



Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE @ DIRECT®
Fuzzy Sets and Systems III (IIII) III-III

FUZZY
sets and systems

www.elsevier.com/locate/fss

Interpretability and learning in neuro-fuzzy systems

Rui Pedro Paiva, António Dourado*

*CISUC-Centro de Informática e Sistemas da Universidade de Coimbra, Departamento de Engenharia Informática,
Faculdade de Ciências e Tecnologia, PÓLO II da Universidade de Coimbra, Pinhal de Marrocos,
Coimbra P 3030, Portugal*

Abstract

A methodology for the development of linguistically interpretable fuzzy models from data is presented. The implementation of the model is conducted through the training of a neuro-fuzzy network, i.e., a neural net architecture capable of representing a fuzzy system. In the first phase, the structure of the model is obtained by means of subtractive clustering, which allows the extraction of a set of relevant rules based on a set of representative input-output data samples. In the second phase, the parameters of the model are tuned via the training of a neural network through backpropagation. In order to attain interpretability goals, the method proposed imposes some constraints on the tuning of the parameters and performs membership function merging. In this way, it will be easy to assign linguistic labels to each of the membership functions obtained, after training. Therefore, the model obtained for the system under analysis will be described by a set of linguistic rules, easily interpretable.

© 2003 Published by Elsevier B.V.

Keywords: System identification; Fuzzy system models; Neuro-fuzzy systems; Clustering; Interpretability; Transparency

1. Introduction

Extracting knowledge from data is a very interesting and important task in information science and technology. Application fields are not only complex industrial production and medical systems but also electronic commerce and leisure activities.

Several studies have already been conducted in terms of the development of modeling and control algorithms for industrial systems based on the so-called intelligent techniques, as a means of integrating “intelligence” into production systems. Fundamentally, such developments aim to overcome some of the limitations and difficulties associated with classical methodologies.

* Corresponding author. Tel.: +351-39-790000; fax: +351-39-701266.

E-mail address: dourado@dei.uc.pt (A. Dourado).

URL: <http://www.dei.uc.pt/~dourado>

1 In this context, more precisely in system modeling, it is sometimes necessary that the resulting
2 models have some transparency, i.e., that their information be interpretable, so as to permit a deeper
3 understanding of the system under study. It is in this issue that fuzzy modeling finds its maximum
4 potential. In fact, fuzzy models have some properties that make them particularly interesting, namely,
5 possibility of linguistic interpretation [29] and universal approximation [4], being the former hardly
6 attained via multi-layer perceptrons (MLPs). However, they have an important limitation, which
7 results from the difficulty to quantify the fuzzy linguistic terms. Therefore, neuro-fuzzy nets appear as
8 an attempt to combine the advantages of fuzzy systems in terms of transparency with the advantages
9 of neural networks regarding learning capabilities.

10 Transparency is a measure of the human linguistic interpretability of the rules issued from the
11 training of the neuro-fuzzy system. In many engineering applications it is a very important property,
12 since it allows the transformation of data (information) into (human) knowledge. This problem
13 has recently been faced for classification problems by Shanahan [25], Fuessel and Isermann [9],
14 Binaghi [2], Sanchez [23], Ishibuchi [12]; in the prediction of process behavior by Maier [18]; in
15 decision making and data mining by Gorzalczany and Piasta [10,15]; in theoretical developments
16 on fuzzy systems by Klement [14]; in intelligent control and robotics by Stoica [26]; in function
17 approximation, by Nauck and Kruse [19]. Jin [13] addresses interpretability by using similarity
18 measures to check the similarity of each rule; the structure and parameters of the fuzzy rules are
19 optimized and interpretability is improved by fine-tuning the fuzzy rules with regularization. Espinosa
20 and Vandewalle [8] propose a methodology to extract rules from data within the framework of
21 linguistic integrity, to guarantee their interpretability in the linguistic context; their approach allows
22 for the inclusion of prior knowledge in the rule base. Abonyi [1] develops neuro-fuzzy systems
23 for Takagi–Sugeno type, maintaining the readability and interpretability of the fuzzy model during
24 and after learning; they use an on-line gradient-descent-based learning algorithm, in an internal
25 model control structure. Roubos and Setnes [22] propose fuzzy clustering with rule simplification by
26 constrained genetic optimization with low-human intervention. For a more detailed review see [11].

27 The methodology presented in this work is carried out in two main stages: in the first one, structure
28 learning is performed, i.e., a set of fuzzy rules is obtained; in the second one, the parameters of the
29 model are tuned, i.e., the parameters of the membership functions of the fuzzy system.

30 The generality of strategies developed by several authors, based on the referred scheme, aim
31 fundamentally at obtaining models with high prediction accuracy. Clearly, if this is the main goal,
32 it is questionable whether fuzzy modeling is the most adequate technique, as pointed out in [19].
33 However, in case model transparency is a fundamental goal, the strategy referred gives no guarantees
34 regarding that objective. In fact, since parameter tuning is carried out under no constraints, highly
35 complex fuzzy databases may come up.

36 Therefore, this paper explores the potential of neuro-fuzzy networks in helping in the construction
37 of real transparent models. Thus, linguistic models, i.e., models whose consequents are fuzzy sets, are
38 used instead of Takagi–Sugeno models [28], where the consequents implement, typically, first-order
39 linear functions, difficult to interpret linguistically. Additionally, parameter learning is constrained
40 and similar membership functions are merged, in order to ease the attribution of linguistic labels to
41 the final functions.

42 The paper is organized as follows. In Section 2, the main issues of fuzzy identification are intro-
43 duced. In Section 3, subtractive clustering, used for structure learning is presented. The unconstrained
parameter learning strategy is described in Section 4. In Section 5, the strategies for implementa-

1 tion of interpretable models are presented. The methodologies are applied, in Section 6, to the
 2 Box–Jenkins gas furnace and to the Mackey–Glass chaotic time series, two well-known benchmark
 3 problems in system modeling and identification. Finally, some conclusions are drawn in Section 7.

2. Fuzzy modeling and identification

5 Dynamical system identification deals with the implementation of models using experimental data.
 6 Thus, when a model is developed based on the theory of system identification, its parameters are
 7 tuned according to some criteria, aiming to obtain a final representation, adequate for the modeling
 8 purposes. In this sense, fuzzy identification is presented as a particular case of system identification,
 9 in which the model is classified as a fuzzy system.

10 Thus, without loss of generality, let us assume a single-input single-output (SISO) model, with
 11 one input, u , and one output, y , from where N data samples are collected (1):

$$Z^N = \{[u(1), y(1)], [u(2), y(2)], \dots, [u(N), y(N)]\}. \quad (1)$$

13 Using data collected from the system, the goal is to obtain a fuzzy model, represented by a set
 of rules of type R_i (2):

$$\begin{aligned} R_i: & \text{ If } y(t-1) \text{ is } A_{1i} \text{ and } y(t-2) \text{ is } A_{2i} \text{ and } \dots \\ & \text{ and } u(t-d) \text{ is } B_{1i} \text{ and } u(t-d-1) \text{ is } B_{2i} \text{ and } \dots \\ & \text{ then } \hat{y}(t) \text{ is } C_{1i}, \end{aligned} \quad (2)$$

15 where d represents the system delay time and A_{ji} , B_{ji} , and C_{ji} denote the linguistic terms associated
 to each input and output. Those terms are defined by their respective membership functions $\mu_{A_{ji}}$,
 17 $\mu_{B_{ji}}$, $\mu_{C_{ji}}$. The former structure is called an *FARX* structure (Fuzzy Auto Regressive with eXogenous
 inputs), as a generalization of the well-known ARX structure. Thus, the selection of a set of rules
 19 of type (2), as well as the definition of the fuzzy sets A_{ji} , B_{ji} , and C_{ji} , are project issues specific to
 fuzzy systems.

21 3. Structure learning

22 In order to obtain a set of g fuzzy conditional rules capable of representing the system under
 23 study, clustering algorithms are particularly suited, since they permit a scatter partitioning of the
 input–output data space, which results in finding only the relevant rules. Comparing to gridbased
 25 partitioning methods, clustering algorithms have the advantage of avoiding the explosion of the rule
 base, a problem known as the “curse of dimensionality.” Some researchers use grid-based partitioning
 27 methods, combined with network pruning. However, based on previous work [20], it is our opinion
 that the results are not as good as the ones obtained from clustering techniques, for the following
 29 reasons: rule-base explosion is avoided in clustering; in grid-partitioning methods with network
 pruning, the wrong nodes may be deleted if the network is not optimized; however, optimization of
 31 a large dimension network is very time consuming; finally, the network must be reoptimized after
 the deletion of nodes.

In this paper, Chiu's subtractive clustering is applied [5]. This scheme possesses some interesting advantages, especially in a neuro-fuzzy identification context. In fact, subtractive clustering is an efficient algorithm, which does not require optimization, being for this reason a good choice for the initialization of neuro-fuzzy networks. Fuzzy c-means and other optimization-based clustering techniques would lead to excessive computer work because they perform an unnecessary optimization phase prior to network training. Also, progressive clustering and compatible cluster merging algorithms are computationally expensive and need metrics for validation of individual clusters [7]. Therefore, despite their potential, they are too complex for a simple initialization of a fuzzy neural network.

Chiu's algorithm belongs to the class of potential function methods, being, more precisely, a variation of the mountain method (see [7]). In this class of algorithms, a set of points are defined as possible group centers, each of them being interpreted as an energy source. In subtractive clustering the center candidates are the data samples themselves, which overcomes the main limitation of the mountain method. In fact, there, the candidates are defined in a grid, leading to "curse of dimensionality" problems.

So, let $Z^N(1)$ be a set of N data samples, z_1, z_2, \dots, z_N , defined in an $m+n$ space, where m denotes the number of inputs and n the number of outputs. In order to make the range of values in each dimension identical, the data samples are normalized, so that they are limited by a hypercube.

As it was referred, it is admitted that each of the samples defines a possible cluster center. Therefore, the potential associated to z_i is (3):

$$P_i(z_i, Z^N) = \sum_{j=1}^N e^{-\alpha \|z_i - z_j\|^2}, \quad \alpha = \frac{4}{r_a^2}, \quad i = 1, 2, \dots, N, \quad (3)$$

where $r_a > 0$ is the *radii* parameter, a constant which defines the neighborhood radius of each point. Thus, points z_j located out of the radius of z_i will have a smaller influence in its potential. On the other hand, the effect of points close to z_i will grow with the proximity. In this way, points with a dense neighborhood will have higher associated potentials.

After computing the potential for each point, the one with the highest potential is selected as the first cluster center. Next, the potential of all the remaining points is reduced. Defining z_1^* as the first group center and denoting its potential as P_1^* , the potential of the remaining points is reduced as in (4):

$$P_i \leftarrow P_i - P_1^* e^{-\beta \|z_i - z_1^*\|^2}, \quad \beta = \frac{4}{r_b^2}, \quad (4)$$

where the constant $r_b > 0$ defines the neighborhood radius with sensitive reductions in its potential.

In this way, points close to the selected center will have their potentials reduced in a more significant manner, and so the probability of being selected as centers diminishes. This procedure has the advantage of avoiding the concentration of identical clusters in denser zones. Therefore, r_b is selected in order to be slightly higher than r_a , so as to avoid closely spaced clusters. Typically, $r_b = 1.5r_a$.

After performing the reduction of potential for all of the candidates, the one with the highest potential is selected as the second cluster. Then, the potential of the remaining points is again

1 reduced. Generically, after determining the r th group, the potential is reduced as follows (5):

$$P_i \leftarrow P_i - P_r^* e^{-\beta \|z_i - z_r^*\|^2}. \quad (5)$$

3 The procedure of center selection and potential reduction is repeated until the following stopping criterion is reached:

Algorithm 1. Stopping criterion for subtractive clustering.

5

If $P_k^* > \varepsilon^{\text{up}} P_1^*$

Accept z_k^* as the next cluster center and continue

Otherwise,

 If $P_k^* < \varepsilon^{\text{down}} P_1^*$

Reject z_k^* and finish the algorithm.

 Otherwise

 Let d_{\min} be the shortest distance between z_k^* and all the centers already found

 If $d_{\min}/r_a + P_k^*/P_1^* \geq 1$

Accept z_k^* as the next cluster center and continue

 Otherwise

Reject z_k^* and assign it the **potential 0.0**.

Select the point with **higher potential** as the new z_k^* .

Repeat the test.

 End If

End If

End If

7 In Algorithm 1, ε^{up} specifies a threshold above which the point is selected as a center with no
 9 doubts and $\varepsilon^{\text{down}}$ specifies the threshold below which the point is definitely rejected. The third case
 is where the point is characterized by a good tradeoff between having a sufficiently high potential
 and being distant enough from the clusters determined before. Typically, $\varepsilon^{\text{up}} = 0.5$ and $\varepsilon^{\text{down}} = 0.15$.

11 As it can be understood from the description of the algorithm, the number of clusters to obtain
 is not pre-specified. However, it is important to note that the *radii* parameter is directly related to
 13 the number of clusters found. Thus, a small radius will lead to a high number of rules, which,
 if excessive, may result in overfitting. On the other hand, a higher radius will lead to a smaller
 15 number of clusters, which may originate underfitting, and so, models with reduced representation
 accuracy. Therefore, in practice it is necessary to test several values for *radii* and select the most
 17 adequate according to the results obtained. However, despite the fact that some *radii* values should be
 tested, this parameter gives an initial hint on the number of clusters necessary [20]. This constitutes
 19 an important advantage over optimization based and other classes of clustering algorithms, when
 little information is known regarding the best number of clusters. Another advantage of subtractive
 21 clustering is that the algorithm is noise robust, since outliers do not significantly influence the choice
 of centers, due to their low potentials.

23 After applying subtractive clustering, each of the obtained clusters will constitute a prototype for
 a particular behavior of the system under analysis. So, each cluster can be used to define a fuzzy

1 rule capable of describing the behavior of the system in some region of the input–output space. Typically, g fuzzy conditional rules of type (6) are obtained:

Rule r :

$$\begin{aligned} &\text{IF } (X_1 \text{ is } LX1^{(r)}) \text{ AND } (X_2 \text{ is } LX2^{(r)}) \text{ AND } \dots \text{ AND } (X_m \text{ is } LXm^{(r)}), \\ &\text{THEN } (Y_1 \text{ is } LY1^{(r)}) \text{ AND } (Y_2 \text{ is } LY2^{(r)}) \text{ AND } \dots \text{ AND } (Y_n \text{ is } LYn^{(r)}), \end{aligned} \quad (6)$$

3 where each of the linguistic terms in the antecedent, $LXj^{(r)}$, has an associated membership function defined as follows (7):

$$5 \quad \mu_{LXj^{(r)}}(x_j) = e^{-\alpha(x_j - x_{rj}^*)^2}, \quad r = 1, 2, \dots, g; \quad j = 1, 2, \dots, m. \quad (7)$$

Here, x_j denotes a numeric value related to the j th input dimension and x_{rj}^* refers to the j th coordinate in the m -dimensional vector x_r^* . Eq. (7) results from the computation of the potential associated to each point in the data space. Clearly, expression (6) is a consequence of using linguistic models, i.e., models in which the consequents are fuzzy sets. Such consequents result naturally from the application of subtractive clustering and are obtained as follows (8):

$$11 \quad \mu_{LYo^{(r)}}(y_o) = e^{-\alpha(y_o - y_{ro}^*)^2}, \quad o = 1, 2, \dots, n, \quad (8)$$

where y_o denotes a numeric value regarding the o th output dimension and y_{ro}^* stands for the j th coordinate in the n -dimensional vector y_r^* .

The definition of an initial structure for Takagi–Sugeno models, in which the terms in the consequents are typically zero and first-order linear functions, is performed similarly. However, since the consequents are not fuzzy sets, the initialization procedure just described applies only to the antecedents. In fact, their values can be easily obtained by means of linear optimization techniques, based on the linear characteristics of the consequents.

19 Comparing (7), (8) and the general equation for Gaussian functions, it turns out that the membership functions considered belong to the referred type. Thus, regarding the standard deviation of each function, expression (9) is obtained:

$$21 \quad \sigma_{rj} = \frac{r_a}{\sqrt{8}}. \quad (9)$$

23 Finally, after the parameterization of the Gaussian membership functions, the data used, previously normalized, are restored to their initial values. In the same way, the function parameters are adjusted to the domains defined for each dimension.

4. Parameter learning

27 4.1. Neuro-fuzzy architecture

After the definition of a structure for the fuzzy model, the model parameters, i.e., the centers and standard deviations of the Gaussian membership functions, must be tuned. In the present work, such task is performed by the training of a fuzzy neural network, based on [16] (Fig. 1).

1 Basically, the previous network architecture is composed of five layers: an input layer, a fuzzi-
 2 fication layer, a rule layer, a union layer and an output or defuzzification layer, in sequential
 3 order.

In order to make the next expressions more readable, the notation used is presented beforehand:

- 5 • m : number of network inputs,
- 6 • n : number of network outputs,
- 7 • g : number of fuzzy rules (groups),
- 8 • $a_i^{(p2)}$: activation of the neuron i in layer 2, regarding the training pattern p (i denotes an input
 9 term: “input”),
- 10 • $a_r^{(p3)}$: activation of the neuron r in layer 3, regarding pattern p (r denotes “rule”),
- 11 • $a_s^{(p4)}$: activation of the neuron s in layer 4, regarding pattern p (s denotes “S-norm”),
- 12 • $a_o^{(p5)} = \hat{y}_o^{(p)}$: activation of the neuron o in layer 5, i.e., output, regarding pattern p (o denotes
 13 “output”),
- 14 • $y_o^{(p)}$: desired activation for neuron o in layer 5, i.e., for the network output, regarding pattern p ;
 15 this is an output sample.

16 In this structure, the *input layer* simply receives data from the external environment and passes
 17 them to the next layer.

18 In the second layer, the *fuzzification layer*, each of the cells corresponds to a membership function
 19 associated to each of the inputs. Defining conventional Gaussian functions, the output of each neuron
 in this layer is given by (10):

$$21 \quad a_i^{(p2)} = e^{-(x_j^{(p)} - c_{ij})^2 / 2\sigma_{ij}^2}, \quad (10)$$

22 where c_{ij} and σ_{ij} represent, respectively, the center and the standard deviation of the i th membership
 23 function corresponding to the j th input. Such parameters are the weights of the links connecting layer
 one and layer two ($LX_j^{(r)}$ in Fig. 1). In the same expression, $x_j^{(p)}$ denotes the p th pattern associated
 25 to input j . Alternatively, it is possible to define two-sided Gaussian functions, which are characterized
 26 by their possibility of being asymmetric and containing a plateau, as a generalization of conventional
 27 functions (Fig. 2). Therefore, the hypothesis of obtaining better results can be formulated, due to
 the increased flexibility of the generalized functions.

28 In case two-sided Gaussians are used, the activation of each of the neurons in this layer is given
 29 by (11):

$$31 \quad a_i^{(p2)} = \begin{cases} e^{-(x_j^{(p)} - c_{ijL})^2 / 2\sigma_{ijL}^2}, & x_j^{(p)} < c_{ijL}, \\ 1, & c_{ijL} \leq x_j^{(p)} \leq c_{ijR}, \\ e^{-(x_j^{(p)} - c_{ijR})^2 / 2\sigma_{ijR}^2}, & x_j^{(p)} > c_{ijR}. \end{cases} \quad (11)$$

32 Here, c_{ijL} and σ_{ijL} represent, respectively, the center and the standard deviation of the left component
 33 of the i th membership function associated to the j th input. For the right component, the index R is
 used. Such parameters form the weights of the links connecting layer one and layer two ($LX_j^{(r)}$ in
 35 Fig. 1). In the same expression, $x_j^{(p)}$ denotes the p th pattern associated to input j .

36 As for the neurons in the *rule layer*, their function consists of performing the antecedent con-
 37 junction of each rule, by means of some T-norm. Normally, algebraic operators, e.g., product, lead

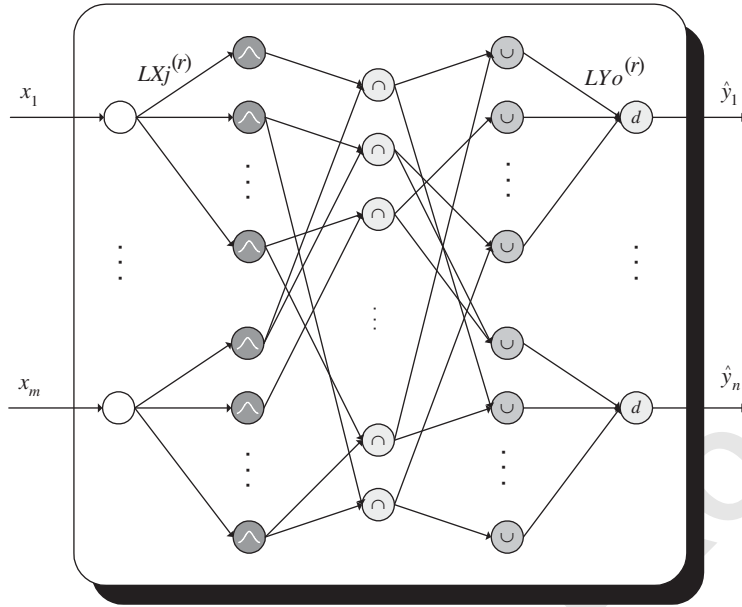


Fig. 1. Neuro-fuzzy network.

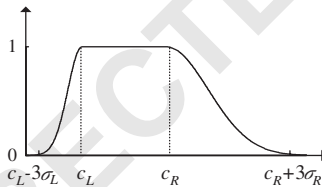


Fig. 2. Two-sided Gaussian function.

1 to better results than truncation operators, e.g., minimum. Namely, they originate smoother output
 2 surfaces and permit the direct application of gradient descent without the artifices needed for trunca-
 3 tion operators, described below. However, it was concluded by experimental testing that truncation
 4 operators lead to better results when some constraints on interpretability are imposed [20]. So, the
 5 *minimum* operator is selected for fuzzy conjunction (12):

$$a_r^{(p3)} = T - \mathop{\text{norm}}_{i=1}^{na_r}(a_i^{(p2)}) = \min_{i=1}^{na_r}(a_i^{(p2)}). \tag{12}$$

7 In (12), na_r stands for the number of inputs in the antecedent of rule r .
 8 The fourth layer, called the *union layer*, is responsible for integrating the rules with the same
 9 consequents, via some S-norm. Once again, truncation operators are preferred, namely the *maximum*

1 operator (13). There, nr_s stands for the number of rules that have neuron s as consequent.

$$a_s^{(p4)} = S - \underset{r=1}{\text{norm}}^{nr_s}(a_r^{(p3)}) = \underset{r=1}{\text{max}}^{nr_s}(a_r^{(p3)}). \quad (13)$$

3 As for the *output layer*, or *defuzzification layer* (d , in Fig. 1), the weights of the links from layer
4 four to layer five ($LYo^{(r)}$ in the same figure) define the parameters of the membership functions
5 associated to the output linguistic terms. Thus, based on these membership functions and on the
6 activation of each rule, its neurons should implement a defuzzification method suited for fuzzy
7 consequents, as the one presented in [16] (14):

$$\hat{y}_o^{(p)} = a_o^{(p5)} = \frac{\sum_{s=1}^{|T(Y_o)|} c_{os} \sigma_{os} a_s^{(p4)}}{\sum_{s=1}^{|T(Y_o)|} \sigma_{os} a_s^{(p4)}}. \quad (14)$$

9 In (14), c_{os} and σ_{os} represent the center and the standard deviation of the s th membership function
10 associated to output o . In case two-sided Gaussians are used, Eq. (15) results, as defined in [21]:

$$\hat{y}_o^{(p)} = a_o^{(p5)} = \frac{\sum_{s=1}^{|T(Y_o)|} \frac{1}{2} (c_{osL} \sigma_{osL} + c_{osR} \sigma_{osR}) a_s^{(p4)}}{\sum_{s=1}^{|T(Y_o)|} \frac{1}{2} (\sigma_{osL} + \sigma_{osR}) a_s^{(p4)}}. \quad (15)$$

13 In the previous expressions, $|T(Y_o)|$ stands for the number of membership functions associated
14 to each linguistic output variable Y_o . The main idea of the proposed defuzzification method is to
15 weight the activation of each rule, not only by the centers, right and left, but also by their standard
16 deviations. Clearly, expression (15) is equivalent to Eq. (14) in case one deals with standard Gaussian
17 functions.

17 4.2. Training methodology

19 Based on the function performed by each neuron, the network is trained in batch mode through
20 backpropagation. The training of the fuzzy neural network starts by defining a criterion for error
21 minimization. An SSE is used. In this way, the total network error E (16), is defined as the sum of
22 squared errors, $E^{(p)}$ (17), computed for each training pattern p , as follows:

$$E = \sum_{p=1}^N E^{(p)}, \quad (16)$$

$$E^{(p)} = \frac{1}{2} \sum_{o=1}^{no} (y_o^{(p)} - \hat{y}_o^{(p)})^2, \quad (17)$$

25 where, $\hat{y}_o^{(p)}$ stands for the p th output pattern regarding the o th output variable, $y_o^{(p)}$ represents the
26 corresponding real output sample and no denotes the number of network outputs.

27 In this way, every pattern is subjected to a forward pass, where signals flow from the input layer
28 to the output layer, allowing the calculation of the error for that specific pattern.

29 Next, starting from the output layer, a backward pass takes place, in which the network parameters
are adjusted towards error minimization. The minimization procedure is conducted iteratively via the

1 gradient descent method, as follows (18):

$$\Delta w = -\gamma \frac{\partial E^{(p)}}{\partial w}, \quad (18)$$

3 where w denotes, generically, any adjustable network parameter, or weight, and γ represents the
5 network learning rate. Typically, the error derivative relative to the weight is calculated by the chain
rule as follows (19):

$$\frac{\partial E^{(p)}}{\partial w} = \frac{\partial E^{(p)}}{\partial a^{(p)}} \frac{\partial a^{(p)}}{\partial w}. \quad (19)$$

7 In the previous expression, $a^{(p)}$ stands for the activation of any neuron in the network. In the
output layer, $a^{(p)}$ is equivalent to some network output, $\hat{y}_o^{(p)}$.

9 As for the delta signals that need to be backpropagated, their values for output neurons are
computed as in (20). For a neuron in layer L_i , its delta value is computed via the chain rule, based
11 on the delta signal of the following layer, L_{i+1} (21). In (21), $n_{L_{i+1}}$ denotes the number of neurons
in layer L_{i+1} :

$$\delta^{(p)} = -\frac{\partial E^{(p)}}{\partial a^{(p)}}. \quad (20)$$

$$\delta^{(pL_i)} = -\frac{\partial E^{(p)}}{\partial a^{(pL_i)}} = \sum_{h=1}^{n_{L_{i+1}}} \left(-\frac{\partial E^{(p)}}{\partial a_h^{(pL_{i+1})}} \frac{\partial a_h^{(pL_{i+1})}}{\partial a^{(pL_i)}} \right) = \sum_{h=1}^{n_{L_{i+1}}} \left(\delta_h^{(pL_{i+1})} \frac{\partial a_h^{(pL_{i+1})}}{\partial a^{(pL_i)}} \right). \quad (21)$$

15 Regarding the five-layered network architecture presented (Fig. 1), the network training equations
were generalized from [16], in order to allow for the integration of two-sided Gaussian functions,
17 as well as the use of truncation operators for conjunction and disjunction.

19 In this way, in regular Gaussians the centers associated to the *output layer* are tuned via Eqs.
(22) and (23):

$$\delta_o^{(p5)} = y_o^{(p)} - \hat{y}_o^{(p)}, \quad (22)$$

$$\frac{\partial E_o^{(p)}}{\partial c_{os}} = -\delta_o^{(p5)} \frac{\sigma_{os} a_s^{(p4)}}{\sum_{k=1}^{|T(Y_o)|} \sigma_{ok} a_k^{(p4)}}. \quad (23)$$

In case two-sided Gaussians are used, Eq. (23) is replaced by (24):

$$\frac{\partial E_o^{(p)}}{\partial c_{osL}} = -\delta_o^{(p5)} \frac{\sigma_{osL} a_s^{(p4)}}{\sum_{k=1}^{|T(Y_o)|} (\sigma_{okL} + \sigma_{okR}) a_k^{(p4)}}. \quad (24)$$

25 The previous equation corresponds to the left-hand side of the Gaussian function (which will be
used throughout this paper). As for the right-hand sided one, the expressions are exactly the same,
except for the subscript L , which is substituted by R .

1 Regarding the standard deviations, their tuning is performed as in (25) for regular Gaussians and
 (26) for two-sided Gaussians:

$$\frac{\partial E_o^{(p)}}{\partial \sigma_{os}} = -\delta_o^{(p5)} \frac{c_{os} a_s^{(p4)} \sum_{k=1}^{|T(Y_o)|} \sigma_{ok} a_k^{(p4)} - a_s^{(p4)} \sum_{k=1}^{|T(Y_o)|} c_{ok} \sigma_{ok} a_k^{(p4)}}{\left[\sum_{k=1}^{|T(Y_o)|} \sigma_{ok} a_k^{(p4)} \right]^2}, \quad (25)$$

$$\frac{\partial E_o^{(p)}}{\partial \sigma_{osL}} = -\delta_o^{(p5)} \frac{c_{osL} a_s^{(p4)} \sum_{k=1}^{|T(Y_o)|} (\sigma_{okL} + \sigma_{okR}) a_k^{(p4)} - a_s^{(p4)} \sum_{k=1}^{|T(Y_o)|} (c_{okL} \sigma_{okL} + c_{okR} \sigma_{okR}) a_k^{(p4)}}{\left[\sum_{k=1}^{|T(Y_o)|} (\sigma_{okL} + \sigma_{okR}) a_k^{(p4)} \right]^2}. \quad (26)$$

5 In the *fourth layer* there are no parameters to adjust. However, the delta signals must be calculated,
 in order to backpropagate them to the inner layers. Those signals are computed based on the same
 signals for the following layer, as stated before, resulting in (27) and (28) for regular and two-sided
 7 Gaussians, respectively:

$$\delta_s^{(p4)} = \sum_{o=1}^{no} \delta_o^{(p5)} \frac{c_{os} \sigma_{os} \sum_{k=1}^{|T(Y_o)|} \sigma_{ok} a_k^{(p4)} - \sigma_{os} \sum_{k=1}^{|T(Y_o)|} c_{ok} \sigma_{ok} a_k^{(p4)}}{\left[\sum_{k=1}^{|T(Y_o)|} \sigma_{ok} a_k^{(p4)} \right]^2}, \quad (27)$$

$$\delta_s^{(p4)} = \sum_{o=1}^{no} \delta_o^{(p5)} \left[\frac{(c_{osL} \sigma_{osL} + c_{osR} \sigma_{osR}) \sum_{k=1}^{|T(Y_o)|} (\sigma_{okL} + \sigma_{okR}) a_k^{(p4)}}{\left[\sum_{k=1}^{|T(Y_o)|} (\sigma_{okL} + \sigma_{okR}) a_k^{(p4)} \right]^2} - \frac{(\sigma_{osL} + \sigma_{osR}) \sum_{k=1}^{|T(Y_o)|} (c_{okL} \sigma_{okL} + c_{okR} \sigma_{okR}) a_k^{(p4)}}{\left[\sum_{k=1}^{|T(Y_o)|} (\sigma_{okL} + \sigma_{okR}) a_k^{(p4)} \right]^2} \right]. \quad (28)$$

11 As for the *third layer*, once again there are no parameters to adjust. So, the only task to perform
 is to calculate the delta signals. Generically, it turns out (29):

$$\delta_r^{(p3)} = \sum_{s=1}^{no_r} \delta_s^{(p4)} \frac{\partial a_s^{(p4)}}{\partial a_r^{(p3)}}, \quad (29)$$

13 where no_r stands for the number of consequents defined for rule r .

15 In the original version [16], the disjunction is performed by means of the bounded-sum. However,
 as stated before, truncation operators are preferable in case interpretability is an objective. Therefore,
 the *maximum* operator is used. However, some care must be taken while calculating the derivative. In
 17 this case, it is necessary to save the index associated to the neuron that originated the maximum.
 Thus, the derivative regarding that element (the “winner”) will be 1, being 0 for the “losers” (30):

$$\frac{\partial a_s^{(p4)}}{\partial a_r^{(p3)}} = \begin{cases} 1, & a_s^{(p4)} = a_r^{(p3)}, \\ 0, & a_s^{(p4)} \neq a_r^{(p3)}. \end{cases} \quad (30)$$

1 In the *second layer*, there are again parameters to adjust. Their tuning is conducted based on Eqs.
 (31) and (32), for the centers, and (33) and (34), for the widths:

$$3 \quad \frac{\partial E_o^{(p)}}{\partial c_{ij}} = \left(\sum_{r=1}^{nr_i} \delta_r^{(p3)} \frac{\partial a_r^{(p3)}}{\partial a_i^{(p2)}} \right) \frac{\partial a_i^{(p2)}}{\partial c_{ij}}, \quad (31)$$

$$\frac{\partial a_i^{(2)}}{\partial c_{ij}} = \frac{x_j - c_{ij}}{\sigma_{ij}^2} e^{-(x_j - c_{ij})^2 / 2\sigma_{ij}^2}, \quad (32)$$

$$5 \quad \frac{\partial E_o^{(p)}}{\partial \sigma_{ij}} = \left(\sum_{r=1}^{nr_i} \delta_r^{(p3)} \frac{\partial a_r^{(p3)}}{\partial a_i^{(p2)}} \right) \frac{\partial a_i^{(p2)}}{\partial \sigma_{ij}}, \quad (33)$$

$$\frac{\partial a_i^{(2)}}{\partial \sigma_{ij}} = \frac{(x_j - c_{ij})^2}{\sigma_{ij}^3} e^{-(x_j - c_{ij})^2 / 2\sigma_{ij}^2}, \quad (34)$$

7 where nr_i represents the number of rules that have neuron i as antecedent and j stands for the input
 variable associated to the i th membership function.

9 In this case, the centers and the standard deviations are tuned in exactly the same manner, both for
 regular and two-sided Gaussians. The only difference is that two-sided Gaussians have two centers
 11 and two standard deviations to adapt.

As in the original version 16, the T-norm is implemented via the *minimum* operator (35):

$$13 \quad \frac{\partial a_r^{(p3)}}{\partial a_i^{(p2)}} = \begin{cases} 1, & a_r^{(p3)} = a_i^{(p2)}, \\ 0, & a_r^{(p3)} \neq a_i^{(p2)}. \end{cases} \quad (35)$$

15 In the previous equation, the same artifice used in (30) was performed, since those operators are
 not continuous.

17 Tuning the parameters without any constraints can lead to inconsistent membership functions. In
 fact, it makes no sense to have neither negative standard deviations nor two-sided Gaussian functions
 with right and left centers exchanged. Therefore, after adjusting the parameters, the integrity of
 19 the membership functions must be verified and guaranteed. Thus, in case the centers in two-sided
 Gaussians get exchanged, it was decided to assign to both of them their mean value. As for the
 21 standard deviations, in case they become negative, they are given a “small” value, which is quantified
 as 1% of the domain amplitude. Formally, it results (36):

$$23 \quad c_L > c_R \Rightarrow \begin{cases} c_L^{\text{new}} = \frac{c_L + c_R}{2}, \\ c_R^{\text{new}} = \frac{c_L + c_R}{2}, \end{cases} \quad (36)$$

$$\sigma < 0 \Rightarrow \sigma = \frac{X_{\max} - X_{\min}}{100},$$

25 where the domain interval is $[X_{\min}, X_{\max}]$. However, it is important to note that, with the imposed
 constraints, the true gradient is not followed any longer. Instead, an approximation is.

1 5. Interpretability

3 The philosophy of fuzzy systems relies on the possibility of linguistic interpretation that char-
 5 acterizes them. Nevertheless, this issue is often ignored, being given prevalent relevance to the
 7 factors related to approximation capabilities. However, as Nauck and Kruse refer [19], in case inter-
 9 pretability is not a major concern, it is important to consider other methods, which might be more
 11 adequate.

7 Thus, as for the neuro-fuzzy identification strategy described in the previous sections, some ques-
 9 tions are addressed regarding the model transparency after learning. In fact, the unconstrained pa-
 11 rameter adjustment may lead to a highly complex set of membership functions, for which it will
 13 be difficult to assign linguistic labels. Therefore, it is important to impose adequate constraints in
 15 parameter tuning, so that interpretability is attained. In the same way, two-sided Gaussian functions
 17 are appealing due to their increased flexibility, which permits the control of function overlapping,
 19 improving function distinguishability.

15 Therefore, three main criteria for model interpretability are defined. The first one, and most im-
 17 portant, is related to the last aspect referred, i.e., function distinguishability. The others result from
 19 human cognitive issues, according to which the number of rules and the number of membership func-
 21 tions associated to each variable should not be excessive. In the present case, the modeler monitors
 23 these issues, in order to obtain a satisfactory tradeoff between model accuracy and interpretability.

19 5.1. Similar membership function merging

21 The first step in order to attain model interpretability consists of finding and merging similar
 23 membership functions.

23 Structure learning by means of clustering techniques leads to a set of initial membership functions
 25 with a high similarity degree. Thus, the model will lack transparency and the number of parameters
 27 to adjust will be excessive, which leads to a higher computational cost. Thus, it seems useful to
 29 merge functions with a high enough similarity degree.

27 In order to perform function merging, directed to the simplification of the rule base, it was
 29 concluded in [24] that S_1 (37) is a very adequate similarity measure. There, the similarity between
 two fuzzy sets A and B is given by the result of the division of the area of their intersection by the
 area of their union:

$$S_1(A, B) = \frac{\|A \cap B\|}{\|A \cup B\|}, \quad (37)$$

31 where the fuzzy intersection and union are performed, respectively, by the *minimum* and *maximum*
 33 operators.

33 After detecting the most similar pair of membership functions, those functions are merged if their
 35 similarity degree is above some threshold. The new function results from averaging the parameters
 37 of the original functions, i.e., the centers and the standard deviations, as depicted in Fig. 3. The
 original functions are represented in dashed lines and the resulting function in solid line.

37 The procedure of membership function merging, one pair for iteration, continues until no more
 39 pairs satisfy the merging threshold. The function created after merging is a merging candidate in the

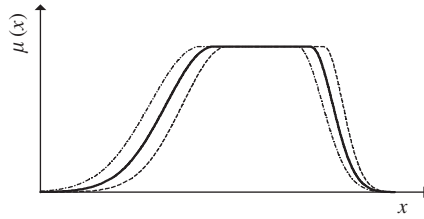


Fig. 3. Membership function merging.

1 following iterations. Therefore, it is important that these functions, obtained via merging, are given a
 2 stronger weight in the merging process. In this way, two fuzzy sets A and B are merged as follows
 3 [24] (38):

$$C_p = \frac{n_A A_p + n_B B_p}{n_A + n_B}. \quad (38)$$

5 In the previous equation, the parameters for the new fuzzy set, C , are computed based on the
 6 weighted mean of the parameters in the original functions. In (38), C_p denotes the vector of pa-
 7 rameters defining fuzzy set C , i.e., its right and left centers and standard deviations, and n_A and
 8 n_B represent the number of pairs of functions merged before A and B were created, respectively.
 9 In case A is the original function, $n_A = 1$; if A is a result of the merging of two original functions,
 10 $n_A = 2$; and so on.

11 As a result of function merging, the rule base must be updated. In fact, the rules related to
 12 the merged fuzzy sets will then contain the new obtained function. Therefore, the premises and
 13 conclusions in the original rules are updated so as to include the new terms. Consequently, the rule
 14 base may be simplified, in case redundant rules appear.

15 5.2. Constrained parameter learning

16 After rule base simplification via function merging takes place, it is important to guarantee that
 17 interpretability is maintained throughout parameter optimization. Thus, it was decided to monitor the
 18 optimization procedure, so that function distinguishability is attained.

19 Therefore, it is assumed that the overlapping degree between two functions is excessive in case
 20 the supreme of the support of the function in the left side, i.e., its right “zero,” goes beyond the
 21 right zero of the function in the right side. The same reasoning applies to the left component of the
 22 functions. Formally, it turns out (39):

$$\begin{aligned} c_{kR} + 3\sigma_{kR} &\leq c_{iR} + 3\sigma_{iR}, \\ c_{kL} - 3\sigma_{kL} &\leq c_{jL} - 3\sigma_{jL}, \end{aligned} \quad (39)$$

23 where k refers to some membership function and i and j stand for its right and left neighbor
 24 functions, respectively. In case function overlapping does not satisfy the constraints presented in
 25 (39), the standard deviations of function k are altered in order to force those conditions. Therefore,

1 the right and left components are changed as in (40) and (41), respectively:

$$\sigma_{kR}^{\text{new}} = \frac{c_{iR} + 3\sigma_{iR} - c_{kR}}{3}, \quad (40)$$

3
$$\sigma_{kL}^{\text{new}} = \frac{c_{jL} - 3\sigma_{jL} - c_{kL}}{-3}. \quad (41)$$

5 Besides monitoring function overlapping, it was concluded that the distance between functions
 6 should also be checked. This procedure aims to avoid inclusions, i.e., functions total or almost
 7 totally included in other functions. Furthermore, the fact that the defined functions are separated
 8 enough also improves model interpretability. Thus, constraint (42) for the minimal distance between
 9 functions was defined:

$$\begin{aligned} c_{iL} - c_{kR} &\leq \alpha(X_{\max} - X_{\min}), \\ c_{kL} - c_{jR} &\leq \alpha(X_{\max} - X_{\min}), \end{aligned} \quad (42)$$

9 where $\alpha \in [0; 1]$ denotes the percentage of the domain $[X_{\min}; X_{\max}]$ used for calculating the minimum
 10 allowed distance. In case this condition does not apply, the function centers are changed as follows
 11 (43):

$$\begin{aligned} c_{kR}^{\text{new}} &= \frac{c_{kR} + c_{iL}}{2} - \frac{\alpha(X_{\max} - X_{\min})}{2}, \\ c_{iL}^{\text{new}} &= \frac{c_{kR} + c_{iL}}{2} + \frac{\alpha(X_{\max} - X_{\min})}{2}. \end{aligned} \quad (43)$$

12 In this situation, the new centers will be based on the average of the right and left centers of
 13 the two functions compared, from which their values are altered in order to guarantee the distance
 14 required.

15 A good tradeoff between model interpretability and accuracy requires that the constraints on pa-
 16 rameter adjustment are somewhat relaxed. In fact, it was concluded experimentally that imposing
 17 strong constraints on the parameters leads to models that are very poor in terms of accuracy. Namely,
 18 the maximum allowable overlapping was tried out with some other criteria, e.g., comparing the left
 19 and right “zeros” of the function under consideration to the centers of its left and right neighbors.
 20 However, though the results obtained for interpretability were satisfactory, the model accuracy was
 21 in general very poor.

22 Therefore, the constraints were somewhat relaxed, resulting in the equations presented above. In
 23 general, these constraints allow a satisfactory model accuracy. However, as a consequence of the
 24 relaxed constraints, interpretability may not be good enough. This limitation can be easily overcome
 25 by periodically merging similar membership functions, i.e., every x epochs.

26 A question that can arise naturally is why not to simply merge similar functions periodically.
 27 This hypothesis was also experimented but the results obtained were not satisfactory regarding inter-
 28 pretability. In fact, some situations occur where functions show no similarity but overlap in complex
 29 ways. For instance, function merging cannot solve inclusions (functions inside other functions) or
 30 functions completely passing through other functions, etc. Instead, the constraints imposed guarantee
 31 that the function overlapping is simple enough to be handled by merging.

6. Simulation results

6.1. The Box–Jenkins gas furnace

The gas furnace [3] is a classic benchmark in system modeling and identification. The data set is composed of 296 input–output pairs. The system input is the gas flowrate (to be burned) to the furnace and the system output is the carbon dioxide concentration in the exhaustion gases. The objective is to predict the output using past values of both the input and the output. In total, 10 input variables are used: $\{y(t-1), y(t-2), y(t-3), y(t-4), u(t-1), u(t-2), u(t-3), u(t-4), u(t-5), u(t-6)\}$. As a consequence of the regression indicated, the final number of input–output pairs reduces to 290.

Several authors have worked out this example, with different number of past values of output and input data. Using Chiu’s method for relevant input selection [6] with the regularity criteria proposed by Sugeno and Yasukawa [27], it was concluded that the most relevant variables to be included in the regression equation are $y(t-1)$ and $u(t-4)$ [20]. Thus, the two referred variables are the only ones used to predict $y(t)$.

Based on the exposed, the gas furnace was modeled using subtractive clustering with $r_a = 0.5$, resulting in three rules. The network, containing two inputs and one output, was trained with a merging threshold of 0.65 and $x = 200$.

After 3000 epochs, the root mean square (RMS) error reached a value of 0.390. As for the number of membership functions for $y(t-1)$, $u(t-4)$ and $y(t)$, 2, 2 and 3 resulted, respectively, leading to 28 adjustable parameters. Fig. 4 presents the approximation results obtained.

As for the membership functions, their shapes are depicted in Fig. 5. As can be seen, it is not too difficult to label each of the membership functions. In the same figure, the labels S , M , and B denote, respectively, the linguistic terms “small,” “medium” and “big.” The fundamental dynamics of the gas furnace are interpreted according to Table 1. The fuzzy system obtained is simple and transparent in the sense that the human judgment can give a clear meaning to the rules.

In Fig. 6, the membership functions obtained for the case of free training (with no interpretability constraints), using the same proposed methodology, are shown. It can be seen that, though they are not very complex (probably because they are only a few), a strong overlapping occurs especially

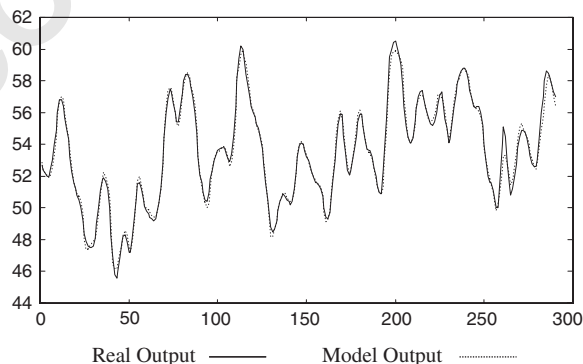


Fig. 4. Box–Jenkins gas furnace: training for interpretability.

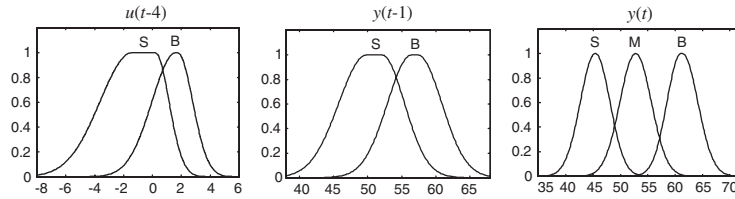


Fig. 5. Gas furnace: membership functions obtained with constrained training.

Table 1
Gas furnace: linguistic description

Rule	$u(t - 4)$	$y(t - 1)$	\Rightarrow	$y(t)$
1	S	S		M
2	S	B		B
3	B	S		S

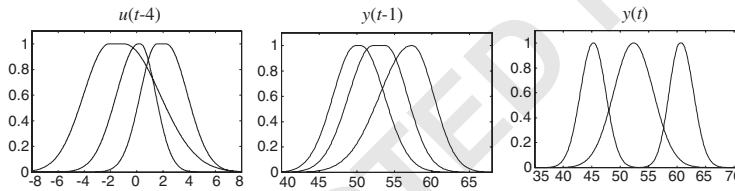


Fig. 6. Gas furnace: membership functions obtained with unconstrained training.

1 in $u(t - 4)$, where it is more difficult to label the membership functions. As for the RMS error, a
 2 value of 0.384 was obtained, which is only slightly below the one for constrained training.

3 This benchmark is characterized by its simplicity regarding interpretability goals. In fact, the RMS
 4 errors for constrained and free training are very similar. This is a direct consequence of the reduced
 5 number of rules and inputs used. Therefore, a more thorough study is presented below.

6.2. The Mackey–Glass chaotic time series

7 One of the most commonly used benchmarks in system identification consists of the prediction of
 8 the Mackey–Glass chaotic time series [17] described by Eq. (44).

$$9 \quad \dot{x}(t) = \frac{0.2x(t - \tau)}{1 + x^{10}(t - \tau)} - 0.1x(t). \quad (44)$$

10 This time series does not show a clear periodic behavior and it is also very sensitive to the initial
 11 conditions. The problem consists of predicting future values of the series.

12 The application of the technique described previously is carried out based on identification data
 13 from the “IEEE Neural Network Council, Standards Committee, Working Group on Data

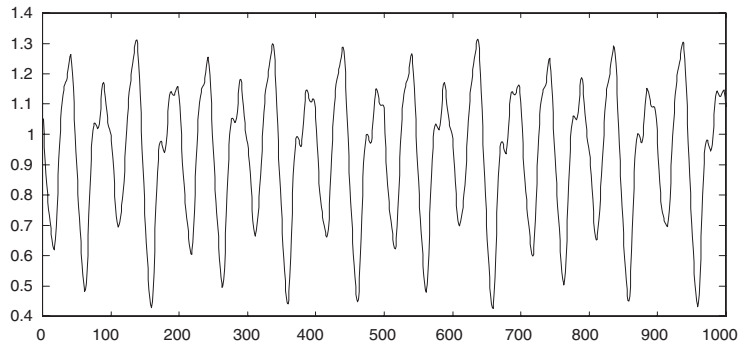


Fig. 7. Chaotic time series: identification data.

1 *Modelling Benchmarks*,” which are also used in the analysis of several other methodologies. So, in
 2 order to obtain a numeric solution for the problem, the fourth-order Runge–Kutta method was ap-
 3 plied. For integration, it was assumed $x(t) = 0$, $t < 0$, and a time interval of 0.1. The initial condition
 4 $x(0) = 1.2$ and the parameter $\tau = 17$ were also defined. In this case, $[x(t - 18), x(t - 12), x(t - 6),$
 5 $x(t)]$ are used to predict $x(t + 6)$. Based on the described parameterization, data were collected from
 6 the interval $t \in [0; 2000]$. Then, 1000 input–output pairs were selected for training, from the inter-
 7 val $t \in [118; 1117]$. The remaining data were used for validation. The data collected are depicted in
 8 Fig. 7.

9 Using the samples obtained, the chaotic time series was modeled, according to the procedures
 10 described in the previous sections. So, parameter r_a was assigned a value of 0.5, resulting in nine
 11 fuzzy rules. As happened with the gas furnace, the network, containing four inputs and one output,
 12 was trained with a merging threshold of 0.65 and $x = 200$. After 800 epochs the RMS error was
 13 0.0228 for the training data and 0.0239 for the test data. As for the number of membership functions
 14 for the variables $x(t - 18)$, $x(t - 12)$, $x(t - 6)$, $x(t)$ and $x(t + 6)$, 5, 4, 5, 4 and 5 were obtained,
 15 respectively, leading to 92 adjustable parameters.

16 In Fig. 8, the approximation results for the test data are depicted, which, by visual inspection,
 17 seem to capture the main dynamics of the system.

18 As for the membership functions, these are presented in Fig. 9. As can be seen, it is not difficult to
 19 assign linguistic terms to each of the membership functions. In the same figure, the labels VS , S , M ,
 20 B and VB denote, respectively, the linguistic terms “very small,” “small,” “medium,” “big” and “very
 21 big.” Thus, the fundamental dynamics of the chaotic time series are interpreted according to Table 2.

22 Comparing to NEXPROX [19] (Table 3), the results obtained show improvements: the number
 23 of rules is smaller, respecting the human cognitive needs, and the RMS error is also smaller. This
 24 results in both better accuracy and interpretability.

25 In Fig. 10, the membership functions obtained for the case of free training, using the same proposed
 26 methodology, are shown. It can be seen that they are impossible to interpret linguistically. As for the
 27 RMS error, 0.0076 was obtained for the unconstrained training. This value is much smaller than the
 28 one obtained for constrained training, as expected. In fact, it can be stated that interpretability and
 29 accuracy are conflicting requirements. As long as interpretability constraints grow more demanding,
 accuracy diminishes and vice versa.

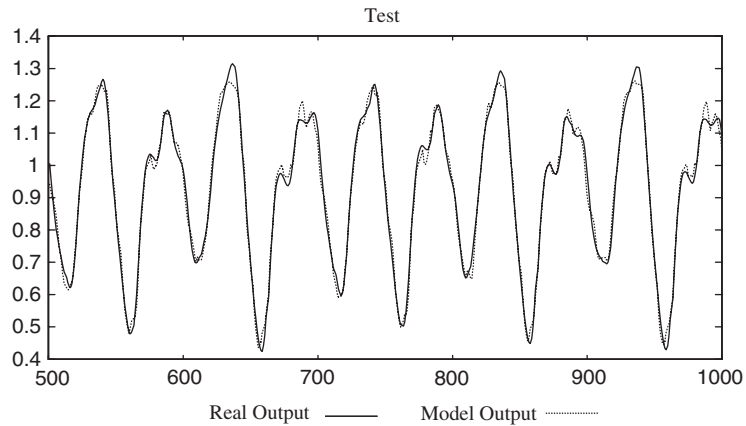


Fig. 8. Chaotic series: output prediction.

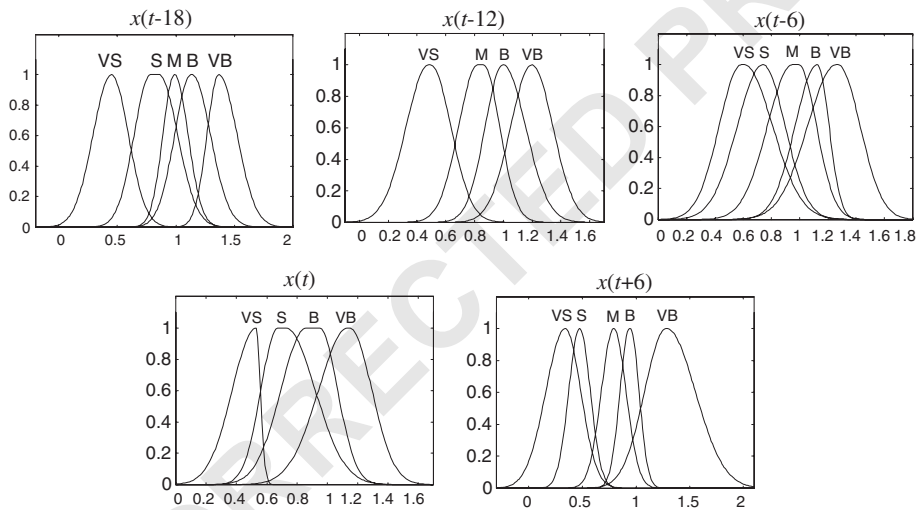


Fig. 9. Chaotic series: membership functions obtained.

1 7. Conclusions

3 In this paper, a neuro-fuzzy methodology for the implementation of real interpretable fuzzy models
 5 is described. By the application of subtractive clustering, an initial structure for the fuzzy model is
 7 obtained, which is then used for the initialization of a fuzzy neural network. However, adjusting
 9 membership function parameters without any constraints usually leads to a complex overlapping between
 functions, which limits interpretability. Therefore, a learning scheme to allow the development of
 interpretable fuzzy models is proposed. The methodology presented is based on the merging of
 similar membership functions and on the constrained tuning of model parameters. This strategy aims
 to improve function distinguishability in terms of distance and overlapping. The approach described

Table 2
Chaotic series: linguistic description

Rule	$x(t - 18)$	$x(t - 12)$	$x(t - 6)$	$x(t)$	\Rightarrow	$x(t + 6)$
1	M	VB	B	VB		B
2	B	VB	M	S		S
3	S	M	M	VB		VB
4	M	M	VS	VB		B
5	S	B	S	VS		M
6	S	VB	VB	B		M
7	S	VS	S	B		B
8	VS	VS	M	B		B
9	VB	VB	VB	B		VS

Table 3
Chaotic series: comparison with other techniques

Method	No. of rules	No. of param.	RMSE
Paiva and Dourado	9	92	0.0239
NEFPROX (A)	129	105	0.0332
NEFPROX (G)	26	38	0.0671

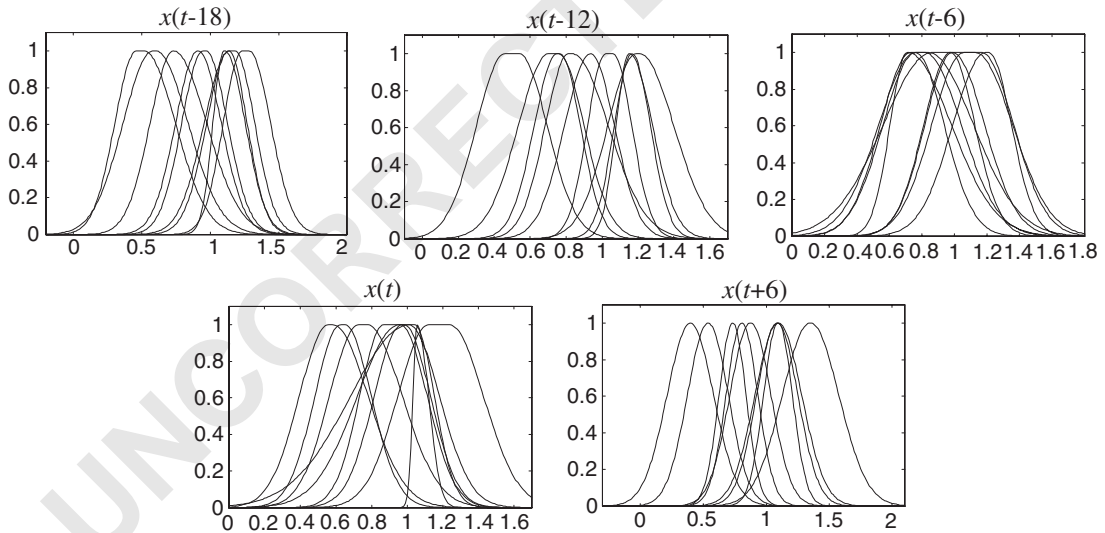


Fig. 10. Chaotic series: membership functions after free training. Compare with Fig. 9.

1 is applied to the prediction of the Mackey–Glass chaotic time series and to the Box–Jenkins gas
 furnace, resulting in a satisfactory tradeoff between model accuracy and interpretability. In more
 3 complex models, e.g., industrial processes, the constraints imposed for interpretability may lead to

1 inaccurate models. Thus, as a summary, it can be said that interpretability bounds accuracy and vice
versa.

3 Acknowledgements

This work was partially supported by the Portuguese Ministry of Science and Technology (MCT),
under the program PRAXIS XXI/Alcine Project.

References

- 5 [1] J. Abonyi, L. Nagy, F. Szeifert, Adaptive fuzzy inference system and its application in modeling and model-based
control, *Chem. Eng. Res. Des.* 77 (1999) 281–290.
- 7 [2] E. Binaghi, I. Gallo, P. Madella, A neural model for fuzzy Dempster–Shafer classifiers, *Internat. J. Approx. Reason.*
25 (2000) 89–121.
- 9 [3] G.E.P. Box, G.W. Jenkins, *Time Series Analysis, Forecasting and Control*, Holden Day, San Francisco, 1970.
- 11 [4] J.L. Castro, Fuzzy logic controllers are universal approximators, *IEEE Trans. Systems, Man Cybernet.* 25 (1995)
629–635.
- 13 [5] S.L. Chiu, Fuzzy model identification based on cluster estimation, *J. Intell. Fuzzy Systems* 2 (1994) 267–278.
- 15 [6] S.L. Chiu, Selecting input variables for fuzzy models, *J. Intell. Fuzzy Systems* 4 (1996) 243–256.
- 17 [7] R.N. Davé, R. Krishnapuram, Robust clustering methods: a unified view, *IEEE Trans. Fuzzy Systems* 5 (1997)
270–293.
- 19 [8] J. Espinosa, J. Vandewalle, Constructing fuzzy models with linguistic integrity from numerical data-AFRELI
Algorithm, *IEEE Trans. Fuzzy Systems* 8 (2000) 591–600.
- 21 [9] D. Fuessel, R. Isermann, Hierarchical motor diagnosis utilizing structural knowledge and a self-learning neuro-fuzzy
scheme, *IEEE Trans. Ind. Electron.* 47 (2000) 1070–1077.
- 23 [10] M.B. Gorzalczany, Z. Piasta, Neuro-fuzzy approach versus rough-set inspired methodology for intelligent decision
support, *Inform. Sci.* 120 (1999) 45–68.
- 25 [11] S. Guillaume, Designing fuzzy inference systems from data: an interpretability-oriented review, *IEEE Trans. Fuzzy
Systems* 9 (2001) 426–443.
- 27 [12] H. Ishibushi, T. Nakashima, T. Murata, Three-objectives genetics-based machine learning for linguistic rule extraction,
Inform. Sci. 136 (2001) 109–133.
- 29 [13] Y.C. Jin, Fuzzy modelling of high-dimensional systems: complexity reduction and interpretability improvement, *IEEE
Trans. Fuzzy Systems* 8 (2000) 212–221.
- 31 [14] E.P. Klement, L.T. Koczy, B. Moser, Are fuzzy systems universal approximators? *Internat. J. Gen. Systems* 28
(1999) 259–282.
- 33 [15] N. Lavrac, Selected techniques for data-mining in medicine, *Artif. Intell. Med.* 16 (1999) 3–23.
- 35 [16] C.-T. Lin, A neural fuzzy control scheme with structure and parameter learning, *Fuzzy Sets and Systems* 70 (1995)
183–212.
- 37 [17] M.C. Mackey, L. Glass, Oscillation and chaos in physiological control systems, *Science* 197 (1977) 287–289.
- 39 [18] H.R. Maier, T. Sayed, B.J. Lence, Forecasting cyanobacterial concentrations using B-spline networks, *J. Comput.
Civil Eng.* 14 (2000) 183–189.
- 41 [19] D. Nauck, R. Kruse, Neuro-fuzzy systems for function approximation, *Fuzzy Sets and Systems* 101 (1999)
261–271.
- [20] R.P. Paiva, Identificação neuro-Difusa: aspectos de interpretabilidade (Neuro-fuzzy identification: interpretability
issues), M.Sc. Thesis, Department of Informatics Engineering, Faculty of Sciences and Technology, University of
Coimbra, Portugal, 1999 (in Portuguese).
- [21] R.P. Paiva, A. Dourado, B. Duarte, Applying subtractive clustering for neuro-fuzzy modelling of a bleaching plant,
in: *Proceedings of the European Control Conference ECC'99*, 1999.

- 1 [22] H. Roubos, M. Setnes, Compact and transparent fuzzy models and classifiers through iterative complexity reduction, *IEEE Trans. Fuzzy Systems* 9 (2001) 516–524.
- 3 [23] L. Sanchez, I. Couso, J.A. Corrales, Combining GP operators with SA search to evolve fuzzy rule based classifiers, *Inform. Sci.* 136 (2001) 175–191.
- 5 [24] M. Setnes, Fuzzy rule-base simplification using similarity measures, M.Sc. Thesis, Department of Electrical Engineering, Delft University of Technology, The Netherlands, 1995.
- 7 [25] J. Shanahan, B.T. Thomas, M. Mirmehdi, M. Martin, N. Campbell, J. Baldwin, A soft computing approach to road classification, *J. Intell. Robot. Systems* 29 (2000) 349–387.
- 9 [26] A. Stoica, Learning eye-arm coordination using neural and fuzzy techniques, in: H.-N. Teodorescu, A. Kandel, L.C. Jain (Eds.), *Soft Computing in Human Related Sciences*, 1999, pp. 37–66.
- 11 [27] M. Sugeno, T. Yasukawa, A fuzzy-logic based approach to qualitative modeling, *IEEE Trans. Fuzzy Systems* 1 (1993) 7–31.
- 13 [28] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modelling and control, *IEEE Trans. Systems, Man Cybernet.* 15 (1985) 116–132.
- 15 [29] L.A. Zadeh, Outline of a new approach to the analysis of complex systems and decision processes, *IEEE Trans. Systems, Man Cybernet.* 3 (1973) 28–44.