Fuzzy Sets and Systems 130 (2002) 189–205

www.elsevier.com/locate/fss

Hierarchical neuro-fuzzy quadtree models

Flávio Joaquim de Souza[a], Marley M.R. Vellasco[b, *], Marco A.C. Pacheco[b]

[a]*Department of Computer & Systems Engineering, UERJ, Rua Sao Francisco Xavier, 524, Bloco D, Rio de Janeiro, RJ, 20550-900, Brazil*

[b]*ICA: Applied Computational Intelligence Laboratory, Department of Electrical Engineering, PUC-Rio, Rua Marques de Sao Vicente 225, Rio de Janeiro, RJ, 22543-900, Brazil*

Received 10 February 2000; received in revised form 10 April 2001; accepted 15 April 2001

Abstract

Hybrid neuro-fuzzy systems have been in evidence during the past few years, due to its attractive combination of the learning capacity of artificial neural networks with the interpretability of the fuzzy systems. This article proposes a new hybrid neuro-fuzzy model, named *hierarchical neuro-fuzzy quadtree* (HNFQ), which is based on a recursive partitioning method of the input space named quadtree. The article describes the architecture of this new model, presenting its basic cell and its learning algorithm. The HNFQ system is evaluated in three well known benchmark applications: the sinc(x, y) function approximation, the Mackey Glass chaotic series forecast and the two spirals problem. When compared to other neuro-fuzzy systems, the HNFQ exhibits competing results, with two major advantages it automatically creates its own structure and it is not limited to few input variables. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Learning; Neuro-fuzzy system; Fuzzy system models; Quadtree recursive partitioning

1. Introduction

Hybrid neuro-fuzzy systems, or simply neuro-fuzzy systems (NFS) [5,8,12–14,16–18,20–22,25,26,29], match the learning capacity and the fault tolerance of artificial neural networks (ANN) [9,15] with the linguistic interpretation ability of the fuzzy systems (FS) [24]. This feature has made them the focus of intense research in the last years. The learning paradigms of neural networks allow the NFS to learn new information from a specific data set. On the other hand, the theory of fuzzy sets allows the NFS to present the information learned in a format which is interpretable by humans.

The standard NFS learning process is divided in two stages: the *structure identification*, where the input/output space partitioning and number of rules are evaluated; and the *parameters identification* where fuzzy weights and weights associated with the rules are adjusted. Depending on the neuro-fuzzy system in question, these stages can occur simultaneously or not.

In the structure identification stage, a pre-defined method to partition the IN/OUT spaces is initially

* Corresponding author. Tel.: +55-21-2529-9445; fax: +55-21-2511-5154.

E-mail addresses: fjsouza@eng.uerj.br (F.J. de Souza), marley@ele.puc-rio.br (M.M.R. Vellasco), marco@ele.puc-rio.br (M.A.C. Pacheco).

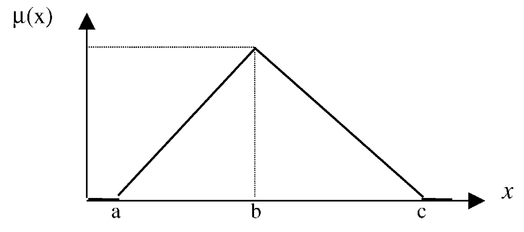

Fig. 1. Membership function of triangular profile.

specified. The result of this partitioning greatly influences the structure and the rule base obtained after the training phase is completed.

In the parameters identification stage, the fuzzy weights are established. These fuzzy weights are the parameters that define the format of the membership functions used in rules antecedents and consequents, as well as the rule's weights. Fig. 1 illustrates a typical example of a triangular membership function $\mu(x)$, used in NFS rules, and its three parameters $(a, b$ and $c)$ which define its format.

Both FS and NFS map fuzzy regions of the input space into fuzzy regions of the output space. This mapping is accomplished through the system's fuzzy rules. The fuzzy regions of the input/output space are determined in the structure identification process, according to a specific method. The most common partitioning methods used by the NFS currently found in literature are: *Fuzzy Grid*, *Adaptive Fuzzy Grid*, *Fuzzy Boxes* and *Fuzzy Clusters*. These methods are illustrated in Fig. 2.

The Fuzzy Grid [5] (Fig. 2a) is a fixed partitioning that does not allow any adjustment in the membership functions. Systems that use this partitioning only update parameters in the consequent of the rule. The Adaptive Fuzzy Grid partitioning in Fig. 2b allows the adjustment in the membership functions shape [12–14]. The Fuzzy Grid partitioning methods (fixed and adaptive) are simple and intuitive. However, as the number of input variable increases, the number of partitions grows accordingly, generating the *curse of dimensionality* problem. Other partitioning methods, presented in Figs. 2c and d, include Fuzzy Box and Fuzzy Cluster partitioning, respectively. The Fuzzy Box (2c) appears in the system fuzzy self-organized map—FSOM [29]. The Fuzzy Cluster (2d) is generated by neural networks such as radial basis function networks (RBFs) [9].

The partitioning process of the IN/OUT space has great influence on the NFS performance in relation to its desirable features (accuracy, generalisation, automatic generation of rules, etc.). This work proposes the use of recursive partitioning methods of the IN/OUT space, resulting in a new class of NFS, called hierarchical neuro-fuzzy quadtree (HNFQ), that overcomes two of the main weaknesses of current NFS: their fixed structure and limited number of inputs. It has been shown by Brown and Harris [2] that a rule base using an hierarchical structure leads to a linear growth in the number of the rules.

The quadtree partitioning was inspired by the quadtree structures proposed by Finkel and Bentley in 1974 [7], which has been widely used in the area of image manipulation and compression. The proposed HNFQ model has been implemented and tested in three classic benchmarks: sinc(x, y) function approximation, Mackey Glass chaotic series forecast, and the two spirals problem.

The remainder of this article is divided in five more sections. The following section describes the most common recursive partitioning methods and their representations. Section 3 introduces the recursive hierarchical neuro-fuzzy model (HNFQ), its basic cell and its architecture. Section 4 presents the HNFQ learning algorithm, which is based on the gradient descent method for parameters adjustment, and uses the

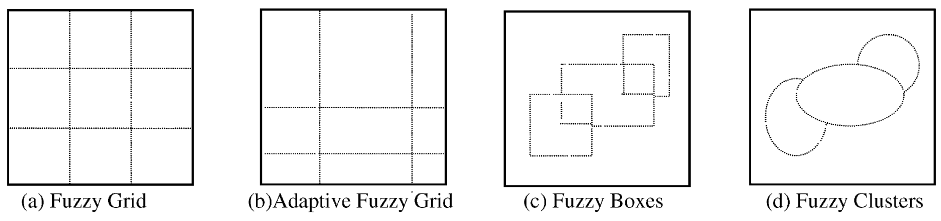

Fig. 2. NFS partitioning methods.

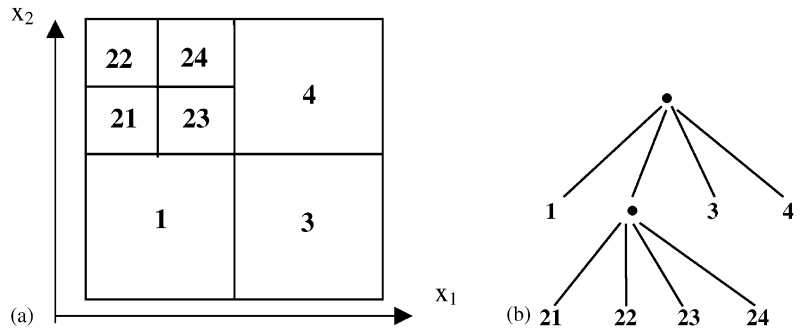

Fig. 3. (a) Quadtree partitioning example, (b) Quadtree tree representing quadtree partitioning.

recursive feature of quadtree partitioning for its structure development. In Section 5, the results of the tests performed with the HNFQ, in three distinct applications, are shown. Finally, Section 6 presents the conclusions and future perspectives of this work.

2. Recursive partitioning

We consider basically two recursive partitioning forms in our NFS hierarchical approach:

(a) BSP partitioning (binary space partitioning) [3,4]; and
(b) quadtree partitioning [7].

Such forms of partitioning the space are said to be recursive because they use recursive processes in their generation. These partitioning forms have been chosen in order to preserve the independence of the input features, which is fundamental to achieve interpretability. Other recursive partitioning forms, such as the polygonal, mixes the input features just like multilayer perceptrons do, losing the interpretation capability.

The following subsections briefly introduce the BSP [3,4] partitioning and detail the quadtree [2] partitioning, on which this article is based.

2.1. BSP partitioning

In this type of partitioning, the space is successively divided in two regions, in a recursive way. This partitioning can be represented by a binary tree that illustrates the successive n-dimensional space subdivisions in two convex subspaces. The construction of this partitioning tree (BSP tree) is a process in which a subspace is divided by a hyper-plan parallel to the co-ordinates axes. This process results in two new subspaces that can be later partitioned by the same method.

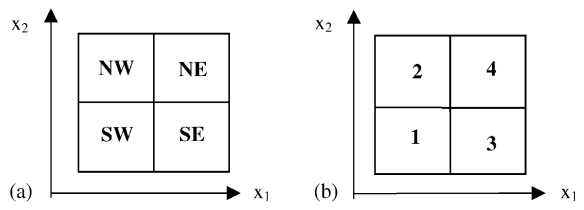

Fig. 4. (a) Conventional quadtree partitions designation, (b) this work convention.

2.2. Quadtree partitioning

In this partitioning, the n-dimensional space is successively subdivided in quadrants that, in turn, can be again subdivided in four regions (quadrants) in a recursive operation.

Fig. 3a illustrates this type of partitioning for the two-dimensional case, and Fig. 3b shows its representative quadtree tree [7]. The designation of partitions showed in this figure follows a different convention from the one usually applied for representing pictures by the quadtree partitioning. Figs. 4a and b illustrate these two designation forms.

The traditional convention uses the directions NW (north–west), NE (north–east), SW (south–west) and SE (south–east), of the "*Compass Rose*", to nominate each quadrant. The convention used in this work

Table 1
Relation between quadrants numeration and μ_2 and μ_4 values

# Quadrant	$x_1(\mu_2)$	$x_2(\mu_4)$
$1 = \mathbf{00} + 1$	Low membership degree (**0**)	Low membership degree (**0**)
$2 = \mathbf{01} + 1$	Low membership degree (**0**)	High membership degree (**1**)
$3 = \mathbf{10} + 1$	High membership degree (**1**)	Low membership degree (**0**)
$4 = \mathbf{11} + 1$	High membership degree (**1**)	High membership degree (**1**)

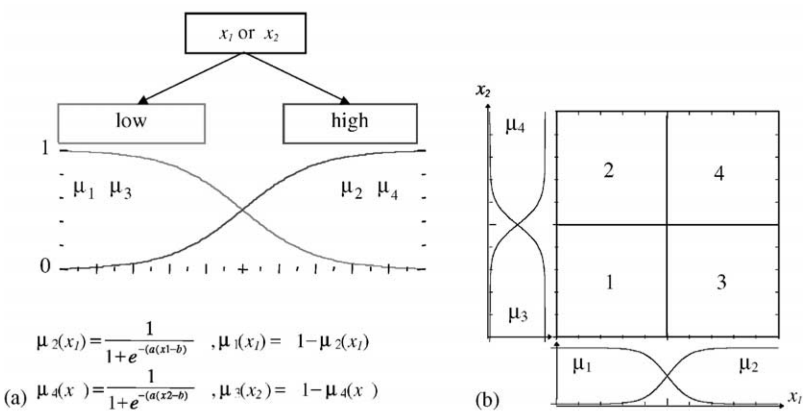
x_1 or x_2
low
high
1
0
μ_1 μ_3
μ_2 μ_4
$\mu_2(x_1) = \frac{1}{1+e^{-(a(x1-b))}}$, $\mu_1(x_1) = 1-\mu_2(x_1)$
$\mu_4(x) = \frac{1}{1+e^{-(a(x2-b))}}$, $\mu_3(x_2) = 1-\mu_4(x)$
(a)
x_2
μ_4
μ_3
2
4
1
3
μ_1
μ_2
x_1
(b)

Fig. 5. (a) The high and low membership functions, (b) division into 4 quadrants performed by the sigmoid membership functions.

is associated to a binary numbering as explained below.

The universe of discourse of the variable x_1 is divided in two fuzzy sets, μ_1 (low) and μ_2 (high), and the universe of discourse of the variable x_2 is divided in two fuzzy sets, μ_3 (low) and μ_4 (high). Table 1 shows the relation of the numbers adopted for the quadrants and the regions where membership functions high (μ_2 and μ_4) assume its minimum or maximum membership degree.

Fig. 5a shows the shape and the analytical expressions of high and low membership functions, which divide the input space generated by x_1 and x_2. The analytical expressions of high and low membership functions are given by a sigmoid function and its complement to 1, respectively. The "a" and "b" constants determine, respectively, the fuzzy set inclination and the middle point of the transition between the values zero and one. Other shape can be used for these membership functions, however, the sigmoid has been chosen due to its simple form. Fig. 5b shows how the four

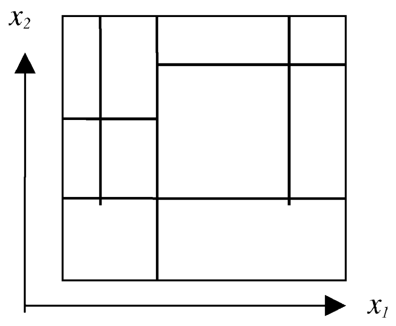

Fig. 6. Adaptive quadtree partitioning.

membership functions (μ_1, μ_2, μ_3 and μ_4) generate the division in 4 quadrants with fuzzy borders.

The quadtree partitioning can be fixed (Fig. 3a) or adaptive as shown in Fig. 6, where the partitioning borders are adjusted during the NFS training (this will be discussed in Section 4). In this case, the regions generated in each subdivision are rectangular and not square shaped, as in the fixed partitioning case.

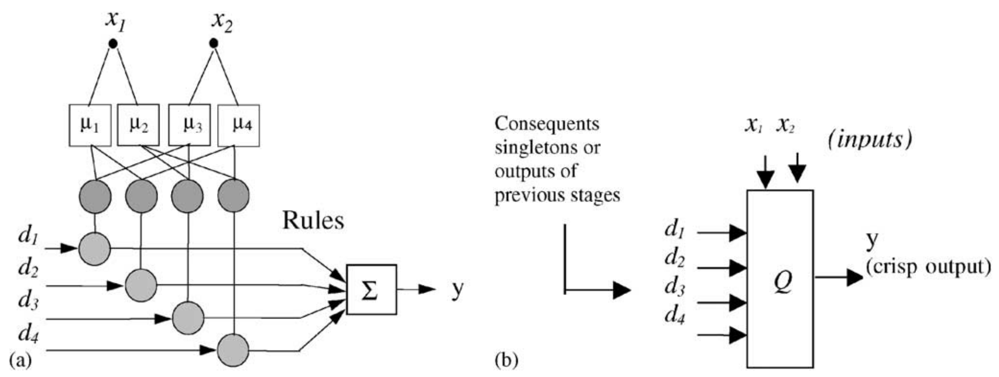

Fig. 7. (a) Hierarchical neuro-fuzzy quadtree, (b) HNFQ simplified cell diagram.

3. Hierarchical neuro-fuzzy quadtree model (HNFQ)

The HNFQ models form a new class of recursive NFS with a dynamic structure, which evolves to increase accuracy, without incurring in the dimensionality problem.

In this model, the generation of rules is obtained with an automatic partitioning process. The combinatory explosion of the number of rules (dimensionality problem) is minimised with a partitioning method that only details the deficient regions in which an increase in the accuracy is necessary. Both types of recursive partitioning, quadtree and BSP, described in Section 2, present this desirable feature.

An hierarchical model in the scope of this work refers to a model where each input space partitioning defines a subsystem that, in turn, is another model with the same structure (recursiveness). This model is inspired by Takagi–Sugeno fuzzy system [13,14], in which a fuzzy rule has the following format:

$$\text{if } x_1 \in \mu_1 \quad \text{and} \quad x_2 \in \mu_2 \quad \text{then } y = f(x_1,x_2). \qquad (1)$$

Usually, the function $f(x_1,x_2)$ is a simple constant (fuzzy singleton) or a linear combination of x_1 and x_2 inputs ($a_1x_1 + a_2x_2 + a_3$). In the HNFQ model, the function in the rule consequent can take various forms: a NFS of the same type; a "singleton"; a linear combination of inputs; or a fuzzy set of a particular shape. When the rule consequent is implemented by a NFS, a subpartitioning of the input space is obtained. This subpartitioning can follow the quadtree format or any other format, such as the BSP.

To accomplish each partitioning in a quadtree format, a mini NFS, called *neuro-fuzzy quadtree cell*, is used. This cell is described in details in the next subsection.

3.1. Neuro-fuzzy quadtree cell

A HNFQ model is composed of one or more standard cells, named neuro-fuzzy quadtree cells. The cell in the highest level of the hierarchy generates the system's output, while the cells in the lowest level of the hierarchy behave as consequents of the higher hierarchy cells. Intermediate and output cells have as consequent the output of the lower hierarchy cells. Section 3.2 presents an example of this model.

A HNFQ cell is a neuro-fuzzy mini-system that performs a quadtree partitioning of a certain space, according to the membership functions described in Section 2, and generates a crisp output (y) after a defuzzification process, as will be shown later. Fig. 7a illustrates a HNFQ cell and Fig. 7b presents its simplified diagram.

The linguistic interpretation of the mapping implemented by the cell in Fig. 7a is given by the following set of rules:

$$\begin{aligned}
&\text{if } x_1 \in \mu_1 && \text{and} && x_2 \in \mu_3 && \text{then } y = d1, && \text{rule 1;}\\
&\text{if } x_1 \in \mu_1 && \text{and} && x_2 \in \mu_4 && \text{then } y = d2, && \text{rule 2;}\\
&\text{if } x_1 \in \mu_2 && \text{and} && x_2 \in \mu_3 && \text{then } y = d3, && \text{rule 3;}\\
&\text{if } x_1 \in \mu_2 && \text{and} && x_2 \in \mu_4 && \text{then } y = d4, && \text{rule 4.}
\end{aligned} \qquad (2)$$

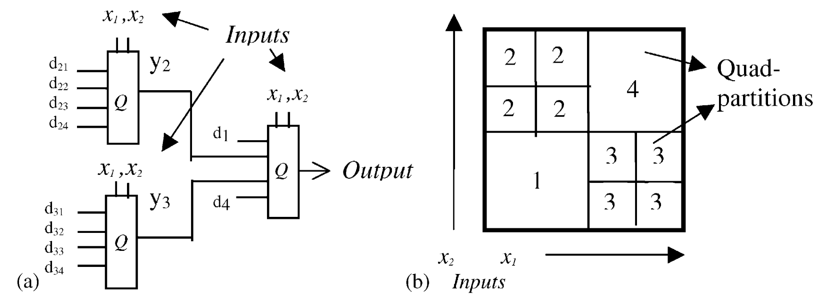

Fig. 8. (a) An example of HNFQ architecture, (b) Input space partitioning of HNFQ architecture.

Each rule corresponds to one of the quadrants shown in Fig. 4b. When the inputs fall in quadrant 1, rule 1 has the greatest firing level. When the inputs fall in quadrant 2, rule 2 has the greatest firing level. When the inputs fall in quadrant 3, rule 3 has the greatest firing level and, finally, when the inputs fall in quadrant 4, it is the rule 4 that has the greatest firing level. Each quadrant, in turn, can be subdivided in four parts, through another HNFQ cell.

In the HNFQ cell, x_1 and x_2 inputs produce the four fuzzy rules antecedents, after the values of $\mu_1(x_1), \mu_2(x_1), \mu_3(x_2), \mu_4(x_2)$ have been computed. The output of a HNFQ cell is given by the weighted mean

$$y = \left(\frac{\sum_{i=1}^{4} \alpha_i \times d_i}{\sum_{i=1}^{4} \alpha_i} \right), \tag{3}$$

where d_i corresponds to a singleton, a linear combination of inputs, or an output of a previous level cell; and α_i's corresponds to the firing levels of each rule, obtained by a T-norm operation of the rules' antecedents. Thus, $\alpha_1 = \mu_1(x_1)\mu_3(x_2)$; $\alpha_2 = \mu_1(x_1)\mu_4(x_2)$; $\alpha_3 = \mu_2(x_1)\mu_3(x_2)$; and $\alpha_4 = \mu_2(x_1)\mu_4(x_2)$.

As can be seen in Fig. 5a, the membership functions have been implemented by sigmoids for μ_2 and μ_4, and by its complement to one $(1 - \mu(x))$, for μ_1 and μ_3. The use of the complement to one simplifies the defuzzification procedure performed by the weighted mean process (Eq. (3)), since the summation in the denominator of Eq. (3) is equal to 1 for any value of x_1 and x_2, as Eq. (4) states

$$\sum_{i=1}^{4} \alpha_i = 1. \tag{4}$$

The output of a basic cell can, therefore, be simplified as shown by Eq. (5). This simplification will also ease the task of obtaining the equations for parameters adjustment (see Appendix A).

$$y = \sum_{i=1}^{4} \alpha_i \times d_i. \tag{5}$$

3.2. HNFQ architecture

HNFQ models are built from the interconnection of the HNFQ cells. This is exemplified in Fig. 8a. The partitioning implemented by the small system illustrated in Fig. 8a is shown in Fig. 8b. Each non-subdivided partition is called *quad-partition*.

In the above example, quadrants 1 and 4 have not been subdivided. Therefore, the consequents of their respective rules are only singletons d_1 and d_4. Quadrants 2 and 3 have been subdivided and the consequents of its rules are the outputs (y_2 and y_3) of subsystems 2 and 3 that, in turn, have, as consequent of its rules, singletons $d_{21}, d_{22}, d_{23}, d_{24}$ and $d_{31}, d_{32}, d_{33}, d_{34}$, respectively.

The output of this system is given by

$$y = \alpha_1 \times d_1 + \alpha_4 \times d_4 + \sum_{i=2}^{3} \sum_{j=1}^{4} \alpha_i \times \alpha_{ij} \times d_{ij}. \tag{6}$$

The partitioning in Fig. 8b can be represented by a quaternary tree ("quadtree tree") with the structure shown in Fig. 9.

In this tree, circles represent internal nodes that correspond to regions that have been subdivided. The squares represent terminals nodes that correspond to quad-partitions, that is, regions that have not been

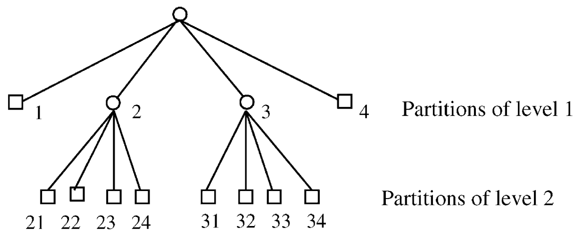

Fig. 9. A quadtree tree representing Fig. 8b partitioning.

subdivided. The root of the tree symbolises the whole space to be partitioned. The set of rules that translate the example of Fig. 8a is

if $x_1 \in \mu_1$ and $x_2 \in \mu_3$ then $y = d_1$

if $x_1 \in \mu_1$ and $x_2 \in \mu_4$ then

{if $x_1 \in \mu_{21}$ and $x_2 \in \mu_{23}$ then $y = d_{21}$

if $x_1 \in \mu_{21}$ and $x_2 \in \mu_{24}$ then $y = d_{22}$

if $x_1 \in \mu_{22}$ and $x_2 \in \mu_{23}$ then $y = d_{23}$

if $x_1 \in \mu_{22}$ and $x_2 \in \mu_{24}$ then $y = d_{24}$}

if $x_1 \in \mu_2$ and $x_2 \in \mu_3$ then

{if $x_1 \in \mu_{31}$ and $x_2 \in \mu_{33}$ then $y = d_{31}$

if $x_1 \in \mu_{31}$ and $x_2 \in \mu_{34}$ then $y = d_{32}$

if $x_1 \in \mu_{32}$ and $x_2 \in \mu_{33}$ then $y = d_{33}$

if $x_1 \in \mu_{32}$ and $x_2 \in \mu_{34}$ then $y = d_{34}$}

if $x_1 \in \mu_2$ and $x_2 \in \mu_4$ then $y = d_4$,

where

- μ_1, μ_2, μ_3 and μ_4 are the membership functions which define the level 1 partitioning.
- μ_{21}, μ_{22}, μ_{23} and μ_{24}, are the membership functions which define the quadrant 2 subdivision. The corresponding parameters "*a*" and "*b*", as seen in equations Eqs. (1) and (2) (Fig. 5a) are as follows: parameter "*a*", which defines the membership functions inclination in this level, is twice the parameter "*a*" of membership functions in the immediately upper level (μ_2); parameter "*b*" is adjusted, so that the average point of transition of the membership functions stays in the middle of quadrant 2.
- μ_{31}, μ_{32}, μ_{33} and μ_{34}, are the membership functions which define the subdivision of the quadrant 3. The parameter "*a*", which defines the membership functions inclination in this level, is twice the parameter "*a*" of the membership functions of the immediately upper level (μ_3). Parameter "*b*" is also adjusted, so that the average point of the membership functions transition stays in the middle of quadrant 3.

From the above discussion, a generic form for the output of a HNFQ system with four levels of hierarchy can be written as

$$\begin{aligned} y = &\sum_{i=1}^{4} \alpha_i \times k_i \times d_i \\ &+ \sum_{i=1}^{4}\sum_{j=1}^{4} \alpha_i \times \alpha_{ij} \times k_{ij} \times d_{ij} \\ &+ \sum_{i=1}^{4}\sum_{j=1}^{4}\sum_{k=1}^{4} \alpha_i \times \alpha_{ij} \times \alpha_{ijk} \times k_{ijk} \times d_{ijk} \\ &+ \sum_{i=1}^{4}\sum_{j=1}^{4}\sum_{k=1}^{4}\sum_{m=1}^{4} \alpha_i \times \alpha_{ij} \times \alpha_{ijk} \times \alpha_{ijkm} \\ &\times k_{ijkm} \times d_{ijkm}, \end{aligned} \tag{7}$$

where

- $\alpha_i, \alpha_{ij}, \alpha_{ijk}, \alpha_{ijkm}$, are the firing levels of the rules in each quad-partition *i*, or *ij*, or *ijk*, or *ijkm*;
- k_i ($k_{ij}, k_{ijk}, k_{ijkm}$), is equal to "1", if the partition *i*, (or *ij*, or *ijk* or *ijkm*) exists and is not subdivided, and is equal to "0" otherwise.
- $d_i, d_{ij}, d_{ijk}, d_{ijkm}$, are the consequents of the existing rules.

In this formula, the simplification obtained from the use of the complementary membership functions ($\mu_{\text{low}} = 1 - \mu_{\text{high}}$) in the defuzzification method of each NFS output has already been considered.

4. Learning algorithm

In the neuro-fuzzy literature, the learning process is generally divided in two parts: the identification of the structure and the adjustments of parameters.

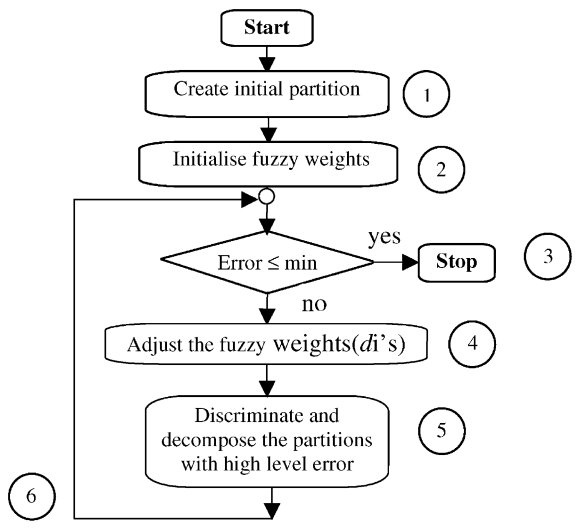
Start
Create initial partition
1
Initialise fuzzy weights
2
yes
Error ≤ min
Stop
3
no
Adjust the fuzzy weights(di's)
4
Discriminate and decompose the partitions with high level error
5
6

Fig. 10. Learning algorithm of HNFQ model.

The HNFQ model follows the same process. However, only one algorithm carries out both learning tasks.

The learning algorithm of HNFQ model is performed in six steps, as described below and illustrated in the flowchart of Fig. 10.

Learning steps:

(1) An initial partitioning is created, dividing the input space into four parts, through two fuzzy sets—*high* and *low*, for each input variable. In this step, the first quadtree cell is created;

(2) The weights d_i (consequents of the first four rules) are initialised with the average of the target values of the output patterns that fall on the quad-partitioning (Fig. 8b) of index i. For example, to calculate the initial value of the weight d_4, all target values that fall in quadrant 4 are summed and then divided by the number of patterns that fall in quadrant 4;

(3) The system total error is then calculated for all the training set, according to the root mean square error expression, given below:

$$E_{\text{rms}} = \sqrt{\frac{1}{L}\sum_{n=1}^{L}(y_n - Y_n)^2}, \qquad (8)$$

where L is the number of patterns in the training set and y_n and Y_n are, respectively, the output value of HNFQ system and the desired output value for the pattern of index "n". When this error is below the desired minimum, the learning process stops; otherwise, the learning process continues with step 4;

(4) The gradient descent method (see Appendix A) adjust the fuzzy weights d_i's;

(5) Each quad-partitioning is evaluated regarding its contribution to the total root mean square error and regarding the acceptable minimum error. Each quad-partitioning with unacceptable error is separated. The evaluation of the error generated for the data set that fall on the partitioning $\boldsymbol{ij}$, for example, is calculated by the following expression:

$$E^{ij}_{\text{rms}} = \sqrt{\frac{1}{L}\sum_{n=1}^{L}\alpha^n_i \times \alpha^n_{ij} \times (y_n - Y_n)^2}, \qquad (9)$$

where α^n_i, and α^n_{ij} are the rules' firing levels for pattern "n", as mentioned at the end of Section 3.2.

For each separated quad-partitioning, a new node in the quadtree tree is allocated. In other words, each separated partitioning is recursively decomposed (divided in 4). In this case, a decomposing process with a fixed partitioning was used. Four new membership functions are then generated, which constitute the four partitions just created. The motivation for including the rule's firing levels (α^n_i and α^n_{ij}) in this formula definition is to measure the contribution of each partition in the total error.

(6) Back to step "3" to continue the learning process.

5. Cases study

The HNFQ system has been evaluated in three benchmarks often found in neuro-fuzzy literature: the sinc(x, y) [29] function approximation, which is a two variable non-linear function, the Mackey Glass chaotic series forecast [13] and the two spirals problem [19]. The next subsections discuss each particular application, present the results obtained by the HNFQ model and compare the performance with other NFS.

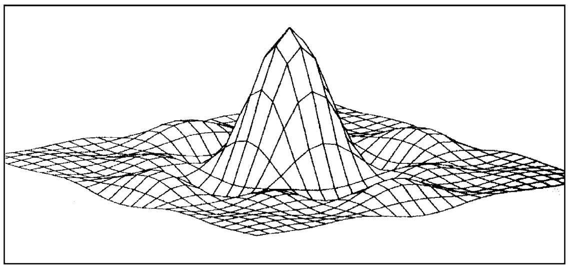
Fig. 11. Sinc function (x, y).

5.1. sinc(x, y) *function approximation*

In this application, the non-linear sinc(x, y) function is approximated by a HNFQ model. Eq. (10) describes the sinc function; its shape, which resembles a Mexican hat, is shown in Fig. 11

$$\operatorname{sinc}(x, y) = \frac{\sin(x)}{x} \cdot \frac{\sin(y)}{y}. \tag{10}$$

A data set of 225 samples was used for training. The inputs (x, y) vary in the interval $[-10, 10]$ and the output $z = \operatorname{sinc}(x, y)$ varies in the interval $[-0.21, 1]$. In the beginning of the training process, the HNFQ model started with a single cell and its structure expanded to minimise the RMS total error. The HNFQ structure evolved as depicted in Figs. 12a–d.

As can be observed from Fig. 12, the HNFQ structure evolved according to the sinc(x, y) function complexity, creating new partitions in regions where the function variations are more accentuated. Fig. 12a illustrates the initial input space partitioning, generated by x and y variables using only a single HNFQ cell. Fig. 12b shows the first partitioning, after the first decomposition, using 5 cells. Figs. 12c and d show the partitioning evolution through the decomposition with 17 and 21 cells, respectively. In total, 21 HNFQ cells have been placed in four levels (1 cell in level 1 + 4 cells in level 2 + 12 cells in level 3 + 4 cells in level 4), with 64 free parameters as the singletons rules consequents.

The final structure of the HNFQ system, at the end of the learning process, is illustrated in Fig. 13.

Following the training process, a set of 625 patterns was used for testing. The total RMS error, for the training and test sets, was 1.5%.

Vuorimaa has applied its FSOM system [29] in the same benchmark application, with similar training and test sets, obtaining errors values of 3.14% and 3.19%, respectively. In the FSOM system, the number of free parameters was 100. This NFS makes use of

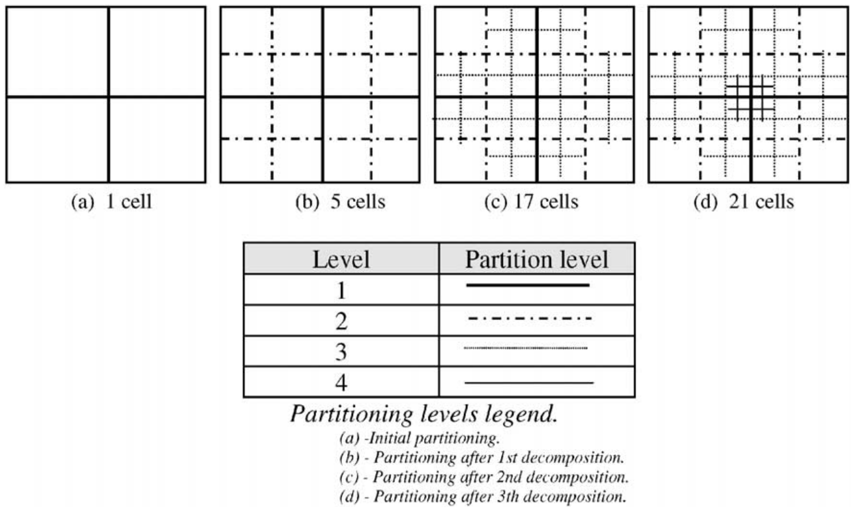
(a) 1 cell
(b) 5 cells
(c) 17 cells
(d) 21 cells
Level
Partition level
1
2
3
4
Partitioning levels legend.
(a) -Initial partitioning.
(b) - Partitioning after 1st decomposition.
(c) - Partitioning after 2nd decomposition.
(d) - Partitioning after 3th decomposition.

Fig. 12. The evolution of the HNFQ structure during the training process.

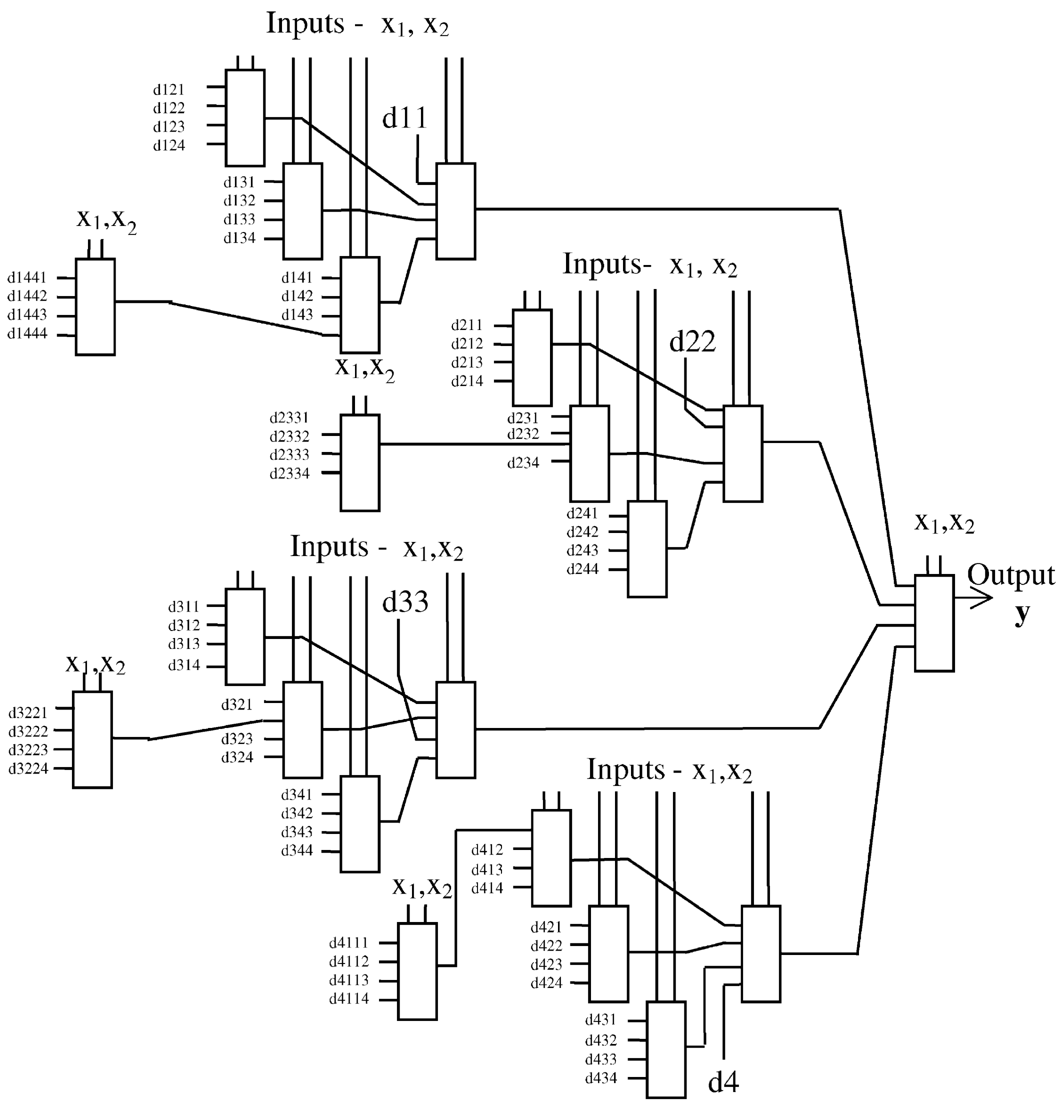

Fig. 13. The HNFQ final structure for the sinc(x, y) function approximation.

Table 2
Comparison of the results obtained by FSOM and HNFQ models in the sinc(x, y) function approximation

Neuro-fuzzy model	RMS error (train. set) (%)	RMS error (tests set) (%)	# Free parameters
FSOM	3.14	3.19	100
HNFQ	1.5	1.5	64

the fuzzy box partitioning (see Fig. 2c), adjusting its antecedent parameters using the LVQ algorithm [29] and the consequent ones using the least square estimator (LSE) algorithm. Table 2 compares the results obtained by the two models: FSOM and HNFQ.

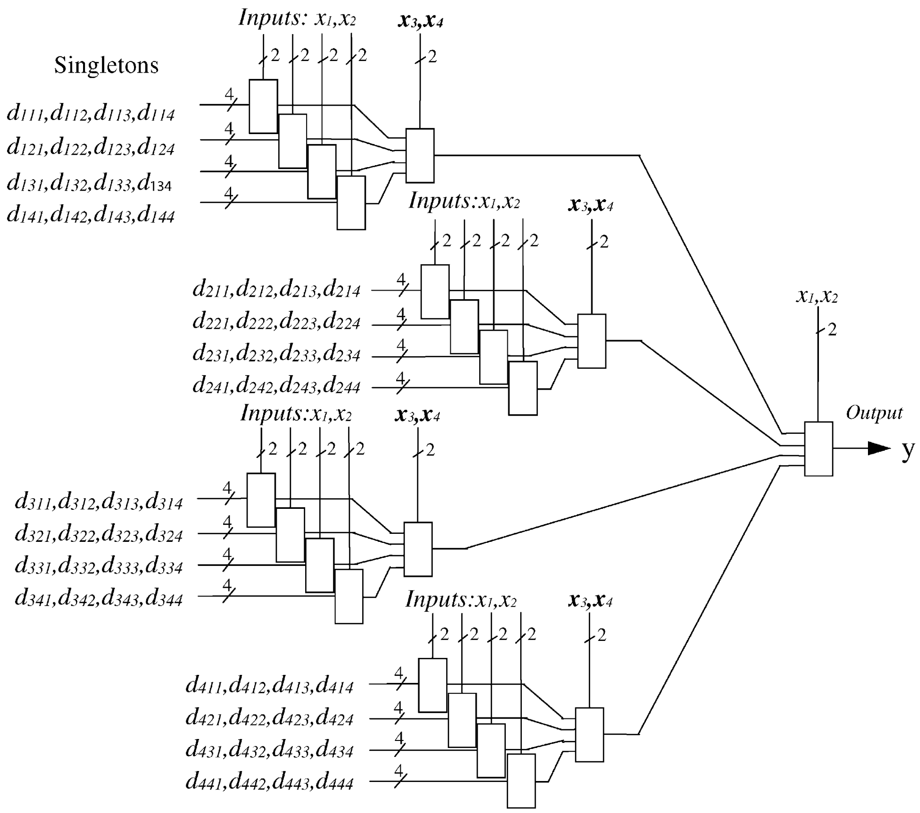

Fig. 14. HNFQ final structure model for the Mackey Glass chaotic series forecast.

5.2. Chaotic series of Mackey Glass

The Mackey Glass chaotic series is one of the most traditional benchmarks in this area. Its expression is given by

$$x'(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t). \qquad (11)$$

In this application, the $x(t-18)$, $x(t-12)$, $x(t-6)$ and $x(t)$ values have been used to forecast the $x(t+6)$ value. The training set was created by numerical integration (Runge–Kutta) with 0.1 steps [13]; for the generation of the time series, values of $x(0)=1.2$ and $t=17$ have been used. One thousand points have been generated, 500 for training and 500 for test.

Since this application has four input parameters, the inputs $(x_1,x_2)=(x(t-18), x(t-12))$ and $(x_3,x_4)=(x(t-6),x(t))$ have been alternated in each level of the HNFQ structure developed during the training phase. Fig. 14 shows the final structure obtained after training. The x_3 and x_4 inputs are shown in boldface to point out the inputs alternation in the architecture. This alternation was carried out automatically at each new level of the structure.

The HNFQ model created in this application presents three levels with 21 (1+4+16) cells and uses 64 free parameters (singletons d_{ijk}). The RMS total error was 0.65% for the test set. Table 3 compares the results obtained by the HNFQ model with the results presented in ANFIS [13] and NEFPROX [26] for this time series.

The ANFIS [13] and NEFPROX [26] systems are neuro-fuzzy models specific for function approximation. The ANFIS system establishes the parameters of the antecedents using a gradient descent algorithm and adjusts its consequent parameters by the LSE algorithm. The NEFPROX [26], on the other hand, uses

Table 3
Comparison between the Mackey–Glass chaotic series simulations for ANFIS, NEFPROX and HNFQ models

Model [Ref.]	RMS error (tests set) (%)	# Free parameters
ANFIS [13]	0.16	104
NEFPROX [26]	3.32	105
HNFQ	0.65	64

an incremental rule learning and adjusts its consequent and antecedent parameters using a kind of modified gradient descent algorithm.

It is important to observe that, although the results obtained by the HNFQ model have been inferior than the results obtained by Jang in its ANFIS model, the number of adjustable parameters used by HNFQ model are almost 60% of the number of adjustable parameters of the ANFIS model. An alternative test to be carried out would be to employ linear combinations of inputs (x_1, x_2, x_3, x_4), instead of using only singletons, for the rules consequences. With this change, the HNFQ model is expected to attain better accuracy with a lesser number of partitioning in this application.

5.3. *The two spirals problem*

This benchmark deals with a classification problem [19]. The problem can be specified as follows: given a certain point in one of the two spirals (Fig. 15), the system must identify to which specific spiral the point belongs to. The data set is composed of 194 patterns, with two inputs and one output. The two inputs correspond to the rectangular co-ordinates (x, y); an output "0" indicates that the point belongs to spiral 1 and an output "1" indicates that the point belongs to spiral 2.

A criteria of 40 : 20 : 40 are usually applied for all reported experiments with this problem. These three numbers indicate the correspondent percentage for the limits of level "0" (spiral 1), an undetermined output, and level "1" (spiral 2). In other words, values between 0 and 0.4 (inclusive) specifies spiral 1; values between 0.4 and 0.6 indicates undefined spiral; and values between 0.6 (inclusive) and 1.0, designates spiral 2.

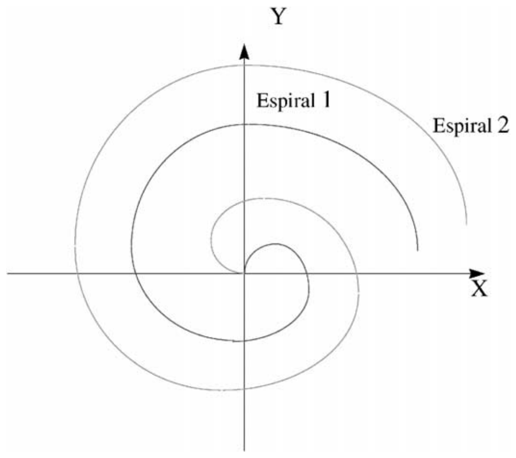

Fig. 15. Two spirals problem.

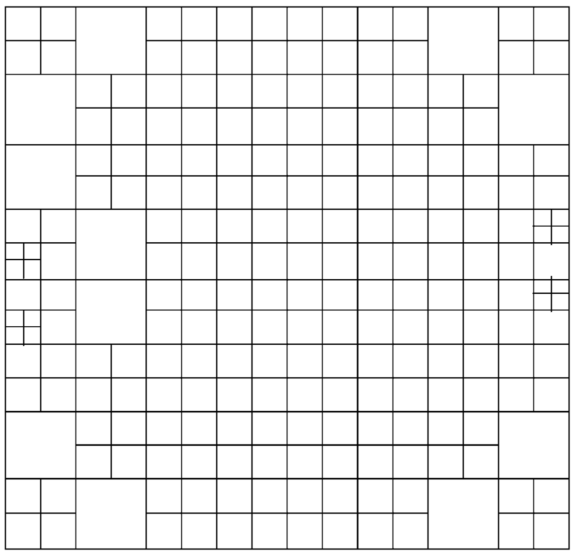

Fig. 16. Final partitioning obtained after training the HNFQ model, for the two spirals problem.

The HNFQ model was trained with 194 patterns. After the training process was completed, the obtained input space partitioning is as shown in Fig. 16. As can be observed, the generated structure reached five levels of subdivisions, with a total of 238 adjustable parameters.

Table 4 below summarises the results obtained with this problem by many authors. Models RN1 and RN2

Table 4
Comparison between the HNFQ and other methods for the two spirals problem

Model	# Epochs	# Parameters	Structure
RN1	20.000	138	2-5-5-5-1
RN2	13.900	281	2-20-10-1
RN3	1.700	—	Cascade correlation
HNFQ	175	238	5 levels

Table 5
Computation performance of the HNFQ model for the three benchmark applications

Benchmark application	Computational time (approximate in s)
sinc(x, y) function	100
Mackey Glass chaotic series	200
Two spiral problem	60

[19] refer to two neural networks trained with back propagation algorithm and with the specific structures described in the last column. The RN3 model corresponds to the results obtained with Falman's cascade correlation network [6]. Finally, the last line of Table 4 exhibits the results obtained with the HNFQ model. All models in Table 4 provided 0% classification error after a certain number of epochs. Therefore, the number of epochs, necessary for the error convergence of each model, was used as the performance criteria. As can be observed, the HNFQ model presents better performance in comparison to the other models. This efficiency results from the automatic generation of the model's structure, in addition to the capability of adjusting its own parameters (a common feature among the other models).

5.4. *Computational performance*

The evaluation of the three benchmarks described in the previous sections has been carried out in an AMD K6 266 MHz machine. The computational time to accomplish the training process is provided in Table 5, for all three applications.

6. Conclusions and future perspectives

This article presented the main ideas about the new hierarchical NFS with the recursive quadtree partitioning. As it has been seen, the recursive partitioning lead to an hierarchical base rule, which is inherent to these types of NFS. In the benchmark applications, the HNFQ has presented good convergence (accuracy), generalisation, and has been able to generate its own structure. There was no need for a random weight initialisation. This leads to a constant replication of the results obtained for the same training parameters (there is nothing random or stochastic in the learning process). Moreover, the knowledge extraction under the format of fuzzy rules and the automatic generation of its structure are the key aspects of this system.

Some parameters are, however, important for achieving good generalisation performance. One of the most important is the decomposition rate (δ), which is a constant parameter that prevents the system's structure from growing indefinitely. It is an a dimensional parameter and acts as a limiting factor for the decomposition process. A very low value can result in a very big structure, compromising the generalisation capability. On the other hand, if a large value is chosen, the structure might be too small to learn the patterns with the desired accuracy.

A kind of input/output hierarchy has also been used in the fuzzy models described by Holve [10,11]. In these models, called Hierarchical Fuzzy Associative Memory (HIFAM), small fuzzy models (mini-systems) are placed in a binary tree structure, where its outputs are used as inputs for others fuzzy models, creating an hierarchical structure.

The main difference between these two fuzzy hierarchical models (HNFQ and HIFAM) is related to the interpretability of the rules obtained in each system. In the HNFQ model, which uses hierarchy through the consequents, the rules are more interpretable, since the inputs are used directly in the rule's antecedent. This is not the case of the HIFAM model, which can use, in the antecedent, intermediate inputs (outputs of previous mini-systems in the hierarchy), causing the resultant rules to be non-interpretable.

This basic HNFQ is being extended, incorporating new features. A new two phase learning algorithm has been implemented, where the consequents are adjusted by the LSE and the rules' antecedents are trained with the resilient back propagation (Rprop). This new approach uses adaptive quadtree partitioning, since the rule's antecedents are also adjusted. This new learning algorithm has provided very good results in various applications such as time series forecasting and classification. Additionally, a new hierarchical model, based on the BSP partitioning, has also been developed, providing good preliminary results. Because the BSP partitioning method successively divides the space in two regions (as opposed to four regions in the quadtree partitioning), it usually creates a smaller neuro-fuzzy structure, resulting in better generalisation performance. These new experiments will be reported in a following paper.

Two other important issues that will be investigated are the attribute selection methods used in the decision tree induction algorithms, such as ID3 and C4.5 by Quinlan [27,28], as well as performance comparison with fuzzy decision tree algorithms—Marsala [1,23].

In the ID3 algorithm, for example, a simple tree can be generated by suitable selection of attributes. An information-based heuristic is used to select these attributes. The heuristic selects the attribute providing the highest information gain, i.e., the attribute which minimises the information needed in the resulting subtrees to classify the elements (patterns). In the HNFQ systems, on the other hand, an attribute pair which minimises the total root mean square error of the system was chosen.

Appendix A Gradient descent calculation

The output of a HNFQ cell is given by the following equation:

$$y = \alpha_1 d_1 + \alpha_2 d_2 + \alpha_3 d_3 + \alpha_4 d_4, \tag{A1.1}$$

where

$$\alpha_1(x_h, x_v) = \mu_1(x_h)\mu_3(x_v)$$
$$= [1 - \mu_2(x_h)][1 - \mu_4(x_v)], \qquad \text{(A1.2)*}$$

$$\alpha_2(x_h, x_v) = \mu_1(x_h)\mu_4(x_v)$$
$$= [1 - \mu_2(x_h)]\,\mu_4(x_v), \qquad \text{(A1.3)*}$$

$$\alpha_3(x_h, x_v) = \mu_2(x_h)\mu_3(x_v)$$
$$= \mu_2(x_h)[1 - \mu_4(x_v)], \qquad \text{(A1.4)*}$$

$$\alpha_4(x_h, x_v) = \mu_2(x_h)\mu_4(x_v) = \mu_2(x_h)\mu_4(x_v), \qquad \text{(A1.5)*}$$

$$\mu_2(x_h) = \mathrm{sig}(a_h(x_h - b_h))$$
$$= 1/[1 + \exp(-a_h(x_h - b_h))], \tag{A1.6}$$

$$\mu_1(x_h) = 1 - \mu_2(x_h), \tag{A1.7}$$

$$\mu_4(x_v) = \mathrm{sig}(a_v(x_v - b_v))$$
$$= 1/[1 + \exp(-a_v(x_v - b_v))], \tag{A1.8}$$

$$\mu_3(x_v) = 1 - \mu_4(x_v), \tag{A1.9}$$

d_1, d_2, d_3, d_4 are the values of the consequents singletons or the output of previous levels, x_h, x_v are the input variables (horizontal and vertical), α_1, α_2, α_3, and α_4 are the firing level of the four rules, μ_1, μ_2, μ_3 and μ_4 are the activation functions of the HNFQ cell, a_h and a_v are the parameters that define the inclination of the μ_1, μ_2, μ_3 and μ_4 membership functions, b_h and b_v are the parameters that define the inflexion point of the μ_1, μ_2, μ_3 and μ_4 membership functions.

In the HNFQ model, the gradient descent method is used to minimise the mean square error. Therefore, it is necessary to calculate the error gradient related to the adjustable parameters of the HNFQ cell: a_h, a_v, b_h, b_v, d_1, d_2, d_3 and d_4.

Defining the mean square error as

$$\varepsilon = \tfrac{1}{2}(y - y^d)^2, \tag{A1.10}$$

where "y" represents the system's output and y^d the target output value. The error (ε) partial derivative in relation to a generic parameter "w" (of which the output y depends on) is given by

$$\frac{\partial \varepsilon}{\partial w} = \frac{\partial \varepsilon}{\partial y}\frac{\partial y}{\partial w}, \tag{A1.11}$$

$$\frac{\partial \varepsilon}{\partial w} = (y - y^d)\frac{\partial y}{\partial w}, \tag{A1.12}$$

$$\frac{\partial \varepsilon}{\partial w} = E\,\frac{\partial y}{\partial w}, \tag{A1.13}$$

where $E = (y - y^d)$.

Eq. (A1.13) shows that it is necessary to calculate the partial derivative of the output y in relation to each adjustable parameter of each HNFQ cell ($a_{\rm h}$, $a_{\rm v}$, $b_{\rm h}$ and $b_{\rm v}$).

Let "$y^{\text{cel\_}k}$" be the output of cell "k" in the HNFQ structure, as shown in Fig. 17 below.

Therefore, $y^{\text{cel\_}k}$ is calculated as

$$y^{\text{cel\_}k} = (\alpha_{1k}\, d_{1k} + \alpha_{2k}\, d_{2k} + \alpha_{3k}\, d_{3k} + \alpha_{4k}\, d_{4k}), \tag{A1.14}$$

where α_{jk} is the firing level of rule j of cell "k", d_{1k} is the consequent *singleton* 1 of cell "k", d_{2k} is the output y^r of cell "r", d_{3k} is the output y^s of cell "s" and d_{4k} the consequent *singleton* 4 of cell "k".

Then, the contribution of the output of cell "k" ($y^{\text{cel\_}k}$) to the system's output y is given by

$$H^{\rm K} = \Pi\alpha_k(y^{\text{cel\_}k}) \tag{A1.15}$$

By substituting Eq. (A1.14) in (A1.15), the equation can be rewritten as follows

$$H^{\rm K} = \Pi\alpha_k(\alpha_{1k}d_{1k} + \alpha_{2k}d_{2k} + \alpha_{3k}d_{3k} + \alpha_{4k}d_{4k}), \tag{A1.16}$$

where factor $\Pi\alpha_k$ corresponds to the product of all firing levels α_i from the cells that link cell "k" to the output y.

In the example of Fig. 17, this value is given by

$$\Pi\alpha_k = \alpha_{1u}\alpha_{2t}, \tag{A1.17}$$

where α_{1u} is the firing level of rule 1 of cell "u" and α_{2t} is the firing level of rule 2 of cell "t".

Therefore, the relation between the total output "y" and the output of cell "k" $y^{\text{cel\_}k}$ is

$$y = H^{\rm K} + \Psi, \tag{A1.18}$$

where Ψ represents the sum of all other influences on the output "y", except from cell "k". By substituting Eqs. (A1.15) or (A1.16) in Eq. (A1.18), "y" can be calculated as

$$y = \Pi\alpha_k(y^{\text{cel\_}k}) + \Psi, \tag{A1.19}$$

$$y = \Pi\alpha_k(\alpha_{1k}d_{1k} + \alpha_{2k}d_{2k} + \alpha_{3k}d_{3k} + \alpha_{4k}\, d_{4k}) + \Psi. \tag{A1.20}$$

A.1. Partial derivatives of the antecedent parameters

The partial derivatives of the output "y" in relation to each parameter $a_{\rm h}^k$, $a_{\rm v}^k$, $b_{\rm h}^k$ or $b_{\rm v}^k$ of cell "k" are then given by the following equations:

$$\partial y/\partial a_{\rm h}^k = \partial(\Pi\alpha_k y^{\text{cel\_}k} + \Psi)/\partial a_{\rm h}^k, \tag{A1.21}$$

$$\partial y/\partial a_{\rm v}^k = \partial(\Pi\alpha_k y^{\text{cel\_}k} + \Psi)/\partial a_{\rm v}^k, \tag{A1.22}$$

$$\partial y/\partial b_{\rm h}^k = \partial(\Pi\alpha_k y^{\text{cel\_}k} + \Psi)/\partial b_{\rm h}^k, \tag{A1.23}$$

$$\partial y/\partial b_{\rm v}^k = \partial(\Pi\alpha_k y^{\text{cel\_}k} + \Psi)/\partial b_{\rm v}^k. \tag{A1.24}$$

Since Ψ and the factor $\Pi\alpha_k$ are independent from $a_{\rm h}^k$, (and $a_{\rm v}^k, b_{\rm h}^k {\rm e} b_{\rm v}^k$), Eq. (A.21) can be rewritten as follows

$$\partial y/\partial a_{\rm h}^k = \Pi\alpha_k \partial y^{\text{cel\_}k}/\partial a_{\rm h}^k. \tag{A1.25}$$

From Eq. (A1.13), the partial derivative of the square error "ε" in relation to $a_{\rm h}^k$ parameter can then be written as

$$\partial\varepsilon/\partial a_{\rm h}^k = E\Pi\alpha_k \partial y^{\text{cel\_}k}/\partial a_{\rm h}^k. \tag{A1.26}$$

By using Eqs. (A1.2)–(A1.5), the following equations apply:

$$\begin{aligned}\alpha_{1k}(x_{\rm h},x_{\rm v}) &= \mu_{1k}(a_{\rm h},b_{\rm h},x_{\rm h})\mu_{3k}(a_{\rm v},b_{\rm v},x_{\rm v})\\ &= [1-\mu_{2k}(a_{\rm h},b_{\rm h},x_{\rm h})][1-\mu_{4k}(a_{\rm v},b_{\rm v},x_{\rm v})],\end{aligned} \tag{A1.27}$$

$$\begin{aligned}\alpha_{2k}(x_{\rm h},x_{\rm v}) &= \mu_{1k}(a_{\rm h},b_{\rm h},x_{\rm h})\mu_{4k}(a_{\rm v},b_{\rm v},x_{\rm v})\\ &= [1-\mu_{2k}(a_{\rm h},b_{\rm h},x_{\rm h})]\mu_{4k}(a_{\rm v},b_{\rm v},x_{\rm v}),\end{aligned} \tag{A1.28}$$

$$\begin{aligned}\alpha_{3k}(x_{\rm h},x_{\rm v}) &= \mu_{2k}(a_{\rm h},b_{\rm h},x_{\rm h})\mu_{3k}(a_{\rm v},b_{\rm v},x_{\rm v})\\ &= \mu_{2k}(a_{\rm h},b_{\rm h},x_{\rm h})[1-\mu_{4k}(a_{\rm v},b_{\rm v},x_{\rm v})],\end{aligned} \tag{A1.29}$$

$$\begin{aligned}\alpha_{4k}(x_{\rm h},x_{\rm v}) &= \mu_{2k}(a_{\rm h},b_{\rm h},x_{\rm h})\mu_{4k}(a_{\rm v},b_{\rm v},x_{\rm v})\\ &= \mu_{2k}(a_{\rm h},b_{\rm h},x_{\rm h})\mu_{4k}(a_{\rm v},b_{\rm v},x_{\rm v}).\end{aligned} \tag{A1.30}$$

As explained in Section 3.1, the membership functions of HNFQ cell have been implemented as sigmoid

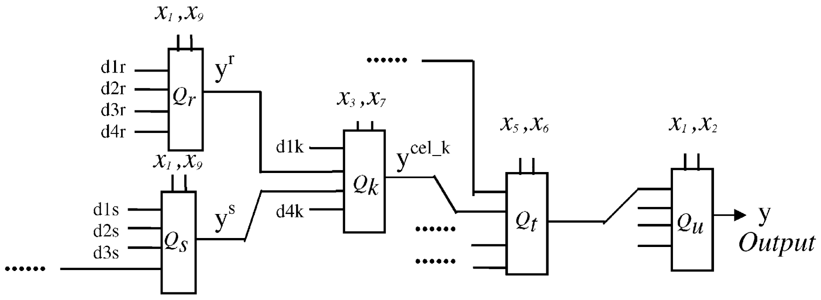

Fig. 17. Detail of a HNFQ system, showing the output of a specific cell "k".

functions (μ_2 and μ_4) and their complements (μ_1 and μ_3). Since the sigmoid function is given by

$$\mathrm{sig}(z) = 1/[1+\exp(-z)]. \tag{A1.31}$$

Then

$$\partial\,\mathrm{sig}(z)/\partial z = \mathrm{sig}(z)[1-\mathrm{sig}(z)]. \tag{A1.32}$$

By Eqs. (A1.27)–(A1.30) the partial derivatives of the rules' firing levels of cell "k" in relation to the parameters a_{h}^k, are the following:

$$\partial\alpha_{1k}/\partial a_{\mathrm{h}}^k = \partial(\mu_{1k}(a_{\mathrm{h}},b_{\mathrm{h}},x_{\mathrm{h}})\mu_{3k}(a_{\mathrm{v}},b_{\mathrm{v}},x_{\mathrm{v}})), \tag{A1.33}$$

$$\partial\alpha_{2k}/\partial a_{\mathrm{h}}^k = \partial(\mu_{1k}(a_{\mathrm{h}},b_{\mathrm{h}},x_{\mathrm{h}})\mu_{4k}(a_{\mathrm{v}},b_{\mathrm{v}},x_{\mathrm{v}})), \tag{A1.34}$$

$$\partial\alpha_{3k}/\partial a_{\mathrm{h}}^k = \partial(\mu_{2k}(a_{\mathrm{h}},b_{\mathrm{h}},x_{\mathrm{h}})\mu_{3k}(a_{\mathrm{v}},b_{\mathrm{v}},x_{\mathrm{v}})), \tag{A1.35}$$

$$\partial\alpha_{4k}/\partial a_{\mathrm{h}}^k = \partial(\mu_{2k}(a_{\mathrm{h}},b_{\mathrm{h}},x_{\mathrm{h}})\mu_{4k}(a_{\mathrm{v}},b_{\mathrm{v}},x_{\mathrm{v}})). \tag{A1.36}$$

By substituting Eq. (A1.32) in the previous equations and in view of Eq. (A1.27)–(A1.30) the partial derivatives of the firing levels α_{ik}'s in relation to the parameter a_{h}^k are given by

$$\partial\alpha_{1k}/\partial a_{\mathrm{h}}^k = \mu_{2k}(a_{\mathrm{h}},b_{\mathrm{h}},x_{\mathrm{h}})\alpha_{1k}(b_{\mathrm{h}}^k-x_{\mathrm{h}}), \tag{A1.37}$$

$$\partial\alpha_{2k}/\partial a_{\mathrm{h}}^k = \mu_{2k}(a_{\mathrm{h}},b_{\mathrm{h}},x_{\mathrm{h}})\alpha_{2k}(b_{\mathrm{h}}^k-x_{\mathrm{h}}), \tag{A1.38}$$

$$\partial\alpha_{3k}/\partial a_{\mathrm{h}}^k = -\mu_{1k}(a_{\mathrm{h}},b_{\mathrm{h}},x_{\mathrm{h}})\alpha_{3k}(b_{\mathrm{h}}^k-x_{\mathrm{h}}), \tag{A1.39}$$

$$\partial\alpha_{4k}/\partial a_{\mathrm{h}}^k = -\mu_{1k}(a_{\mathrm{h}},b_{\mathrm{h}},x_{\mathrm{h}})\alpha_{4k}(b_{\mathrm{h}}^k-x_{\mathrm{h}}). \tag{A1.40}$$

For the parameters $a_{\mathrm{v}}^k, b_{\mathrm{h}}^k, \mathrm{e}b_{\mathrm{v}}^k$, the partial derivatives are calculated in a similar way.

Finally, by using the four last equations and Eq. (A1.26), the partial derivatives of the error value "ε" in relation to the a_{h}^k parameter in the antecendent of cell "k" are given by

$$\begin{aligned}\partial\varepsilon/\partial a_{\mathrm{h}}^k = {}& E\Pi\alpha_k[\mu_{2k}\alpha_{1k}d_{1k}+\mu_{2k}\alpha_{2k}d_{2k}-\mu_{1k}\alpha_{3k}d_{3k}\\ &-\mu_{1k}\alpha_{4k}d_{4k}](b_{\mathrm{h}}^k-x_{\mathrm{h}}).\end{aligned} \tag{A1.41}$$

In analogy,

$$\begin{aligned}\partial\varepsilon/\partial b_{\mathrm{h}}^k = {}& E\Pi\alpha_k[-\mu_{2k}\alpha_{1k}d_{1k}-\mu_{2k}\alpha_{2k}d_{2k}\\ &+\mu_{1k}\alpha_{3k}d_{3k}+\mu_{1k}\alpha_{4k}d_{4k}](a_{\mathrm{h}}^k),\end{aligned} \tag{A1.42}$$

$$\begin{aligned}\partial\varepsilon/\partial a_{\mathrm{v}}^k = {}& E\Pi\alpha_k[\mu_{4k}\alpha_{1k}d_{1k}-\mu_{3k}\alpha_{2k}d_{2k}\\ &+\mu_{4k}\alpha_{3k}d_{3k}-\mu_{3k}\alpha_{4k}d_{4k}](b_{\mathrm{v}}^k-x_{\mathrm{v}}),\end{aligned} \tag{A1.43}$$

$$\begin{aligned}\partial\varepsilon/\partial b_{\mathrm{v}}^k = {}& E\Pi\alpha_k[-\mu_{4k}\alpha_{1k}d_{1k}+\mu_{3k}\alpha_{2k}d_{2k}\\ &-\mu_{4k}\alpha_{3k}d_{3k}+\mu_{3k}\alpha_{4k}d_{4k}](a_{\mathrm{v}}^k).\end{aligned} \tag{A1.44}$$

A.2. Partial derivatives of the consequent parameters

From Eq. (A1.20), the partial derivatives of the output "y" in relation to each parameter d_j ($j = 1$, 2, 3 and 4), of the cell "k" are given by the following equations:

$$\partial y/\partial d_j^k = \Pi\alpha_k\partial y^{\mathrm{cel\_}k}/\partial d_j^k. \tag{A1.45}$$

From Eq. (A1.13), the partial derivative of the square error "ε" in relation to each of the previous parameters can then be written as

$$\partial\varepsilon/\partial d_j^k = E\Pi\alpha_k \partial y^{\text{cel\_}k}/\partial d_j^k \quad (j = 1,\ 2,\ 3 \text{ and } 4). \tag{A1.46}$$

By using Eq. (A1.1), it follows that

$$\partial y^{\text{cel\_}k}/\partial d_j^k = \alpha_{jk} \quad (j = 1,\ 2,\ 3 \text{ and } 4). \tag{A1.47}$$

Finally, as can be seen from Eqs. (A1.27)–(A1.30) and Eq. (A1.47), the partial derivatives of the error value "ε" in relation to the consequent parameters of the cell "k" are given by

$$\partial\varepsilon/\partial d_1^k = E\Pi\alpha_k(\mu_{1k}\mu_{3k}), \tag{A1.48}$$

$$\partial\varepsilon/\partial d_2^k = E\Pi\alpha_k(\mu_{1k}\mu_{4k}), \tag{A1.49}$$

$$\partial\varepsilon/\partial d_3^k = E\Pi\alpha_k(\mu_{2k}\mu_{3k}), \tag{A1.50}$$

$$\partial\varepsilon/\partial d_4^k = E\Pi\alpha_k(\mu_{2k}\mu_{4k}). \tag{A1.51}$$

References

[1] B. Bouchon-Meunier, C. Marsala, Learning fuzzy decision rules, in: J.C. Bezdec, D. Dubois, H. Prade (Eds.), Fuzzy Sets in Approximate Reasoning and Information Systems, ISBN 0-7923-8584-5, 1999, pp. 279–304.

[2] M. Brown, K.M. Bossley, D.J. Mills, C.J. Harris, High dimensional neurofuzzy systems: overcoming the curse of dimensionality, Proceedings of the International Joint Conference on Fuzzy Systems and the 2nd International Fuzzy Engineering Symposium Yokohama, Japan, March 1995, pp. 2139–2146.

[3] N. Chin, S. Feiner, Near real-time shadow generation using BSP trees, Computer Graphics, SIGGRAPH '89 Proceedings, vol. 23(3), July 1989, pp. 99–106.

[4] Y. Chrysanthou, M. Slater, Computing dynamic changes to BSP trees. Computer Graphics Forum, EUROGRAPHICS '92 Proceedings, vol. 11(3), September 1992, pp. 321–332.

[5] P.K. Dash, G. Ramakrishna, A.C. Liew, S. Rahman, Fuzzy neural networks for time-series forecasting of electric load, IEE Proc. Gener. Transm. Distrib. 142 (5) (1995) 535–545.

[6] S. Fahlman, C. Lebiere, The cascade-correlation learning architecture, Technical Report CMU-CS-90-100, Carnegie Mellon University, Pittsburgh, PA, August 1991.

[7] R.A. Finkel, J.L. Bentley, Quad trees, a data structure for retrieval on composite keys, Acta Inform. 4 (1974) 1–9.

[8] S.K. Halgamuge, M. Glesner, Neural networks in designing fuzzy systems for real world applications, Fuzzy Sets and Systems 65 (1994) 1–12.

[9] S. Haykin, Neural Networks—A Comprehensive Foundation, Macmillan College Publishing Company, Inc., New York, 1998.

[10] R. Holve, Rule generation for hierarchical fuzzy systems, Proceedings of the Annual Conference of the North American Fuzzy Information Processing Society—NAFIPS, September 1997, NY, USA, pp. 444–449.

[11] R. Holve, Investigation of automatic rule generation for hierarquical fuzzy systems, Proceedings of Fuzzy IEEE, 1998 IEEE World Congress On Computational Intelligence, May 4–9, Anchorage, Alaska, pp. 973–978.

[12] M. Iskarous, K. Kawamura, Intelligent Control Using a Neuro-Fuzzy Network, Center for Intelligent Systems, Vanderbilt University, Nashville, USA.

[13] J.S.R. Jang, ANFIS: adaptive-network-based fuzzy inference system, IEEE Trans. Systems Man Cybernet. 23 (3) (1993) 665–685.

[14] J.S.R. Jang, C.T. Sun, Neuro-fuzzy modeling and control, Proceedings of the IEEE, March 1995.

[15] B. Kosko, Neural Networks and Fuzzy Systems, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1992.

[16] R. Kruse, D. Nauck, Choosing appropriate neuro-fuzzy models, Proceedings of the EUFIT'94, Aachen, 1994, pp. 552–557.

[17] R. Kruse, D. Nauck, Learning methods for fuzzy systems, Proceedings of the 3rd German GI-Workshop 'Neuro-Fuzzy Systeme', Darmstadt, Germany, 1995.

[18] R. Kruse, D. Nauck , NEFCLASS—a neuro-fuzzy approach for the classification of data, Proceedings of the 1995 ACM Symposium on Applied Computing, Nashville.

[19] K.J. Lang, M.J. Witbrock, Learning to tell two spirals apart, Proceedings of the 1988 Connectionist Models Summer School, Morgan Kaufmann, 1988.

[20] K.M. Lee, D.H. Kwak, H. Leekwang, Tuning of fuzzy models by neural networks, Fuzzy Sets and Systems 76 (1995) 47–61.

[21] C.T. Lin, C.S.G. Lee, Neural-network-based fuzzy logic control and decision system, IEEE Trans. Comput. 40 (12) (1991) 1320–1336.

[22] C.T. Lin, C. Lin, C.S.G. Lee, Fuzzy adaptive learning control network with on-line neural learning, Fuzzy Sets and Systems 71 (1995) 25–45.

[23] C. Marsala, Apprentissage inductif en présence de données imprécises: construction et utilisation d'arbres de décision flous, Thèse de Doctorat de l'Universit' de Paris, 1998.

[24] J.M. Mendel, Fuzzy logic systems for engineering: a tutorial, Proc. IEEE 83 (3) (1995) 345–377.

[25] D. Nauck, A fuzzy perceptron as a generic model for neuro-fuzzy approaches, Proceedings of the Fuzzy-Systeme'94, 2nd GI-Workshop, Munich, Siemens Corporation, October, 1994.

[26] D. Nauck, R. Kruse, Neuro-fuzzy systems for function approximation, Faculty of Computer Science, Neural and Fuzzy Systems, Otto-von-Guericke University of Magdeburg.

[27] J.R. Quinlan, Simplifying decision trees, Internat. J. Man–Machine Stud. 27 (1987) 221–234.

[28] J.R. Quinlan, (1993). C4.5: programs for machine learning, Morgan Kaufmann, San Mateo.

[29] P. Vuorimaa, Fuzzy self-organizing map, Fuzzy Sets and Systems 66 (1994) 223–231.