is completed. □

References

[1] M.Y. Chen, D.A. Linkens, Rule-base self-generation and simplification for data-driven fuzzy models, Fuzzy Sets and Systems 142 (2004) 243–265.
[2] C.C. Chuang, S.F. Su, S.S. Chen, Robust TSK fuzzy modeling for function approximation with outliers, IEEE Transactions on Fuzzy Systems 9 (6) (2001) 810–821.
[3] M. Delgado, A.F. Gomez-Skarmeta, F. Martin, A fuzzy clustering-based rapid prototyping for fuzzy rule-based modeling, IEEE Transactions on Fuzzy Systems 5 (2) (1997) 223–233.
[4] M.R. Emami, I.B. Turksen, A.A. Goldenberg, Development of a systematic methodology of fuzzy logic modeling, IEEE Transactions on Fuzzy Systems 6 (3) (1998) 346–361.
[5] J. Espinosa, J. Vandewalle, Constructing fuzzy models with linguistic integrity from numerical data—AFRELI algorithm, IEEE Transactions on Fuzzy Systems 8 (5) (2000) 591–600.
[6] F. Hoppner, F. Klawonn, R. Kruse, T. Runkler, Fuzzy Cluster Analysis, John Wiley & Sons, Chichester, 1999, pp. 203–216.
[7] Y. Jin, Fuzzy modeling of high-dimensional systems: complexity reduction and interpretability improvement, IEEE Transactions on Fuzzy Systems 8 (2) (2000) 212–221.
[8] S.J. Kang, C.H. Woo, H.S. Hwang, K.B. Woo, Evolutionary design of fuzzy rule base for nonlinear system modeling and control, IEEE Transactions on Fuzzy Systems 8 (1) (2000) 37–45.
[9] E. Kim, M. Park, S. Ji, M. Park, A new approach to fuzzy modeling, IEEE Transactions on Fuzzy Systems 5 (3) (1997) 328–337.
[10] B. Kosko, Fuzzy Engineering, Prentice Hall International, New Jersey, 1997, pp. 139–173.
[11] B. Kosko, Optimal fuzzy rules cover extrema, International Journal of Intelligent Systems 10 (2) (1995) 249–255.
[12] C.N.P. Reyes, Coevolutionary Fuzzy Modeling, Springer-Verlag, Berlin, 2004, pp. 18–33.
[13] T.J. Ross, Fuzzy Logic with Engineering Applications, John Wiley & Sons, Chichester, 2004, pp. 476–521.
[14] M. Setnes, R. Babuska, U. Kaymak, H.R.N. Lemke, Similarity measures in fuzzy rule base simplification, IEEE Transactions on Systems, Man, and Cybernetics 28 (3) (1998) 376–386.
[15] M. Sugeno, T. Yasukawa, A fuzzy-logic-based approach to qualitative modeling, IEEE Transactions on Fuzzy Systems 1 (1) (1993) 7–31.
[16] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Transactions on Systems, Man, and Cybernetics 15 (1) (1985) 116–132.
[17] D. Tikk, G. Biro, T.D. Gedeon, L.T. Koczy, J.D. Yang, Improvements and critique on Sugeno's and Yasukawa's qualitative modeling, IEEE Transactions on Fuzzy Systems 10 (5) (2002) 596–606.
[18] L.X. Wang, J.M. Mendel, Generating fuzzy rules by learning from examples, IEEE Transactions on Systems, Man, and Cybernetics 22 (6) (1992) 1414–1427.

[19] O. Wolkenhauer, Data Engineering: Fuzzy Mathematics in Systems Theory and Data Analysis, John Wiley & Sons, New York, 2001, pp. 109–128.
[20] C.C. Wong, C.C. Chen, A hybrid clustering and gradient descent approach for fuzzy modeling, IEEE Transactions on Systems, Man, and Cybernetics 29 (6) (1999) 686–693.
[21] Z.Y. Xing, W.L. Hu, L.M. Jia, A fuzzy clustering based approach for generating interpretable fuzzy models, in: Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai, China, August 2004, 2004, pp. 2093–2097.
[22] R.R. Yager, D.P. Filev, Unified structure and parameter identification of fuzzy models, IEEE Transactions on Systems, Man, and Cybernetics 23 (4) (1993) 1198–1205.
[23] J. Zhao, V. Wertz, R. Gorez, A fuzzy clustering method for identification of fuzzy models for dynamic systems, in: IEEE International Symposium on Intelligent Control, OH, USA, August 1994, 1994, pp. 172–177.
[24] H.J. Zimmermann, Fuzzy Set Theory and Its Applications, Kluwer Academic Publishers, Boston, 1996, pp. 203–216.