Available online at www.sciencedirect.com

SCIENCE DIRECT®

Fuzzy Sets and Systems 153 (2005) 403–427

FUZZY sets and systems

www.elsevier.com/locate/fss

TaSe, a Taylor series-based fuzzy system model that combines interpretability and accuracy

L.J. Herrera*, H. Pomares, I. Rojas, O. Valenzuela, A. Prieto

Department of Computer Architecture and Computer Technology, University of Granada, 18071 Granada, Spain

Received 15 January 2004; received in revised form 22 November 2004; accepted 25 January 2005
Available online 16 March 2005
Communicated by Gomide

Abstract

Typically, Takagi–Sugeno–Kang (TSK) fuzzy rules have been used as a powerful tool for function approximation problems, since they have the capability of explaining complex relations among variables using rule consequents that are functions of the input variables. But they present the great drawback of the lack of interpretability, which makes them not to be so suitable for a wide range of problems where interpretability of the obtained model is a fundamental key. In this paper, we present a novel approach that extends the work by Bikdash (IEEE Trans. Fuzzy Systems 7 (6) (1999) 686–696), in order to obtain an interpretable and accurate model for function approximation from a set of I/O data samples, which make use of the Taylor Series Expansion of a function around a point to approximate the function using a low number of rules. Our approach also provides an automatic methodology for obtaining the optimum structure of our Taylor series-based (TaSe) fuzzy system as well as its pseudo-optimal rule-parameters (both antecedents and consequents).
© 2005 Elsevier B.V. All rights reserved.

Keywords: Curse of interpretability; Curse of dimensionality; Fuzzy system models; System identification; Function approximation; Complete rule-based fuzzy systems

1. Introduction

Function approximation deals with the identification of the underlying model present in a set of training I/O data points. Several approaches and paradigms have been applied to solve this problem [7,36,37].

* Corresponding author.
E-mail address: jherrera@atc.ugr.es (L.J. Herrera).

0165-0114/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.fss.2005.01.012

Among them, soft computing techniques [40] have proved successful in dealing with this topic. Though each one of them has its own well-known number of advantages, Fuzzy Logic has the advantage that the underlying model of the final designed system is transparent to the scientist/engineer designer. Fuzzy Logic is simple on its roots and has also the possibility of using linguistic values to describe the input and output of the system, thus improving the understandability of the system [11].

When using Fuzzy Logic for function approximation, two main approaches might be taken. On one hand, we have clustering techniques that perform a marginal subdivision of the input space, depending on the number of rules used to reach the objective [1,3,13,26,32,33]. This approach has the drawback that the whole input space might not be covered properly. On the other hand, grid-based fuzzy systems (GBFS) provide a thorough coverage of the whole input space which has made them widely used in the literature [29,31,37].

However this last approach, i.e., GBFS, has an important drawback; the number of rules increases exponentially with the number of input variables and the number of membership functions (MFs) per variable, also known as the curse of dimensionality problem [2]. For a medium-complexity problem, we might end up with hundreds or even thousands of rules which leads to two main problems: first, this high number of rules makes any further computational treatment of this set of rules extremely difficult and second and more important, the transparency of fuzzy logic becomes useless when having such a huge number of rules for medium-sized problems. The understandability of the system vanishes and the advantage of using fuzzy logic disappears.

Takagi–Sugeno–Kang (TSK) rules and the fuzzy inference method proposed by these authors [35] are widely utilised in function approximation problems using fuzzy logic. The main difference with more traditional (i.e., [22]) fuzzy rules is that the consequent of the rules are a function of the input variables values. This approach has demonstrated to have a powerful representative capability, being able to describe non-linear mappings using a small number of simple rules. But in spite of this, TSK rules suffer from the lack of interpretability [39].

Nevertheless, the still excessive number of rules and the lack of interpretability caused by the usage of zero or one-order TSK rules, though providing excellent performance results for function approximation, make the usage of fuzzy logic useless for several types of problems where the understandability and interpretability or the possibility of providing a manageable model for further work is a must. In this paper, we present an approach that can overcome these two problems: the curse of dimensionality and the *curse of interpretability*, keeping the original key idea of GBFSs.

Many works have addressed the problem of the loss of interpretability in fuzzy modelling [14], most of them based on Mamdani-type fuzzy systems [27]. In general, there are different aspects related to the interpretability concept in TSK systems, some of which also apply to Mamdani FS. We have already mentioned the need of maintaining a low number of rules in the system in order to keep its global interpretability, but the transparency [41] (generation of interpretable fuzzy sets, i.e., transparent partition of the input space) and the interaction of the global and local models, are also key concepts in the interpretability of a TSK fuzzy model [18,39].

In relation to the transparency concept, several works have addressed this problem [27,33]. In this work, we propose to use a grid partitioned (GBFS) input space that avoids overlapping. With respect to the decrease in the number of rules for a GBFS, little work has been done with respect to the operation of high-order TSK rule consequents for function approximation. We will demonstrate the aptitude of this kind of fuzzy rules for function approximation and their good performance, while reducing the number of rules needed to perform the approximation. The proposed method furthermore, brings interpretability to the local models that compose the global TaSe System, using a key concept in modelling non-linear systems, as is the Taylor series expansion of a function around a point.

We also present an automatic approach for parameter optimisation given a set of I/O training data and an initial configuration of MFs per input variable. This automatic algorithm will find a pseudo-optimal localisation of the centres of the MFs for each variable to get the lowest error (in the sense of least squares) possible for the given configuration.

Structure identification has also been a focus of study in the literature. It deals with which the best structure for a GBFS given some requirements on accuracy, interpretability and/or complexity of the final model is. In our case we have designed a structure identification approach for our proposed adapted high-order TSK fuzzy system. It automatically identifies the pseudo-optimal simplest structure given a training set of I/O data points.

The structure of the remaining of the paper stays as follows: Section 2 describes high-order TSK fuzzy rules with the algorithm to identify the optimal coefficients for the polynomial consequents and the possible advantages of using this type of rules. Section 3 deals with the interpretability of the system, property achieved thanks to the use of a specific type of MFs and a rule consequent structure that resembles the Taylor Series Expansion around a point (the centres of the MFs), thus stating the basis of the Taylor series-based (TaSe) fuzzy systems. Section 4 explains the method used for parameter identification, that is, the localisation of the pseudo-global optimum of the parameters defining the fuzzy system. Section 5 presents the part of the algorithm that deals with the structure identification problem and which finally will identify the simplest configuration. Section 6 presents and analyses the results obtained when applying the presented method to three examples commonly used in the literature. Finally, some conclusions are drawn in Section 7.

2. Fuzzy systems with high-order TSK rules

A TSK fuzzy model consists of a set of ‘K’ IF-THEN rules that typically have the form

$$R^k : \text{ IF } x_1 \; is \; MF_1^k \text{ AND } \ldots \text{ AND } x_n \; is \; MF_n^k \text{ THEN } y = \alpha_0^k + \alpha_1^k x_1 + \cdots + \alpha_n^k x_n, \tag{1}$$

where the MF_i^k are fuzzy sets characterised by $MF_i^k(x_i)$, α_i^k are real-valued parameters and x_i are the input variables. The consequent of the rules in the majority of the systems presented in the literature is simply a scalar, i.e., zero-order TSK rules. The performance of this type of systems has been demonstrated to be reasonable for several problem examples [28,31,36].

The greatest drawback that emerges when using GBFSs with this kind of TSK rules (somehow attenuated for linear-consequent rules, as in (1)) is the unmanageable number of rules that arises for problems with moderated complexity. The number of rules for a problem with n input variables and mf_i for each input variable is equal to

$$K = \prod_{i=1}^{n} mf_i. \tag{2}$$

The number of rules is hence an exponential function of the number of input variables and the number of MFs per input variable. The resulting number of rules might render the system non-understandable and unmanageable, thus loosing its usability. Several approaches that have attempted to overcome this obstacle have arisen in the literature in previous years with partial success [5,9,10]. Nevertheless, most of them break the original Grid-based structure to solve the problem, thus losing the advantages and intuition that GBFS provide.

Now, notice that as Buckley noted [6], the consequent of TSK fuzzy rules can be generalised as follows:

$$R^k: \text{ IF } x_1 \, is \, MF_1^k \text{ AND } \dots \text{ AND } x_n \, is \, MF_n^k \text{ THEN } y = Y_k(\vec{x}), \tag{3}$$

where $Y_k(\vec{x})$ is a polynomial of any order. For s-order polynomial consequents, they can be expressed as

$$Y_k(\vec{x}) = w_0^k + \vec{w}_1^k \cdot \vec{x} + \vec{x}^{\mathrm{T}} W_2^k \vec{x} + \cdots + \left\langle W_s^k \cdot (\vec{x} \otimes \cdots \otimes \vec{x}) \right\rangle, \tag{4}$$

where w_0^k is a zero-order coefficient, $\vec{w}_1^k$ is a vector of 1-order coefficients, W_2^k is a triangular matrix of 2-order coefficients and W_s^k is a triangular s-dimensional matrix of coefficients and $\otimes$ is the tensor product.

In this paper, we will make use of high-order polynomial rule consequents to overcome the curse-of-dimensionality problem. The main advantage of this type of fuzzy rules, compared to that offered by zero and first-order TSK rules, is the increase in expressive power that each rule can provide by itself. That is, a fewer number of rules might be able by themselves to identify the implicit function underlined by a set of I/O training points (additional advantages of high-order rules will be discussed in further sections).

In order to take an insight of how this kind of fuzzy systems work, let us recall the expression of the output of a fuzzy system for a given input $\vec{x}$, considering weighted-average defuzzification

$$F(\vec{x}) = \frac{\sum_{k=1}^{k} \mu_k(\vec{x}) \cdot Y_k(\vec{x})}{\sum_{k=1}^{K} \mu_k(\vec{x})} \tag{5}$$

being $\mu_k(\vec{x})$ the activation value of the rule k, which can be expressed as (using the product as the T-norm)

$$\mu_k(\vec{x}) = MF_1^k(x_1) \cdot MF_2^k(x_2) \cdot \ldots \cdot MF_n^k(x_n). \tag{6}$$

For the sake of simplicity, now consider a one-dimensional (1-D) input space and a triangular-partitioned input variable [31,32] with two MFs. In this case, two rules ($k = 1, 2$) will be activated by any single point in the input interval.

Considering zero-order TSK fuzzy rules, the output can then be expressed as (given the addition-to-unity property [29])

$$\begin{aligned} F(x) &= \mu_1(x) \cdot y_1 + \mu_2(x) \cdot y_2 = \mu_1(x) \cdot y_1 + \big(1 - \mu_1(x)\big) \cdot y_2 \\ &= y_2 + \mu_1(x) \cdot (y_1 - y_2). \end{aligned} \tag{7}$$

$F(x)$ thus resulting in a first-order polynomial expression in x (since $\mu_1(x)$ is a first-order polynomial, see [29]).

Considering now s-order polynomial consequent rules, the output becomes

$$F(x) = Y_2(x) + \mu_1(x) \cdot (Y_1(x) - Y_2(x)), \tag{8}$$

$F(x)$ will hence result in a $s+1$-order polynomial expression with coefficients coming from the parameter coefficients of the two rules, therefore being a clearly more powerful approximator.

Now let us go over on how to estimate from a given set of I/O training data points, the optimal values for the high-order polynomial rule coefficients. The least-squares error (LSE) approach is commonly the method used to optimise such consequents. LSE tries to minimise the error function

$$J = \sum_{m \in D} \left(y^m - F\left(\vec{x}^m\right)\right)^2, \tag{9}$$

where y^m is the desired output for point $\vec{x}^m$ in the data set D, and $F\left(\vec{x}^m\right)$ (5) is the output of our approximator system.

In the particular case of second-order polynomial coefficients, the number of parameters to be optimised by least squares is given by

$$K \cdot \left(1 + n + \tfrac{1}{2}\left(n^2 + n\right)\right), \tag{10}$$

where K is the number of rules and n is the dimension of the input space.

Notice the great complexity of the problem when having a high number of input variables with a moderated number of MFs per variable. The degree of redundancy in the parameters could be considerable, making the problem ill-conditioned. Due to the fact that the output function (5) is linear with respect to all the consequent parameters (see appendix), several well-known mathematical methods could be applied to extract one optimum solution. Among these methods, singular value decomposition [12] (SVD) has been successfully implemented since it allows us to discard little significant values avoiding redundancy and unmanageable solutions. The efficiency of this method will be shown in the examples.

When facing a specific problem, it could be arguable if it is more convenient to use a big amount of zero or first-order TSK rules than a few number of high-order rules. From the computational point of view, the difference for a given error tolerance is not obvious (see Section 6). From the number-of-fuzzy-rules point of view it has been already noticed the dramatic decrease that could be achieved using high-order TSK rules. Finally, from the interpretability point of view, traditionally zero-order fuzzy rules have been chosen due to its easier understandability. However, as the number of fuzzy rules increases, the interpretability property dramatically vanishes (the curse of interpretability problem). High-order TSK rules are commonly regarded as non-interpretable [39] but, as we will see in the next section, under certain conditions it will be possible to endow this class of fuzzy rules with interpretability, thus gathering both the low number-of-rules property and the interpretability property in the same fuzzy system.

3. Interpretability issues for TaSe fuzzy systems

So far, we have presented how to use high-order polynomial consequent rules in TSK fuzzy systems, and the performance advantages that it might provide. Now let us analyse the extra properties that can be added to our fuzzy system by using a specific type of MFs. As we will see, these extra properties will make this type of GBFS suitable and convenient for a wide range of problems.

3.1. A brief introduction to Taylor series expansion

Let $f(x)$ be a function defined in an interval with an intermediate point a, for which we know the derivatives of all orders. The first-order polynomial

$$p_1(x) = f(a) + f'(a)(x - a) \tag{11}$$

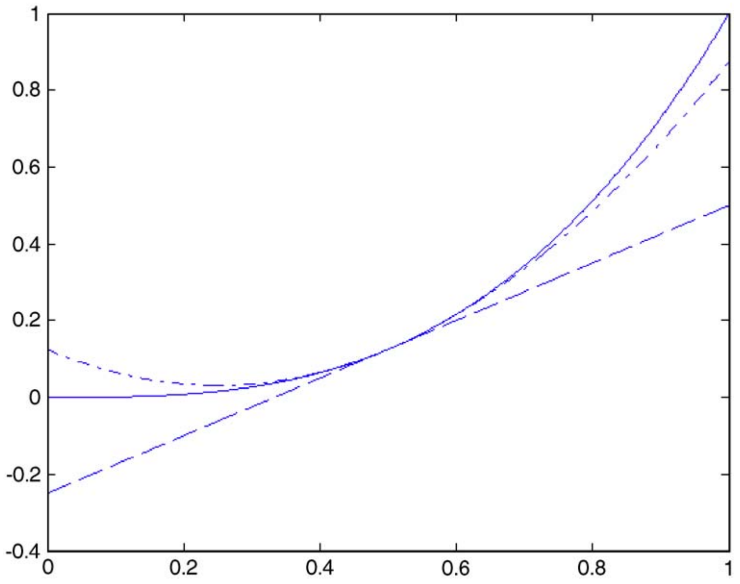

Fig. 1. Taylor series expansion example. The original function (solid line) is $f(x) = x^3$. See how the polynomial $p_1(x)$ for $a = 0.5$ is tangent to $f(x)$ at this point (dashed line). $p_2(x)$ (dashed-dotted line) performs a very good approximation for the points in the vicinity of $a = 0.5$. Note that since $f(x)$ is a third-order polynomial, $p_3(x)$ would be exactly the same as $f(x)$.

has the same value as $f(x)$ in the point $x = a$ and also the same first-order derivative at this point. Its graphic representation is a tangent line to the graph of $f(x)$ at the point $x = a$.

Considering the second derivative for $f(x)$ in $x = a$, the second-order polynomial

$$p_2(x) = f(a) + f'(a)(x-a) + \tfrac{1}{2} f''(a)(x-a)^2 \tag{12}$$

has the same value as $f(x)$ in $x = a$, and the same first- and second-order derivatives at this point. For the points in the vicinity of $x = a$, $f(x)$ will be more similar to $p_2(x)$ than $p_1(x)$. Therefore, we can expect that forming a *n*th-order polynomial $p_n(x)$ from the *n*th first derivatives of $f(x)$ in $x = a$, in the same way as we did for $p_1(x)$ and $p_2(x)$, the resulting polynomial will get very close to $f(x)$ in the neighbourhood of $x = a$. See Fig. 1 as an example.

Taylor theorem states that if a function $f(x)$ defined in an interval has derivatives of all orders, it can be approximated near a point $x = a$, as its Taylor series expansion around that point

$$\begin{aligned} f(x) = {} & f(a) + f'(a)(x-a) + \frac{1}{2} f''(a)(x-a)^2 \ldots \\ & + \frac{1}{n!} f^{(n)}(a)(x-a)^n + \frac{1}{(n+1)!} f^{(n+1)}(c)(x-a)^{n+1}, \end{aligned} \tag{13}$$

where the last term refers to the error term in which *c* is a point between *x* and *a*.

Taylor series expansion opens a door for the approximation of any function through polynomials, that is just like to say through the addition of a number of simple functions. It is therefore a fundamental key in the field of Function Approximation Theory and Mathematical Analysis. Taylor series expansion will also provide us a way to bring interpretation to the TSK fuzzy systems by taking a certain type of rule antecedents that has a number of interesting properties.

In the n-dimensional case, Taylor series expansion is adapted in the following form:

$$f(\vec{x}) = f(\vec{a}) + (\vec{x}-\vec{a})^{\mathrm{T}}\left[\frac{\partial f}{\partial \vec{x}_i}(\vec{a})\right]_{i=1\ldots n} + \frac{1}{2}(\vec{x}-\vec{a})^{\mathrm{T}} W (\vec{x}-\vec{a}) + \frac{1}{3!}\left\langle W^3 \cdot (\vec{x}-\vec{a}\otimes\vec{x}-\vec{a}\otimes\vec{x}-\vec{a})\right\rangle + \cdots, \tag{14}$$

where W is a triangular matrix of dimensions $n \times n$ having the values

$$\left[\frac{\partial f}{\partial \vec{x}_i \partial \vec{x}_j}(\vec{a})\right]_{i=1\ldots n;\ j=i\ldots n}$$

and W^3 is a 3-D matrix having the corresponding multi-partial derivatives for $\vec{x}=\vec{a}$.

It would be a considerable powerful achievement if we could interpret the consequents of the rules as the (truncated) Taylor series expansion for the approximated function in the centres of each rule. Before going further, let us study what requirements our MFs must accomplish in order to be able to give this interpretation to the output of the system.

3.2. MFs to preserve the interpretability of high-order fuzzy rules

In order to be able to interpret the output of a rule as the truncated Taylor series expansion around at certain point, a couple of properties are desired. First, the overlapping degree of all MFs should vanish to zero at every rule centre, in order the output of the system at each rule centre to be influenced exclusively by its corresponding rule. And second, the type of MFs must allow the output of the system to be continuous and n times differentiable where n indicates the order of the consequent polynomial of the fuzzy rule, in order to be able to identify the rule consequent coefficients as the partial derivatives of order n of the function output at the centres.

Orderly local membership function (OLMF) Bases of order p over a m-dimensional grid G have a number of interesting properties as noted by Marwan Bikdash [4]. The requirements that a set of MFs in a grid must fulfil to form a OLMF basis are basically

- all MFs are local (i.e., non-negative and vanishing with the distance), defined in a delimited domain and of the same type
- every MF extreme point must coincide with the centre of the adjacent MF (they form a partition, thus avoiding uncontrolled overlapping of the MFs [27,33])
- all MFs are p times differentiable and the pth derivative of the MF is continuous in all its domain
- the pth derivative of the MF vanishes at its centre and at its boundaries
- the basis must accomplish the *addition-to-unity* property [32].

Notice that for the most common types of MFs, these requirements are not fulfilled. For example for triangular partition, the MFs are not derivable in the centres of the rules (neither it happens in the trapezoidal partition); for the Gaussian MF partition, although its gradient vanishes at the centres and the MFs is p times differentiable, the MFs are not local. OLMF bases assure that the output of the system will be continuous and p-times derivable, and that the composing MFs are local.

In order to construct a local MF that is consistent with all these requirements, let us consider a spline MF defined by the following three parameters $[a, b, c]$ where, a and c are the boundaries of the local MF and b is the centre of the MF.

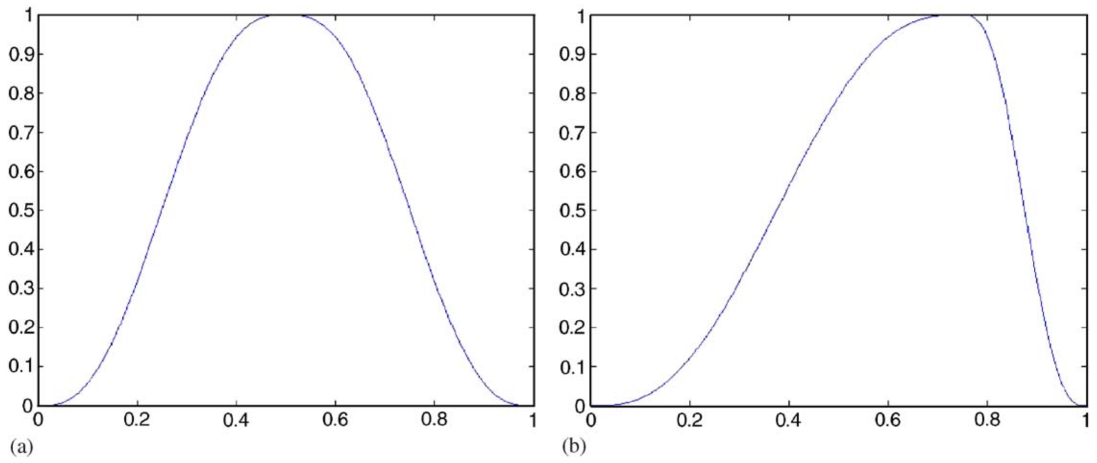

Fig. 2. Two examples of the membership degree for the explained type of TaSe MF.

In the special case in which $p = 2$, i.e., bases of order two, the above requirements amounts to the following conditions on the MF

$$\begin{aligned} &MF\,(a) = 0, \quad MF\,(b) = 1; \\ &MF'\,(a) = 0, \quad MF'\,(b) = 0; \\ &MF''\,(a) = 0; \quad MF''\,(b) = 0. \end{aligned} \tag{15}$$

Using Hermite Interpolation, the $MF(x, [a, b, c])$ for x in $[a, b]$, i.e., the left-hand side of the MF, gives

$$MF_i\,(x, [a, b, c]) = \frac{1}{(b-a)^3}\,(x-a)^3 \left[1 - \frac{3\,(x-b)}{(b-a)} + \frac{6\,(x-b)^2}{(b-a)^2}\right] \quad \text{for } x \in [a, b]\,. \tag{16}$$

Due to the *addition-to-unity* property requirement, the right-hand side of the MF will be

$$MF_i\,(x, [a, b, c]) = 1 - MF_{i+1}\,(x, [b, c, d]) \quad \text{for } x \in [b, c]\,. \tag{17}$$

Fig. 2 shows a graphical example of this kind of MFs. It can be noted apart from the continuity and derivability of the function, that the gradient of the MF vanishes in the centre and in the boundaries of the interval where it is defined.

Now, given any input point, the final formula for the output of the system is simplified to

$$F(\vec{x}) = \sum_{k=1}^{K} \mu_k\,(\vec{x})\,Y_k\,(\vec{x})\,, \tag{18}$$

where the denominator of (5) becomes 1 due to the addition-to-unity property and where it must be recalled that the rules have the form (3) and that the output of each rule is a polynomial in the form (14).

3.3. Interpretability of the fuzzy rules

Marwan Bikdash demonstrated in [4] that given a complete TSK rule-based fuzzy system, where

1. the input MFs form a OLMF basis of order p for every input dimension and
2. the consequent-side of each rule is written in the rule-centred form shown in (3) and (14), being $Y_k(\vec{x})$ polynomials of degree n,

then for $n \Leftarrow p$, every $Y_k(\vec{x})$ can be interpreted as a truncated Taylor series expansion of order n of the output of the fuzzy system $F(\vec{x})$ about the point $x = a$, the centre of the kth rule.

Considering therefore that we have a method to obtain the optimal high-order TSK rule consequent coefficients (14) for function approximation given a training set of I/O data points, and a MF distribution that form a set of OLMF basis, then we can interpret the consequents of the rules $Y_k(\vec{x})$ as the truncated Taylor Series expansion around the centres of the rules of the output of the system; and, moreover, of the underlying modelled function given by the I/O training data points. In the limit case where the function is perfectly approximated by our system, the rule consequents will coincide with the Taylor Series expansions of that function about the centre of each rule, having reached total interpretability and total approximation for the function to be approximated. Our TaSe Fuzzy System comprising the type of rule antecedents explained in Section 3.2 and the type of rule consequents explained in Section 3.1, provides a general fully interpretable model, and as we will see in the next sections, a general accurate model for function approximation from a set of I/O data points.

In [4], Bikdash used directly the (available) Taylor series expansion of the function around the rule centres to approximate the function with the TSK fuzzy system. Notice that these rule consequents though having strong interpretability, are not the optimal consequents in the least-squares sense; note that the Taylor series expansion is an approximation for a function in the vicinity of the reference point. Therefore even using a high number of MFs, the error obtained by the method in [4] is seldom small enough (compared to a system with similar complexity with consequents optimised by LSE) and hence, the system output barely represents a good approximation of the data we are modelling.

Consider also that for function approximation we usually have the only information of the I/O training data set. No additional information about the function to be approximated is given, neither the derivatives of the function w.r.t. any point. Also no accurate enough method exists to obtain the derivatives from the training points to perform the approximation as the author did.

Finally, the method we are presenting in this paper will also deal with finding the optimal points to set the rule centres to obtain both interpretability and pseudo-optimal function approximation as we will see in next section.

4. Parameter adjustment

Several authors have dealt with the problem of parameter adjustment for function approximation using GBFSs [19,25,29,36,39]. Some of them have centred their research in the use of genetic algorithms due to the high exploring capability associated with them. Nevertheless, the computational and time cost that this type of approaches has, sometimes is too expensive. In our case, as mentioned above in Section 2, the rule consequent coefficients can be optimally obtained using least squares and SVD to solve the resulting linear equation system. As to the MF centres, the Levenberg–Marquardt optimisation procedure [23] has been employed, starting from a previously selected initial configuration, to reach a pseudo-optimal configuration of the centres of the MFs for all the variables.

That initial configuration of the centres of the MFs for the Levenberg–Marquardt algorithm is calculated using the approach presented by Pomares et al. [29] and which will be explained below. The use of this method, called the ‘error equidistribution method’, was successfully tested in [28,29] for a triangular partition configuration and constant consequent rules. As we will see in the simulations section, this approach also works properly for our Tase fuzzy system with OLMF bases and Taylor series-based TSK rules.

4.1. Searching the starting point: the error equidistribution method

The objective of this approach is to look for that configuration which homogeneously distributes the error throughout all regions defined by the MF-grid. This idea can be mathematically translated into having at each side of every MF centre the same amount of error, according to the error function given in (9) and the training data set D. That is

$$\sum_{\substack{m\in D\\ x_n^m\in\left[c_n^{i_n-1},c_n^{i_n}\right]}} \mathrm{e}^2(\vec{x}^m) \approx \sum_{\substack{m\in D\\ x_n^m\in\left[c_n^{i_n},c_n^{i_n+1}\right]}} \mathrm{e}^2(\vec{x}^m) \tag{19}$$

for the i_nth centre defined in the input variable x_n $c_n^{i_n}$.

The method consists in an iterative process with two phases: one for calculating a slope parameter for each centre, and a second one for centres moving according to these slope values.

In the first phase, centre $c_n^{i_n}$ is associated with a slope value $p_n^{i_n}$

$$p_n^{i_n} = \frac{1}{\sigma_y^2}\left(\sum_{\substack{m\in D\\ x_n^m\in\left[c_n^{i_n-1},c_n^{i_n}\right]}} \left(y^m - F\left(\vec{x}^m\right)\right)^2 - \sum_{\substack{m\in D\\ x_n^m\in\left[c_n^{i_n},c_n^{i_n+1}\right]}} \left(y^m - F\left(\vec{x}^m\right)\right)^2\right) \tag{20}$$

being σ_y^2 the standard deviation of the output data, here used as a normalisation constant. A positive value of the parameter $p_n^{i_n}$ means that the contribution of the left-hand side of the MF to the error is higher than the right-hand side one; therefore we would have to move the centre of the MF to the left to counteract this effect, and vice versa.

Once the slope parameter has been calculated for all MF centres of the system, in phase two of the process, we perform the following movement of the centres:

$$\Delta c_n^{i_n} = \begin{cases} \dfrac{c_n^{i_n-1} - c_n^{i_n}}{b} \dfrac{p_n^{i_n}}{p_n^{i_n} + \dfrac{1}{T_n^{i_n}}} & \text{if } p_n^{i_n} \geqslant 0 \\[2ex] \dfrac{c_n^{i_n+1} - c_n^{i_n}}{b} \dfrac{\left|p_n^{i_n}\right|}{\left|p_n^{i_n}\right| + \dfrac{1}{T_n^{i_n}}} & \text{if } p_n^{i_n} < 0. \end{cases} \tag{21}$$

Here b is the active radius, which is the maximum variation distance and is used to guarantee that the order of the MF location remains unchanged; $T_n^{i_n}$ is the temperature which will control how far the centre will be moved in each iteration and which will be decreased as the algorithm finds its equilibrium. The approach described will work iteratively moving the centres until the error on each side of the MF centres keeps balanced. In this work we will use $b = 2$ and $T_n^{i_n} = 100$.

4.2. Reaching the solution: the Levenberg–Marquardt algorithm

In the previous step, we have reached a configuration assumed to be near a good (pseudo-optimum) solution, the last step is to find the exact local minimum that we are looking for. To accomplish this task, numerous methods are available in the bibliography (steepest descent, conjugate gradient, Newton–Ramphson method, Levenberg–Marquardt method, etc.) In this paper, we use the Levenberg–Marquardt algorithm since its characteristics of robustness and efficiency make it especially suitable for this kind of optimisation problems. The explicit expressions for the partial derivatives with respect to every parameter involved in the fuzzy system can be found in appendix.

5. Structure identification of the TaSe fuzzy system

Little work has been done for structure identification in GBFSs. Genetic algorithms have been also the main focus of research for this problem. Few automatic algorithms have been proposed [25,28,34].

Suppose that we have a training data set and a validation data set for the structure identification algorithm. The starting point for our structure identification approach will be the simplest case of configuration, having one MF per input variable. A single rule in the form (3) is therefore given with its consequent in the form (14). As stated before, using least squares and linear regression methods, we can obtain the optimal coefficients for the consequent part of that single rule to best approximate the training data set.

Now from this simple configuration, new MFs will be added to the system while the validation error does not increase. That is, we will iteratively add one MF in the input variable where the error decreases more, until a final configuration where the lowest validation error has been reached. From this approach now comes the question of how we will identify the variable in which the error will decrease more, and in which the MF should be added. First, notice that adding one MF on any input variable j in our TaSe fuzzy system means

- Increasing considerably the approximative power of the system since this implies the addition of $\prod_{i=1,\, i\neq j}^{n} mf_i$ rules.
- Due to this expressive power and the need to avoid redundancy, parameter adjustment must be taken into account in order to make a correct decision. If we do not carefully adjust the centres of the MFs when deciding about taking or not a certain system configuration, unneeded MFs and rules might be unnecessary added, thus considerably increasing the complexity of the system as noted in Eqs. (2) and (10).
- Besides, adding one single second-order rule to the system implies the addition of

$$\left(n^2+n\right)/2+n+1 \tag{22}$$

coefficients in the consequent part of the rule, where n is the number of input variables.

Notice also that due to the high expressive power of each one of the Taylor series-based fuzzy rules, only few MFs will typically be needed per input variable.

Now, having exposed these reasons, we should be very careful with the selection of an algorithm for structure identification. Genetic algorithms, for example, become unfeasible due to its low efficiency when dealing with this type of problems. A greedy approach is provided now that assures to find a minimal pseudo-optimal configuration in an automatic way. We will simply test every possible new configuration, adding one MF on each input variable, and we will retain that configuration where the validation error is lower. This approach is possible thanks to the fact that we have provided an efficient and automatic parameter adjustment algorithm with optimal rule consequent coefficients selection and pseudo-optimal MF centre adjustment. The final algorithm stays as follows:

```
While validation error decreases (or minimum validation error limit not reached)
    For $I = 1..$ number of input variables
        Consider adding 1 MF to variable I
        Optimise centres and consequents (Sections 2 and 4)
        Get the error
    End
    Add 1 MF to the variable where adding 1 MF resulted in a lower validation error.
End.
```

This simple approach will assure to find a pseudo-optimal configuration with the lowest error, given the training and the validation data sets, using the training set for optimal consequent coefficients calculation and MF centres adjustment and the validation set for structure identification.

6. Simulations

This section provides three examples commonly used in the literature to clarify the main characteristics of the proposed method. As it is usual in papers concerning the problem of function approximation, examples of analytical functions are used, as these enable the estimated function to be compared with the original one, for any desired point. In all cases, the algorithm starts by assigning one MF per input variable, and thus there is an initial fuzzy system with just one rule independent of the inputs.

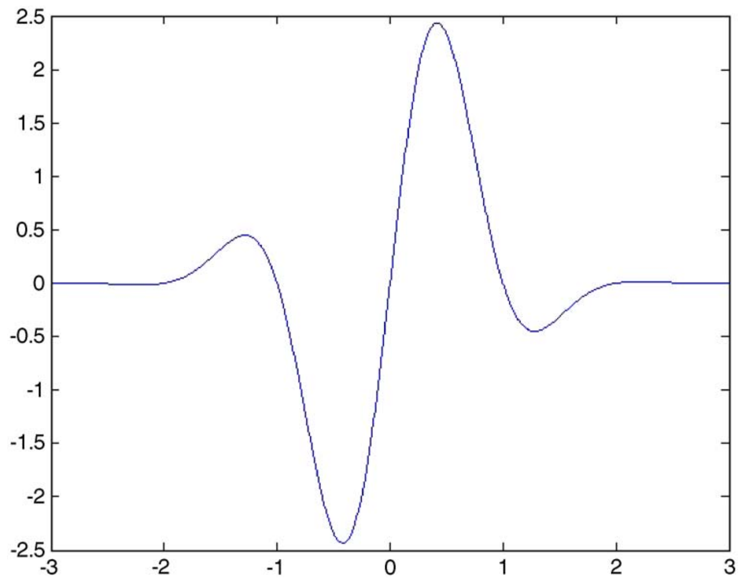

Fig. 3. Original noiseless function to be approximated.

The error measure used in the two first examples is the normalised root-mean square error (NRMSE) that is defined as

$$NRMSE = \sqrt{\frac{\overline{e^2}}{\sigma_y^2}}, \tag{23}$$

where σ_y^2 is the variance of the output data, and $\overline{e^2}$ is the mean-square error between the system output and the I/O data set output. In this way, the *NRMSE* index describes the performance of the approximation, making it independent of scale factors or number of data. For the third example (the Mackey–Glass time series) the root-mean square error (*RMSE*) ($\sqrt{\overline{e^2}}$) has been used since it's the measure mostly used in the literature for this specific benchmark.

6.1. Detailed application of the algorithm to a 1-D function

To demonstrate the proposed methodology, we will consider a representative example for the problem of function approximation and for the problem of the curse of interpretability. A number of well-known example functions can be found in the literature for both issues. The whole algorithm will be applied to a 1-D function, presented in [24] and also studied in [29] and for which we will apply our model, which, apart from providing excellent results for function approximation, will bring interpretability to the model, thanks to the Taylor series-based fuzzy rules.

The function presented by J.H. Nie (see Fig. 3) can be expressed as

$$y(x) = 3\mathrm{e}^{-x^2} \sin(\pi x) + \xi \quad \text{where } x \in [-3, 3] \text{ and } \xi \text{ noise.} \tag{24}$$

Since it is a 1-D function, our algorithm works trivially, starting from one MF in the single input variable, and then adding new MFs until the validation error stops decreasing. In every stage of the algorithm, the centres are calculated using the procedures presented in Section 4. Least squares will work in every iteration for the calculation of the optimal rule consequent coefficients for the given MF centres distribution.

To demonstrate the execution of the algorithm, we will take an execution example that has 100 randomly distributed training points with Gaussian noise 0.1, and 100 validation points of the same characteristics. We will now see in detail how the algorithm evolves. First, taking only one MF, the single resultant rule centred in $x = 0$ is

$$\text{IF } x \text{ IS } 0 \text{ THEN } y = -0.0\,(x-0)^2 + 0.037\,(x-0) + 0.11 \tag{25}$$

with a validation $NRMSE = 0.998$.

Adding one more MF to the system, results in a system with the following two rules, centred at the definition interval boundaries

$$\begin{aligned}&\text{IF } x \text{ IS } -3 \text{ THEN } y = 4.77(x+3)^2 + 0.63(x+3) - 0.51,\\ &\text{IF } x \text{ IS } 3 \text{ THEN } y = -4.78(x-3)^2 + 0.59(x-3) + 0.47\end{aligned} \tag{26}$$

with $NRMSE = 0.846$.

Once a third MF is added, the parameter adjustment algorithm explained in Section 4 will be executed to optimise the position of the middle MF. Since the boundaries of each MF coincides with the centres of its adjacent MFs in every input variable (see Section 3.2), only the centres of the MFs must be optimised, discarding those two centred at the domain $[-3, 3]$ boundaries. The initial value of the mobile MF centre will be 0, since the MFs are initially equally distributed in the interval $[-3, 3]$. After the execution of the error equidistribution method, the value for the variable centre is 1.029; and with this new initial centre configuration, the Levenberg–Marquardt Algorithm reaches the pseudo-optimal value for the variable centre equal to 1.066, and the rules are finally expressed as

$$\begin{aligned}&\text{IF } x \text{ IS } -3 \text{ THEN } y = -18.7(x+3)^2 + 0.81(x+3) + 0.35,\\ &\text{IF } x \text{ IS } 1.07 \text{ THEN } y = 16.1(x-1.07)^2 - 3.41(x-1.07) - 0.08,\\ &\text{IF } x \text{ IS } 3 \text{ THEN } y = -8.41(x-3)^2 + 4.09(x-3) + 0.50.\end{aligned} \tag{27}$$

Resulting in a model with validation NRMSE = 0.434.

Adding one more MF implies the optimisation of two MF centres. Both begin with equidistributed values in the interval $[-3, 3]$, thus -1 and +1. The error equidistribution method finds the new initial configuration of MF centres -0.8 and 0.59. The Levenberg–Marquardt algorithm, using this new initial configuration, finds the sub-optimal values -0.71 and 0.71. The error obtained with this configuration is $NRMSE = 0.096$

$$\begin{aligned}&\text{IF } x \text{ IS} - 3 \text{ THEN } y = -7.60(x+3)^2 + 0.93(x+3) - 0.04,\\ &\text{IF } x \text{ IS} - 0.71 \text{ THEN } y = 2.62(x+0.71)^2 - 6.03(x+0.71) - 1.37,\end{aligned}$$

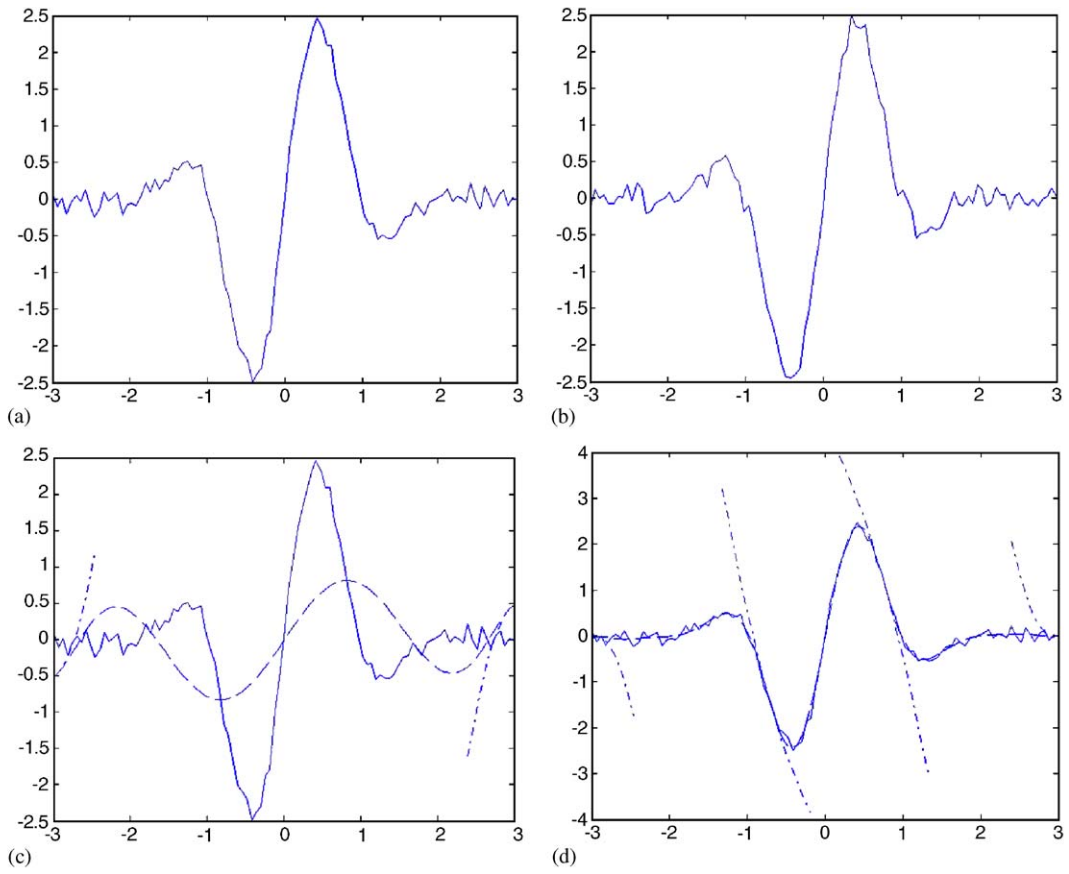

Fig. 4. (a) Training set with 100 data points for the function $y(x)$ with additive Gaussian noise $\mathcal{N}(0, 0.1)$. (b) Validation set with 100 data points for the function $y(x)$ with additive Gaussian noise $\mathcal{N}(0, 0.1)$. (c) Function approximation using two rules (dashed line). The consequent polynomials of the two rules are also shown (dash-dotted lines). (d) Function approximation with four rules. See how only with four rules the objective function has been approximated (dashed line). See also how the form of consequent polynomials is similar to the output function in the points near to the MF centres $= \{-3, -0.71, 0.71, 3\}$ (dash-dotted lines).

$$\text{IF } x \text{ IS } 0.71 \text{ THEN } y = -2.24(x-0.71)^2 - 5.96(x-0.71) + 1.30,$$

$$\text{IF } x \text{ IS } 3 \text{ THEN } y = 7.22(x-3)^2 + 0.91(x-3) + 0.01. \tag{28}$$

Fig. 4 shows the training (a) and validation (b) data sets used for the presented example. It is also shown in (c) a snapshot state of the algorithm after the two rules structure has been analysed by the algorithm. It contains the original data set, the output obtained with these two rules and how both rule consequent polynomials resemble the output of the system at the vicinity of the rule centres. In (d) the final TaSe model output is represented with the original data set. The final model keeps the noise completely filtered. Note also for the four rules centred in $\{-3, -0.71, 0.71, 3\}$, how the representation of the rule consequent polynomial is practically identical to the model output at the very vicinity of the rule centres. The automatic algorithm presented in this paper achieves a model that has accuracy in the approximation, and that is fully interpretable thanks to the Taylor series concept.

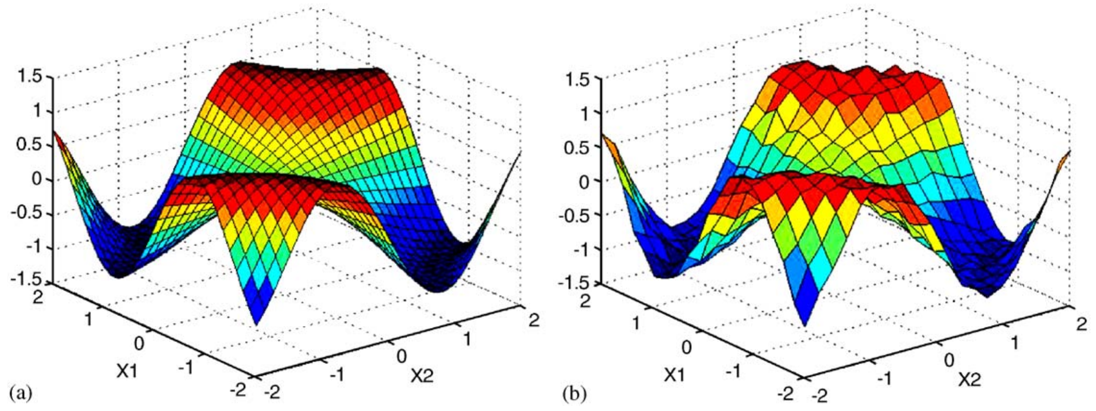

Fig. 5. (a) Function f_1 with 1000 equidistributed data points and (b) function f_1 with Gaussian noise $\mathcal{N}(0, 0.05)$ and 400 randomly distributed data points.

Adding one more MF leaded to an increase in the validation error, therefore the optimal model obtained is the previous one with 4 rules. The average error obtained from 10 different executions equals to 0.105, with standard deviation 0.012. Notice also the remarkable low number of rules needed (see Section 2) to approximate the function, defined by the training and validation data sets. Making a short comparison, the number of rules needed to approximate this function using constant coefficients for example is much higher as noted in [29].

6.2. Application to a 2-D function

Now consider the 2-D function f_1 taken from [7]. The whole algorithm will be now executed, including the structure identification sub-algorithm. This function f_1 (see Fig. 5) can be expressed as

$$f_1(x_1, x_2) = \sin(x_1 \cdot x_2) + \xi \quad \text{where } x_1,\ x_2 \text{ uniform in } [-2, 2];\ \xi \text{ noise}. \tag{29}$$

As noted in the previous example, the algorithm begins with a very simple configuration (1 MF per input variable = 1 single rule) and works considering more complex configurations, while obtaining the optimal parameters for each one of them. The decision as to which variable should contain a new MF was discussed in Section 5. The example considered now is an intuitive example of how such a decision is reached. It is apparent from Eq. (29) that both input variables influence the final value of the function equally. Therefore the algorithm presented must take this into account and endow each input variable with a similar number of MFs, thus improving the degree of approximation while maintaining the lowest possible level of system complexity.

From this function we will extract 400 training data points and 400 validation data points, both randomly distributed and which will be corrupted with $\mathcal{N}(0, 0.05)$ additive Gaussian noise. As we said before, the initial system configuration is 1 MF per input variable resulting in 1 single rule. Least squares provides the rule consequent coefficients, thus the rule centred in $\{x_1 = 0,\ x_2 = 0\}$ can be expressed as

IF x_1 IS 0 AND x_2 IS 0 THEN

$$\begin{aligned} y = &-0.0(x_1-0)^2 + 0.27(x_1-0)(x_2-0) - 0.0(x_2-0)^2 \\ &+ 0.54(x_1-0) + 0.54(x_2-0) + 1.09 \end{aligned} \tag{30}$$

Table 1
Trace of the structure identification algorithm for the example f_1

#MFs		*NRMSE* (5% noise)		Adding 1 MF in variable $X1$	Adding 1 MF in variable $X2$
$X1$	$X2$	Training	Validation		
1	1	0.776	0.772	0.562	0.563
2	1	0.560	0.562	0.561	0.113
2	2	0.108	0.113	0.0954	0.0975
2	3	0.0933	0.0954	0.0844	0.0820
3	3	0.0731	**0.0820**	0.0830	0.0844
3	4	Error increases adding 1 MF in any variable			

with error $NRMSE = 0.774$. Now a decision should be made as to in which variable should we add one more MF so that the error is lower. As explained in Section 5, the two alternatives will be tested, i.e., adding one MF in the first variable and adding one MF in the second one. The alternative that provides lower error will be chosen, so that the MF is permanently added to the system configuration for further iterations of the structure identification algorithm. The MF configuration 2×1 gives an error $NRMSE = 0.562$, and the configuration 1×2 provides an error $NRMSE = 0.563$. Notice the similarity in performance provided by both structures. As mentioned above, the function f_1 we are considering is symmetric, therefore this situation is expectable. According to the errors obtained by both structures tested, one MF will be added permanently in the first variable. The two rules, centred in $\{x_1 = -2, x_2 = 0\}$, and $\{x_1 = 2, x_2 = 0\}$ of the new eventual system are

$$\text{IF } x_1 \text{ IS } -2 \text{ AND } x_2 \text{ IS } 0 \text{ THEN}$$

$$\begin{aligned} y = {} & 0.05(x_1+2)^2 - 0.97(x_1+2)(x_2-0) - 0.0(x_2-0)^2 \\ & - 1.93(x_1+2) - 0.07(x_2-0) - 0.14 \end{aligned}$$

$$\text{IF } x_1 \text{ IS } +2 \text{ AND } x_2 \text{ IS } 0 \text{ THEN}$$

$$\begin{aligned} y = {} & -0.04(x_1-2)^2 - 0.95(x_1-2)(x_2-0) + 0.0(x_2-0)^2 \\ & - 1.94(x_1-2) + 0.05(x_2-0) + 0.11. \end{aligned} \tag{31}$$

In the next step, the structure identification sub-algorithm will check the performance of the two possible alternatives, i.e., MF structure 3×1 and MF structure 2×2. The first alternative gives an error $NRMSE = 0.561$, and the second one gives an error $NRMSE = 0.113$. See here the great difference between adding one MF in one or another input variable. This is caused again by the symmetry of the function f_1 considered, that will lead to an equal distribution of the MFs among the input variables until the validation error stops decreasing. The complete execution of the structure identification algorithm can be inspected in Table 1. It also shows for each algorithm stage, the error provided by the chosen configuration, as well as the validation errors obtained by the two possible alternatives that can take place in each step, so that the reader can easily check the correctness of the decisions made by the algorithm.

The optimal structure found for our model has 3 MFs per input variable. The *NRMSE* obtained is 0.0820, and the 9 rules obtained by the whole algorithm execution are

IF x_1 IS − 2 AND x_2 IS − 2 THEN

$$y = -0.24(x_1+2)^2 - 0.34(x_1+2)(x_2+2) - 1.08(x_2+2)^2 + 2.12(x_1+2) + 1.81(x_2+2) - 0.83$$

IF x_1 IS − 2 AND x_2 IS − 0.06 THEN

$$y = -0.24(x_1+2)^2 + 1.12(x_1+2)(x_2+0.06) - 0.05(x_2+0.06)^2 + 0.05(x_1+2) - 1.97(x_2+0.06) - 0.10$$

IF x_1 IS − 2 AND x_2 IS 2 THEN

$$y = 0.74(x_1+2)^2 - 0.23(x_1+2)(x_2-2) + 1.05(x_2-2)^2 - 1.98(x_1+2) + 1.77(x_2-2) + 0.82$$

IF x_1 IS + 0.13 AND x_2 IS − 2 THEN

$$*y = -1.05(x_1-0.13)^2 + 0.92(x_1-0.13)(x_2+2) + 0.14(x_2+2)^2 - 2.05(x_1-0.13) - 0.03(x_2+2) - 0.22$$

IF x_1 IS + 0.13 AND x_2 IS − 0.06 THEN

$$y = 0.10(x_1-0.13)^2 + 1.10(x_1-0.13)(x_2+0.06) + 0.11(x_2+0.06)^2 - 0.12(x_1-0.13) + 0.18(x_2+0.06) - 0.01$$

IF x_1 IS + 0.13 AND x_2 IS + 2 THEN

$$y = 0.46(x_1-0.13)^2 + 1.01(x_1-0.13)(x_2-2) - 0.29(x_2-2)^2 + 2.10(x_1-0.13) + 0.02(x_2-2) + 0.26$$

IF x_1 IS + 2 AND x_2 IS − 2 THEN

$$y = 2.64(x_1-2)^2 - 0.03(x_1-2)(x_2+2) + 1.94(x_2+2)^2 + 1.88(x_1-2) - 1.82(x_2+2) + 0.79$$

IF x_1 IS + 2 AND x_2 IS − 0.06 THEN

$$y = 0.19(x_1-2)^2 + 0.99(x_1-2)(x_2+0.06) - 0.63(x_2+0.06)^2 + 0.15(x_1-2) + 2.04(x_2+0.06) - 0.09$$

$$\text{IF } x_1 \text{ IS} + 2 \text{ AND } x_2 \text{ IS} + 2 \text{ THEN}$$

$$\begin{aligned} y = &-2.22(x_1-2)^2 - 0.08(x_1-2)(x_2-2) - 0.49(x_2-2)^2 \\ &- 1.90(x_1-2) - 1.81(x_2-2) - 0.78. \end{aligned} \tag{32}$$

See also in Table 1 how adding one more MF to any input variable leads to an increase in the validation error, thus forcing the algorithm to stop and returning the MF configuration 3×3 with rule centres shown in Eq. (32). The average error obtained from 10 different executions equals to 0.085, with standard deviation 0.005.

Fig. 6 shows the output obtained by our model as well as the rule consequent polynomial representations for the rules of the intermediate MF configurations 2×2 and 3×3. See how the output of the model resembles, already with the 2×2 configuration, the original f_1 function. The final configuration 3×3 provides a noiseless output very similar to the original f_1 function ($NRMSE = 0.0820$). See also in (b) and (e), the outputs provided by each rule polynomial for both configurations 2×2 and 3×3. Note the high similarity in gradient between the polynomials output (b) and (e), and the model output at the vicinity of the rule centres. A more accurate comparison can be checked in (d) and (f). Thus we have obtained an accurate and fully interpretable model for the approximation of the function f_1, provided we are given the training and validation data sets.

From this example it must be also noticed the remarkable low number of interpretable rules needed to accurately approximate the objective function f_1. See that only with 9 TaSe rules, the error obtained for the noisy example is $NRMSE = 0.0820$. Similar approaches with constant or linear rule consequents need a much higher number of rules to obtain similar results [29].

6.3. Application to the Mackey–Glass time series

The Mackey–Glass chaotic time series [21] is a very well-known benchmark for system modelling, that has been widely used in the literature. This time series is described by the following delay differential equation

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \frac{ax(t-\tau)}{1 + x^{10}(t-\tau)} - bx(t). \tag{33}$$

One thousand data points were generated with an initial condition $x(0) = 1.2$ and $\tau = 17$ using the fourth-order Runge–Kutta method [15]. To make the comparisons with earlier works fair, we chose the parameters so that the training vectors for the model have the following format:

$$[x(t-18), x(t-12), x(t-6), x(t); x(t+6)]. \tag{34}$$

In Fig. 7 we show the section of 1000 data points used in this study, selecting the first 500 for the training and validation data sets (randomly choosing 400 and 100, respectively, of the 500), and the final 400 for test.

Table 2 shows the evolution of the structure identification algorithm, that starts with an initial MF configuration of 1 MF per variable and iterates adding 1 MF to the selected variable according to the algorithm presented in Section 5.

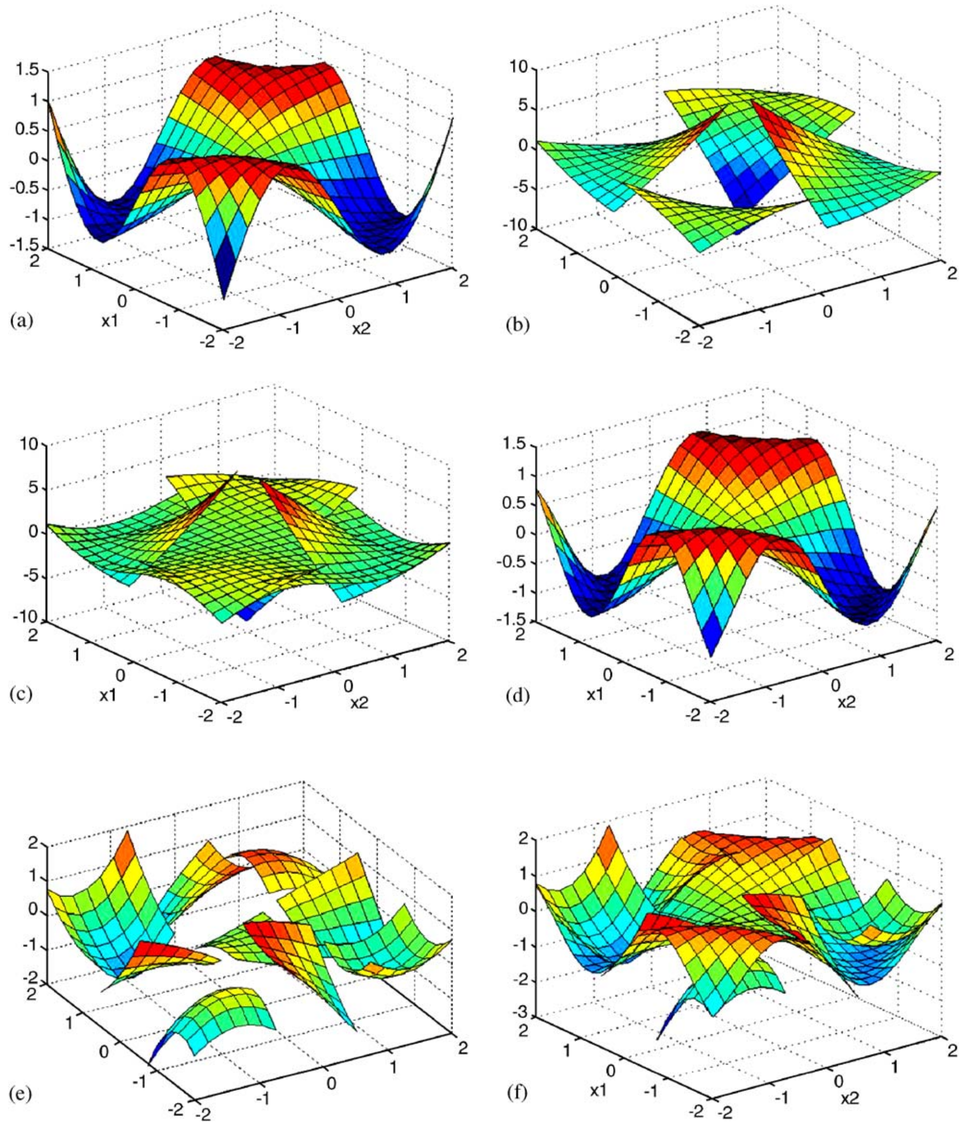

Fig. 6. (a) Output of the model with 2MFs per variable, i.e., with 4 rules. (b) Rule consequent polynomials of the 4 rules in the model with 2 × 2 MF distribution. (c) Combined representation. (d) Output of the model with 3MFs per variable, i.e., with 9 rules. (e) Rule consequent polynomials of the 9 rules in the model with 3 × 3 MF distribution and (f) Combined representation.

Finally, Table 3 compares the prediction accuracy of different computational paradigms presented in the bibliography for this benchmark problem. In order to perform an equal comparison (also in complexity), we show our test error for three MF configurations: $3 \times 2 \times 1 \times 1$ with 91 ($15 \times 6 + 1$ mobile centre) parameters and $3 \times 2 \times 1 \times 2$ with 181 ($15 \times 12 + 1$ mobile centre) parameters and $3 \times 3 \times 1 \times 3$ with 408 ($15 \times 27 + 3$ mobile centres) (see Eq. (10)). The results drawn from this example show the convenience of the TaSe model, not only due to its interpretability and transparency properties and the

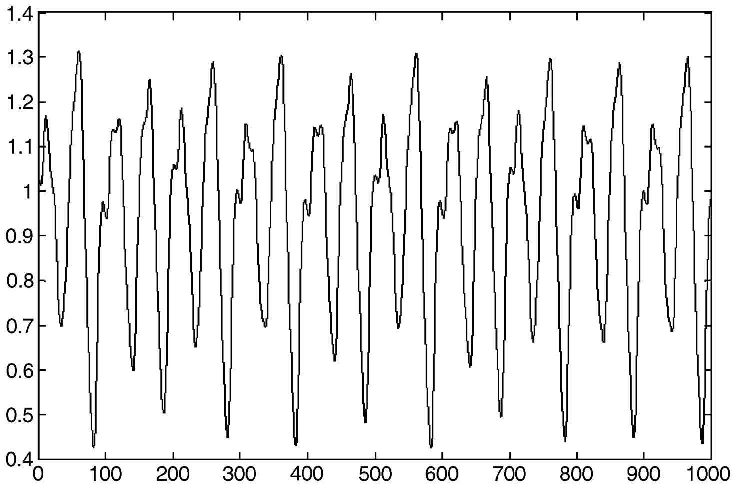

Fig. 7. Mackey–Glass chaotic time series with $\tau = 17$.

Table 2
Trace of the structure identification algorithm for the Mackey–Glass time-series problem

#MFs				*RMSE* (for predicting $x(t+6)$)			Adding	Adding	Adding	Adding
$x(t-18)$	$x(t-12)$	$x(t-6)$	$x(t)$	Training	Validation	Test	1 MF in variable $x(t-18)$	1 MF in variable $x(t-12)$	1 MF in variable $x(t-6)$	1 MF in variable $x(t)$
1	1	1	1	0.032	0.029	0.034	0.013	0.0080	0.016	0.021
1	2	1	1	0.0084	0.008	0.0098	0.0037	0.0069	0.0056	0.0046
2	2	1	1	0.0032	0.0037	0.0035	0.0027	0.0031	0.0034	0.0033
3	2	1	1	0.0024	0.0027	0.0028	0.0025	0.0025	0.0027	0.0024
3	2	1	2	0.0020	0.0024	0.0024	0.0022	0.0020	0.0025	0.0021
3	3	1	2	0.0014	0.0020	0.0017	0.0019	0.0021	0.0022	0.0018
3	3	1	3	0.0011	0.0018	0.0013	Error increases adding 1 MF to any variable			

interaction between the local and global models, but also due to its high degree of accuracy for function approximation and time series prediction problems. Note that with only 6 rules, the TaSe model is able to identify the non-linear behaviour of the Mackey–Glass time series better than many other time series prediction methods proposed in the literature (with a similar model complexity), being the consequents of these rules the Taylor series expansion of the output of the model centred in their respective rule centres.

7. Conclusions

In order to avoid the lack of interpretability in Takagi–Sugeno–Kang (TSK) fuzzy rules, in this paper we have presented a powerful tool for function approximation problems using a novel fuzzy system with a specific type of rule antecedents and Taylor Series based rule consequents (TaSe Fuzzy systems).

Table 3
Comparison results of the prediction error of different methods for prediction step equal to 6 (500 training data)

Method		Test *RMSE*
Auto regressive model		0.19
Cascade correlation NN		0.06
Back-prop. NN		0.02
Sixth-order polynomial		0.04
Linear predictive method		0.55
Kim and Kim (genetic algorithm and fuzzy system) [20]	5 MFs	0.0492
	7 MFs	0.0422
	9 MFs	0.037
ANFIS and fuzzy system (16 rules) [17]		0.007
Classical RBF (with 23 Neurons) [8]		0.0114
PG-RBF [30]		0.0030
TaSe fuzzy system with 6 rules		0.0028
TaSe fuzzy system with 12 rules		0.0024
Optimal TaSe fuzzy system with 27 rules		0.0013

This system is thus endowed with both the approximating capabilities of TSK fuzzy rules, and the interpretability advantages of traditional fuzzy systems, since every rule consequent can be regarded as the Taylor series expansion at rule centres. We have also provided a complete and automatic methodology for both parameter estimation and structure identification of the TaSe FS. Finally the goodness and the suitability of the algorithm have been tested in detail through two complete examples for function approximation and the well-known benchmark Mackey–Glass time series.

Acknowledgements

This work has been partially supported by the Spanish CICYT Project TIN2004-01419.

Appendix

Now, we present the partial derivatives of the error function given in (9) with respect to each rule consequent coefficient. Using the expressions for all the rule coefficients, we will obtain a set of linear equation system that will provide the optimal rule consequents given the specific set of data points *D*, as explained in Section 2. The generic expression for any zero-order (w_0), first-order ($w_{v_1}, \ldots, w_{v_1}$) or second order coefficient ($w_{v_{11}}, \ldots, w_{v_{nn}}$) is

$$\frac{\partial J}{\partial w_s^j} = 2 \sum_{m \in D} \left(y^m - \frac{\sum_{k=1}^K \mu_k\left(\vec{x}^m\right)\left(w_0^k + \vec{w}_1^k \cdot \vec{x}^m + \frac{1}{2}\left(\vec{x}^m\right)^{\mathrm{T}} W_2^k \vec{x}^m\right)}{\sum_{k=1}^K \mu_k\left(\vec{x}^m\right)} \right) \frac{\mu_j\left(\vec{x}^m\right) \cdot f_{w_s^j}\left(\vec{x}^m\right)}{\sum_{k=1}^K \mu_k\left(\vec{x}^m\right)},$$

$$s = \{0, v_1, \ldots, v_n, v_{11}, \ldots, v_{nn}\}, k = 1 \ldots K. \quad (35)$$

Being $f_{w_s^j}(\vec{x}^m)$ equal to 1 for $s = 0$, $x_{v_l}^m$ for $s = v_l$ and $x_{v_l}^m \cdot x_{v_h}^m$ for $s = v_{lh}$. Now the partial derivatives lead to the following three types of equations:

$$\begin{aligned}\text{for } s = 0, \quad \frac{\partial J}{\partial w_s^j} = 0 \Rightarrow &\sum_{k=1}^{K} \sum_{i} w_i^k \sum_{m \in D} \frac{\mu_k(\vec{x}^m) \cdot f_{w_i^k}(\vec{x}^m) \cdot \mu_j(\vec{x}^m)}{\sum_{j=1}^{K} \mu_j(\vec{x}^m)} \\ &= \sum_{m \in D} \frac{y^m \cdot \mu_j(\vec{x}^m)}{\sum_{k=1}^{K} \mu_k(\vec{x}^m)}, \end{aligned} \tag{36}$$

$$\begin{aligned}\text{for } s = v_l, \quad \frac{\partial J}{\partial w_s^j} = 0 \Rightarrow &\sum_{k=1}^{K} \sum_{i} w_i^k \sum_{m \in D} \frac{\mu_k(\vec{x}^m) \cdot f_{w_i^k}(\vec{x}^m) \cdot \mu_j(\vec{x}^m) \cdot x_{v_l}^m}{\sum_{j=1}^{K} \mu_j(\vec{x}^m)} \\ &= \sum_{m \in D} \frac{y^m \cdot \mu_j(\vec{x}^m) \cdot x_{v_l}^m}{\sum_{k=1}^{K} \mu_k(\vec{x}^m)}, \end{aligned} \tag{37}$$

$$\begin{aligned}\text{for } s = v_{lh}, \quad \frac{\partial J}{\partial w_s^j} = 0 \Rightarrow &\sum_{k=1}^{K} \sum_{i} w_i^k \sum_{m \in D} \frac{\mu_k(\vec{x}^m) \cdot f_{w_i^k}(\vec{x}^m) \cdot \mu_j(\vec{x}^m) \cdot x_{v_l}^m \cdot x_{v_h}^m}{\sum_{j=1}^{K} \mu_j(\vec{x}^m)} \\ &= \sum_{m \in D} \frac{y^m \cdot \mu_j(\vec{x}^m) \cdot x_{v_l}^m \cdot x_{v_h}^m}{\sum_{k=1}^{K} \mu_k(\vec{x}^m)}. \end{aligned} \tag{38}$$

For the optimisation of the position of the MF centres, gradient descent-based methods can be applied in order to find a local optimal position given a starting MF centres configuration. Then it is necessary to calculate the variation of the function J, as given in Eq. (9), with respect to the position of each MF centre. For this purpose we must find

$$\frac{\partial J}{\partial c_p^{i_p}} = -2 \sum_{m \in D} \left[\left(y^m - F\left(\vec{x}^m\right) \right) \left(\frac{\partial F(\vec{x}^m)}{\partial c_p^{i_p}} \right) \right]. \tag{39}$$

From (18) we have

$$\begin{aligned}\frac{\partial F(\vec{x}^m)}{\partial c_p^{i_p}} &= \frac{\partial}{\partial c_p^{i_p}} \left(\frac{\sum_{k=1}^{K} \mu_k(\vec{x}^m)\, Y_k(\vec{x}^m)}{\sum_{k=1}^{K} \mu_k(\vec{x}^m)} \right) = \frac{\partial}{\partial c_p^{i_p}} \left(\sum_{k=1}^{K} \mu_k\left(\vec{x}^m\right) Y_k\left(\vec{x}^m\right) \right) \\ &= \sum_{k=1}^{K} Y_k\left(\vec{x}^m\right) \left(\frac{\partial \mu_k(\vec{x}^m)}{\partial c_p^{i_p}} \right). \end{aligned} \tag{40}$$

Now the partial derivative for each rule firing strength function, since only the MFs of variable p can depend on centre $c_p^{i_p}$, stays as

$$\frac{\partial \mu_k(\vec{x}^m)}{\partial c_p^{i_p}} = \frac{\partial \prod_{j=1}^{n} \mu_k^j\left(x_j^m\right)}{\partial c_p^{i_p}} = \frac{\partial \mu_k^p\left(x_p^m\right)}{\partial c_p^{i_p}} \prod_{\substack{j=1 \\ j \neq p}}^{n} \mu_k^j\left(x_j^m\right). \tag{41}$$

Finally, from (16) and (17), after some algebra it may be seen that

$$\frac{\partial \mu_k^p\left(x_p^m\right)}{\partial c_p^{i_p}} = \begin{cases} \dfrac{30 \cdot (x-a)^2 \cdot (x-b)^3}{(b-a)^6} & \text{if } x_p^m \in [a,b] \text{ and } c_p^{i_p} = a \text{ in } MF_p^{i_p+1}(x,[a,b,c]), \\ -\dfrac{30 \cdot (x-a)^3 \cdot (x-b)^2}{(b-a)^6} & \text{if } x_p^m \in [a,b] \text{ and } c_p^{i_p} = b \text{ in } MF_p^{i_p}(x,[a,b,c]), \\ \dfrac{30 \cdot (x-b)^3 \cdot (x-c)^2}{(c-b)^6} & \text{if } x_p^m \in [b,c] \text{ and } c_p^{i_p} = b \text{ in } MF_p^{i_p}(x,[a,b,c]), \\ -\dfrac{30 \cdot (x-b)^2 \cdot (x-b)^3}{(c-b)^6} & \text{if } x_p^m \in [b,c] \text{ and } c_p^{i_p} = c \text{ in } MF_p^{i_p-1}(x,[a,b,c]). \end{cases} \tag{42}$$

It must be finally noted that the centres found at the extremes of each variable are considered to be fixed as their positions depend exclusively on the minimum and maximum values of the range of each variable.

References

[1] P. Angelov, D. Filev, An approach to online identification of Takagi–Sugeno fuzzy models, IEEE Trans. Systems Man Cybernet. 34 (1) (2004) 484–498.

[2] R. Bellman, Adaptive Control Processes: A Guided Tour, Princeton University Press, Princeton, NJ, 1961.

[3] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, New York, 1981.

[4] M. Bikdash, A highly interpretable form of Sugeno inference systems, IEEE Trans. Fuzzy Systems 7 (6) (1999) 686–696.

[5] W. Brockman, O. Huwendiek, Neuro-fuzzy approach for modeling complex functional mappings, Proc. 2000 IEEE Internat. Conf. on Systems Man and Cybernetics—-SMC2000, Omnipress, Madison, 2000, 3734–3739.

[6] J.J. Buckley, Sugeno-type controllers are universal controllers, Fuzzy Sets and Systems 25 (1993) 299–303.

[7] V. Cherkassky, D. Gehring, F. Mulier, Comparison of adaptive methods for function estimation from samples, IEEE Trans. Neural Networks 7 (4) (1996) 969–984.

[8] K.B. Cho, B.H. Wang, Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction, Fuzzy Sets and Systems 83 (1995) 325–339.

[9] F.L. Chung, On multistage fuzzy neural network modeling, IEEE Trans. Fuzzy 8 (2) (2000) 125–142.

[10] F.J. de Souza, M.M.R. Vellasco, M.A.C. Pacheco, Hierarchical neuro-fuzzy quadtree models, Fuzzy Sets and Systems 130 (2) (2002) 189–205.

[11] D. Driankov, H. Hellendoorn, M. Reinfrank, An Introduction to Fuzzy Control, Springer, Berlin, 1996.

[12] G. Golub, C.V. Loan, Matrix Computations, The Johns Hopkins University Press, Baltimore, 1996.

[13] J. Gonzalez, I. Rojas, H. Pomares, J. Ortega, A. Prieto, A new clustering technique for function approximation, IEEE Trans. Neural Network 13 (1) (2002).

[14] S. Guillaume, Designing fuzzy inference systems from data: an interpretability-oriented review, IEEE Trans. Fuzzy Systems 9 (2001) 426–443.

[15] http://neural.cs.nthu.edu.tw/jang/benchmark/

[17] J.S.R. Jang, C.T. Sun, E. Mizutani, Neuro-Fuzzy and Soft Computing, Prentice-Hall, Englewood Cliffs, NJ, 1997.

[18] T.A. Johansen, R. Babuska, Multiobjective identification of Takagi–Sugeno fuzzy models, IEEE Trans. Fuzzy Systems 11 (6) (2003) 847–860.

[19] D. Kim, C. Kim, Forecasting time series with genetic fuzzy predictor ensemble, IEEE Trans. Fuzzy Systems 5 (4) (1997) 523–535.

[20] D. Kim, C. Kim, Forecasting time series with genetic fuzzy predictor ensemble, IEEE Trans. Fuzzy Systems 5 (4) (1997) 523–535.

[21] M.C. Mackey, L. Glass, Oscillation and chaos in physcological control systems, Science 197 (1977) 287–289.

[22] E.H. Mamdani, S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, Internat. J. Man-Mach. Stud. 7 (1) (1975) 1–12.

[23] J.J. More, The Levenberg–Marquardt algorithm implementation and theory, Lect. Notes Math. 630 (1978) 105–116.
[24] J.H. Nie, T.H. Lee, Rule-based modelling: fast construction and optimal manipulation, IEEE Trans. Systems Man Cybernet. 26 (6) Part A (1996) 728–738.
[25] Z. Ning, Y.S. Ong, K.W. Wong, K.T. Seow, Parameter identification using memetic algorithms for fuzzy systems, 4th Internat. Conf. on Intelligent Technologies, December 17–19, 2003, Thailand, Chiang Mai.
[26] R.P. Paiva, A. Dourado, Structure and parameter learning of neuro-fuzzy systems: a methodology and a comparative study, J. Intell. Fuzzy Systems 11 (2001) 147–161.
[27] R.P. Paiva, A. Dourado, Interpretability and learning in neuro-fuzzy systems, Fuzzy Sets and Systems 147 (2004) 17–38.
[28] H. Pomares, I. Rojas, J. González, A. Prieto, Structure identification in complete rule-based fuzzy systems, IEEE Trans. Fuzzy 10 (3) (2002) 349–359.
[29] H. Pomares, I. Rojas, J. Ortega, J. Gonzalez, A. Prieto, A systematic approach to a self-generating fuzzy rule-table for function approximation, IEEE Trans. Systems Man Cybernet. 30 (3) (2000) 431–447.
[30] I. Rojas, H. Pomares, J.L. Bernier, J. Ortega, B. Pino, F.J. Pelayo, A. Prieto, Time series analysis using normalized PG-RBF network with regression weights, NeuroComputing 42 (2002) 267–285.
[31] I. Rojas, H. Pomares, J. Ortega, A. Prieto, Self-organized fuzzy system generation from training examples, IEEE Trans. Fuzzy Systems 8 (1) (2000) 23–36.
[32] E.H. Ruspini, A new approach to Clustering, Information and Control, 15 (1969) 22–32.
[33] M. Setnes, R. Babuska, U. Kaymak, H.R. van Nauta Lemke, Similarity measures in fuzzy rule base simplification, IEEE Trans. Systems Man Cybernet. 28 (3) (1998) 376–386.
[34] M. Sugeno, G.T. Kang, Structure identification of fuzzy model, Fuzzy Sets and Systems 28 (1) (1988) 15–33.
[35] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modelling and control, IEEE Trans. Systems Man Cybernet. 15 (1985) 116–132.
[36] L.X. Wang, Adaptive Fuzzy Systems and Control, Design and Stability Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1994.
[37] L.X. Wang, J.M. Mendel, Generating fuzzy rules by learning from examples, IEEE Trans. Systems Man Cybernet. 22 (6) (1992) 1414–1427.
[38] B. Wu, X. Yu, Fuzzy modelling and identification with genetic algorithm based learning, Fuzzy Sets and Systems 113 (3) (2000) 351–365.
[39] J. Yen, L. Wang, C.W. Gillespie, Improving the interpretability of TSK fuzzy models by combining global learning and local learning, IEEE Trans. Fuzzy Systems 6 (4) (1998) 530–537.
[40] L.A. Zadeh, Soft computing and fuzzy logic, IEEE Software (1994) 48–56.
[41] S.M. Zhou, J.Q. Gan, Improving the interpretability of Takagi–Sugeno fuzzy model by using linguistic modifiers and a multiple objective learning scheme, Internat. Joint Conf. on Neural Networks IJCNN2004, Budapest, 2004.