ISSN
Number 1 | November 2012

COMPUTATIONAL INTELLIGENCE SERIES

Inverted Hierarchical Neuro-Fuzzy BSP System: A Novel Neuro-Fuzzy Model for Pattern Classification and Rule Extraction in Databases

Laercio Brito Gonçalves
Marley Maria Bernardes Rebuzzi Vellasco
Marco Aurélio Cavalcanti Pacheco
Flavio Joaquim de Souza

Laboratório
Inteligência Computacional Aplicada

COMPUTATIONAL INTELLIGENCE SERIES

Number 1 | November 2012

Inverted Hierarchical Neuro-Fuzzy BSP System: A Novel Neuro-Fuzzy Model for Pattern Classification and Rule Extraction in Databases

Laercio Brito Gonçalves
Marley Maria Bernardes Rebuzzi Vellasco
Marco Aurélio Cavalcanti Pacheco
Flavio Joaquim de Souza

CREDITS
Publisher:
MAXWELL / LAMBDA-DEE
Sistema Maxwell / Laboratório de Automação de Museus, Bibliotecas Digitais e Arquivos
http://www.maxwell.lambda.ele.puc-rio.br/
Organizer:
Marley Maria Bernardes Rebuzzi Vellasco
Cover:
Ana Cristina Costa Ribeiro

Inverted Hierarchical Neuro-Fuzzy BSP System: A Novel Neuro-Fuzzy Model for Pattern Classification and Rule Extraction in Databases

Laercio Brito Gonçalves, Marley Maria Bernardes Rebuzzi Vellasco, *Member, IEEE*, Marco Aurélio Cavalcanti Pacheco, and Flavio Joaquim de Souza

***Abstract*—This paper introduces the Inverted Hierarchical Neuro-Fuzzy BSP System (HNFB^{-1}), a new neuro-fuzzy model that has been specifically created for record classification and rule extraction in databases. The HNFB^{-1} is based on the Hierarchical Neuro-Fuzzy Binary Space Partitioning Model (HNFB), which embodies a recursive partitioning of the input space, is able to automatically generate its own structure, and allows a greater number of inputs. The new HNFB^{-1} allows the extraction of knowledge in the form of interpretable fuzzy rules expressed by the following: *If x is A and y is B, then input pattern belongs to class Z*. For the process of rule extraction in the HNFB^{-1} model, two fuzzy evaluation measures were defined: 1) *fuzzy accuracy* and 2) *fuzzy coverage*. The HNFB^{-1} has been evaluated with different benchmark databases for the classification task: Iris Dataset, Wine Data, Pima Indians Diabetes Database, Bupa Liver Disorders, and Heart Disease. When compared with several other pattern classification models and algorithms, the HNFB^{-1} model has shown similar or better classification performance. Nevertheless, its performance in terms of processing time is remarkable. The HNFB^{-1} converged in less than one minute for all the databases described in the case study.**

***Index Terms*—Binary space partitioning (BSP), neuro-fuzzy systems, pattern classification, rule extraction.**

I. Introduction

NEURO-FUZZY [1] are hybrid systems that combine the learning capacity of neural nets [2] with the linguistic interpretability power of fuzzy inference systems [3]. These systems have been evaluated quite intensively for the pattern recognition task. This is mainly due to a number of factors: the applicability of the learning algorithms developed for neural nets; the possibility of promoting implicit and explicit knowledge integration; and the possibility of extracting knowledge in the form of fuzzy rules. Most of the existing neuro-fuzzy systems, however, present limitations with regard to the number of inputs allowed and/or to the limited (or nonexistent) form to create their own structure and rules [4]–[6].

The Hierarchical Neuro-Fuzzy Binary Space Partitioning Systems (HNFB) [7]–[9] allow a greater number of inputs and, in addition, are able to generate their own structure, thus automatically creating their rule base. These models make use of a recursive partitioning method that successively divides the input space into two regions. This recursive partitioning may be represented by a binary tree that illustrates the successive hierarchical subdivisions of the input space. It has been shown by Brown and Harris [10] that a rule base using a hierarchical structure leads to a linear growth in the number of rules.

The original HNFB created by de Souza [7], [8], however, is not ideal for pattern classification applications for two reasons. The first reason is because the HNFB model has only one output, and in order to use it as a classifier, it is necessary to create different range values at the output, each range representing a particular class. This range criterion may have a negative influence on the system's performance, since ranges that are defined as being contiguous in the model may be attempting to represent totally disjoint classes in the input space. The second reason is because the original HNFB model makes use of the Takagi–Sugeno [11] inference method, which reduces the rule base interpretability.

The main objective of this work was then to create a new hierarchical neuro-fuzzy binary space partitioning (BSP) model dedicated to pattern classification and rule extraction, called the Inverted HNFB or HNFB^{-1}, which is able to extract classification rules such as the following: If x is A and y is B, then input pattern belongs to class Z. This new hierarchical neuro-fuzzy model is denominated *inverted* because it applies the learning process of the original HNFB to generate the model's structure and then inverts it in order to validate the results and extract the rule base.

This paper has been divided into four additional sections. Section II summarizes the original HNFB system, describing its architecture, its basic cell, and the learning algorithm. Section III introduces the new HNFB^{-1} model with a description of its basic cell and its architecture. Section III also describes the rule extraction process, presenting the definition of *fuzzy accuracy* and *fuzzy coverage* used to evaluate the generated

Manuscript received August 1, 2003; revised March 10, 2004 and July 12, 2004. This work was supported in part by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), DF, Brazil and FAPERJ (Fundação de Amparo a Pesquisa do Rio de Janeiro), Rio de Janeiro, Brazil. This paper was recommended by Associate Editor A. Bouzerdoum.

L. B. Gonçalves, M. M. B. R. Vellasco, and M. A. C. Pacheco are with the Electrical Engineering Department, Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro 22453-900, Brazil (e-mail: laercio@ele.puc-rio.br; marley@ele.puc-rio.br; marco@ele.puc-rio.br).

F. J. de Souza is with the Computing and Systems Engineering Department, State University of Rio de Janeiro (UERJ), Rio de Janeiro 20559-900, Brazil (e-mail: fjsouza@eng.uerj.br).

Digital Object Identifier 10.1109/TSMCC.2004.843220

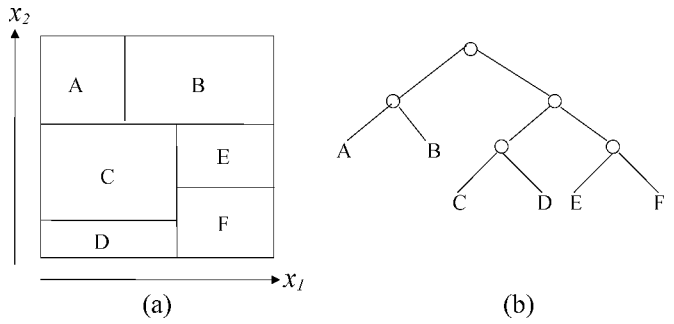

Fig. 1. (a) BSP partitioning. (b) BSP tree referring to BSP partitioning.

rules. Section IV presents the case studies, describing the benchmark databases employed and the performance obtained with the HNFB^{-1} model in the classification task. Finally, the conclusions that have been drawn from this study are presented in Section V, together with model extensions that are being undertaken.

II. Hierarchical Neuro-Fuzzy BSP Model

Current neuro-fuzzy systems present two basic limitations: the low number of inputs with which they can efficiently work and the limited (and in some cases, nonexistent) process through which they create their own structure and rules. The first limitation is related to the "curse of dimensionality" generated by the large number of rules that results from the partitioning of the input space in a grid form.[1]

The HNFB makes use of a hierarchical partitioning called binary space partitioning (BSP), which divides the space successively, in a recursive way, in two regions. Fig. 1 presents an example of a BSP partitioning in a two-dimensional (2-D) space, defined by variables x_1 and x_2.

BSP partitioning is flexible and minimizes the problem of the exponential growth of rules, since it only creates new rules locally according to the training set. Its main advantage is that it automatically builds its own structure. This type of partitioning employs recursive processes in its generation, which results in models with a hierarchy in their structure and, consequently, hierarchical rules.

A. Basic Neuro-Fuzzy BSP Cell

An HNFB cell can be seen as a neuro-fuzzy minisystem that performs binary fuzzy partitioning of the input space. The HNFB cell generates a precise (crisp) output after a defuzzification process. Fig. 2(a) illustrates the cell's structure. In this cell, x represents the input variable; ρ and μ are the membership functions *low* and *high*, respectively, that generate the antecedents of the two fuzzy rules; and d_i is the consequent of rule i. Fig. 2(b) presents a simplified schematic of the cell.

The linguistic interpretation of the mapping implemented by the HNFB cell is given by the following set of rules:

[1]A neuro-fuzzy system with five input variables, each with its universe of discourse divided into four fuzzy sets, may obtain a total of 1024 rules (4^5). If the number of inputs is raised to 20, and the same division of the universes of discourse is used, the result is an unmanageable total of 1.099.511.627.776 rules (4^{20}).

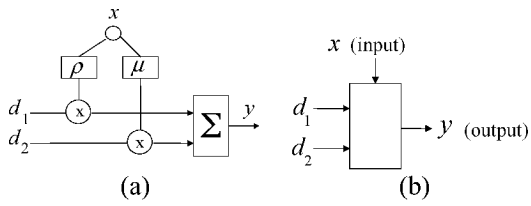

Fig. 2. (a) Neuro-fuzzy BSP cell. (b) Neuro-fuzzy BSP cell schematic symbol.

Rule 1: If $x \in \rho$, then $y = d_1$.
Rule 2: If $x \in \mu$ then $y = d_2$.

Each rule corresponds to one of the two partitions generated by BSP partitioning. When the inputs occur in partition 1, it is rule 1 that has a higher firing level. When they occur in partition 2, it is rule 2 that has a higher firing level. Each partition can in turn be subdivided into two parts by means of another HNFB cell. The profiles of membership functions ρ and μ are presented by the following:

$$\mu(x) = \frac{1}{1 + \exp\left[-(ax - b)\right]} \tag{1}$$

$$\rho(x) = 1 - \mu(x) \tag{2}$$

where b is the sigmoid inflexion point and a is the sigmoid inclination at $x = b$. The (crisp) output y of an HNFB cell, which corresponds to the cell's defuzzification process, is given by the weighted average shown in

$$y = \frac{\rho(x) * d_1 + \mu(x) * d_2}{\rho(x) + \mu(x)}. \tag{3}$$

Considering that the membership function ρ is the complement to 1 of membership function μ, the HNFB cell's output can be expressed by

$$y = \sum_{i=1}^{2} \alpha_i * d_i \tag{4}$$

where α_i represents the rule i firing level, given by $\alpha_1 = \rho(x)$; $\alpha_2 = \mu(x)$, and each d_i in (4) corresponds to one of the three possible consequents:

1) a *singleton*—the case where d_i = constant (Takagi–Sugeno order 0);
2) a *linear combination* of the inputs—the case where

$$d_i = \sum_{k=0}^{n} w_k x_k \qquad \text{(Takagi-Sugeno order 1)}$$

where x_k is the system's kth input, w_k represents the weights of the linear combination, and n is equal to the total number of inputs; the w_0 weight, with no input, corresponds to a constant value (bias);
3) the *output of a cell* of a previous level—the case where $d_i = y_j$, where y_j represents the output of a generic cell "j", whose value is also calculated by (4).

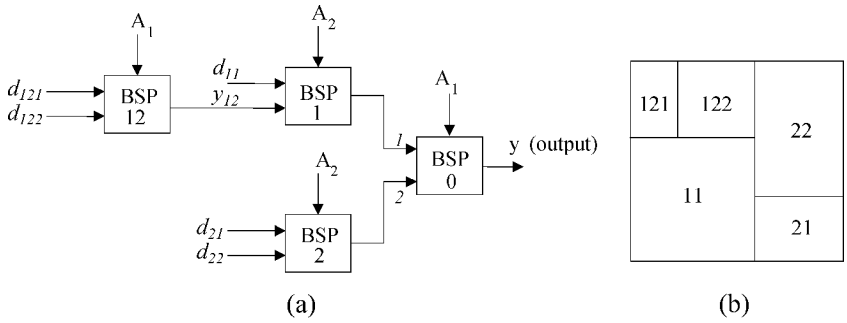

Fig. 3. (a) Example of an HNFB system. (b) Input space partitioning of the HNFB system.

B. HNFB Architecture

An HNFB model may be described as a system that is made up of interconnected HNFB cells. Fig. 3 illustrates an HNFB system along with the respective partitioning of the input space.

In this system, the initial partitions 1 and 2 ("BSP 0" cell) have been subdivided; hence, the consequents of its rules are the outputs of subsystem 1 and subsystem 2, respectively. In turn, subsystem 1 has as consequents d_{11} and y_{12}, while subsystem 2 has d_{21} and d_{22} as its consequent. Note that, consequent y_{12} is the output of the "BSP 12" cell. The output of the system in Fig. 3(a) is given by

$$y = \alpha_1.(\alpha_{11}.d_{11} + \alpha_{12}.(\alpha_{121}.d_{121} + \alpha_{122}.d_{122})) + \alpha_2.(\alpha_{21}.d_{21} + \alpha_{22}.d_{22}). \quad (5)$$

The example in Fig. 3 is linguistically translated by the following set of rules:

```
If A1 is low (A1 ∈ ρ0) then
  {If A2 is low (A2 ∈ ρ1) then y = d11
   If A2 is high (A2 ∈ μ1) then
        {If A1 is low (A1 ∈ ρ12) then
         y = d121
         If A1 is high (A1 ∈ μ12) then
         y = d122}
}
  If A1 is high (A1 ∈ μ0) then
    {If A2 is low (A2 ∈ ρ2) then y = d21
     If A2 is high (A2 ∈ μ2) then y = d22}
```

where

- $\boldsymbol{\rho_0}$, μ_0 membership functions that define the level 0 partition, which corresponds to the "BSP 0" cell;
- $\boldsymbol{\rho_1}$, μ_1 membership functions that define the subdivision 1, which corresponds to the "BSP 1" cell;
- $\boldsymbol{\rho_2}$, μ_2 membership functions that define the subdivision 2, which corresponds to the "BSP 2" cell;
- $\boldsymbol{\rho_{12}}$, μ_{12} membership functions that define the subdivision 12, which corresponds to the "BSP 12" cell.

C. Learning Algorithm

In the neuro-fuzzy literature the learning process is generally divided in two parts: 1) the identification of the structure and 2) the adjustments of parameters. The HNFB model follows the same process. However, only one algorithm carries out both learning tasks simultaneously. The HNFB system has a training algorithm based on the gradient descent method [2] for learning the structure of the model (the linguistic rules) and its fuzzy weights. The consequents (d_i's) and the parameters that define the profiles of the membership functions of the antecedents (parameters a and b) are regarded as the fuzzy weights of the neuro-fuzzy system.

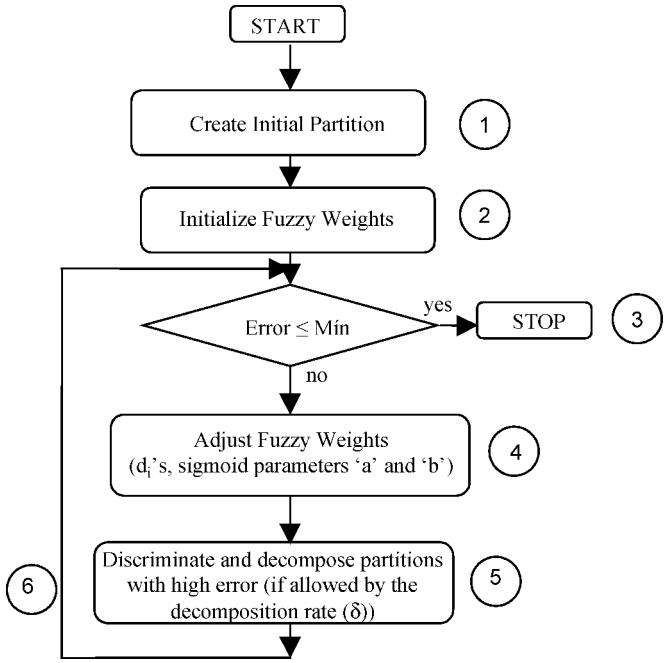

Fig. 4. Learning algorithm of HNFB model.

The learning algorithm of the HNFB model is performed in six steps, as illustrated in the flowchart of Fig. 4.

Learning Steps:

1) An initial bipartition is created, dividing the input space into two parts, through two fuzzy sets—*high* and *low*, for the input variable x (see Section III-E for variable selection methodology). In this step, the first BSP cell is created, which is called root cell.
2) Each weight d_i ($i \in \{1,2\}$), consequents of the first two rules, are initialized with the average of the target values of the output patterns that fall in the bipartitioning of index i. For example, to calculate the initial

value of the weight d_2, all target values that fall in bipartition 2 are summed and then divided by the number of patterns that fall in bipartition 2. This process also applies to the bias constant in the case of using linear combination as the rule's consequent. The b parameter (the sigmoid inflexion point, see (1)) is initialized with a value equal to the half of the universe of discourse of the cell's input variable. The other sigmoid parameter a is initialized, heuristically, with a value equal to twice the inverse of the universe of discourse. The following equations illustrate the initialization process of these parameters:

$$a = \frac{2}{(\mathrm{Lim}_S - \mathrm{Lim}_I)} \tag{6}$$

$$b = \frac{(\mathrm{Lim}_S + \mathrm{Lim}_I)}{2} \tag{7}$$

where Lim_I and Lim_S are the lower and upper limits of the universe of discourse of the cell's input variable.

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

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

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

4) The least-mean squares (LMS) [2] is used to calculate the rule's consequents (d_i), and the gradient descent method is employed to adjust the fuzzy weights a and b (rule's antecedents parameters).

5) Each bipartition is evaluated regarding its contribution to the total rms error and regarding the acceptable minimum error. Each bipartition with an unacceptable error is separated. The evaluation of the error generated for the data set that falls on the partitioning $\boldsymbol{ij}$, for example, is calculated by

$$E^{ij}_{\mathrm{rms}} = \sqrt{\frac{1}{L}\sum_{n=1}^{L}\alpha_i^n \times \alpha_{ij}^n \times (y_n - Y_n)^2} \tag{9}$$

where α_i^n and α_{ij}^n are the rules' firing levels for pattern n, as mentioned in Section II-A.

For each separated bipartition, a new node in the BSP tree is allocated, if not limited by the decomposition rate (δ), as explained hereafter. In other words, each separated partitioning is recursively decomposed (divided in 2). Two new membership functions are then generated for the selected variable (see Section III-E), which constitute the two partitions just created. The motivation for including the rule's firing levels (α_i^n and α_{ij}^n) 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.

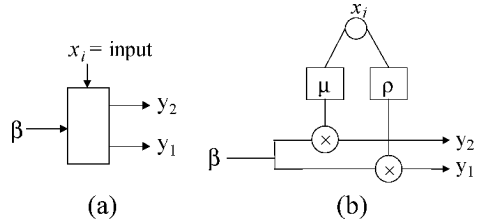

Fig. 5. (a) HNFB^{-1} cell schematic symbol. (b) HNFB^{-1} cell.

1) Decomposition Rate: In order to prevent the system's structure from growing indefinitely, a parameter, named decomposition rate (δ), was created. It acts as a limiting factor for the decomposition process. Its value is usually in the interval [0.001, 0.05]. During the learning process, the decomposition rate is constantly compared with the population density of patterns that fall in a specific bipartition. When population density of patterns (the rate between the number of patterns of a bipartition and the total number of patterns) falls below the decomposition rate, this bipartition cannot be further partitioned, which limits the size of the structure.

Therefore, a very low value for δ can result in a very big structure, compromising the generalization 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.

III. Inverted Hierarchical Neuro-Fuzzy BSP Model

The new inverted HNFB model, or HNFB^{-1}, was specifically designed to overcome the deficiencies of the original model for performing pattern classification tasks. This new system supports as many outputs as the number of classes desired, which considerably improves the interpretability of the rules. In addition, since it does not need to define ranges to specify pattern's classes, its performance is also improved with regard to classification problems.

The following subsections describe, respectively, the structure (input space partitioning) learning algorithm, the basic cell that composes the new HNFB^{-1} model, and the inverted classification architecture.

A. Learning the Input Space Partitioning

The HNFB^{-1} model makes use of the same learning algorithm that is employed in the original HNFB model [7] to establish the input space partitioning. Among the different training configurations of the original HNFB, the HNFB^{-1} model utilizes the LMS method to calculate the rules' consequents (d_i's), and the *gradient descent* method to calculate the rules' antecedent parameters (a and b in (1)). This configuration provided faster convergence. After this first learning phase, the structure is inverted, and the architecture of the HNFB^{-1} model is obtained. The basic cell of this new inverted structure is described hereafter.

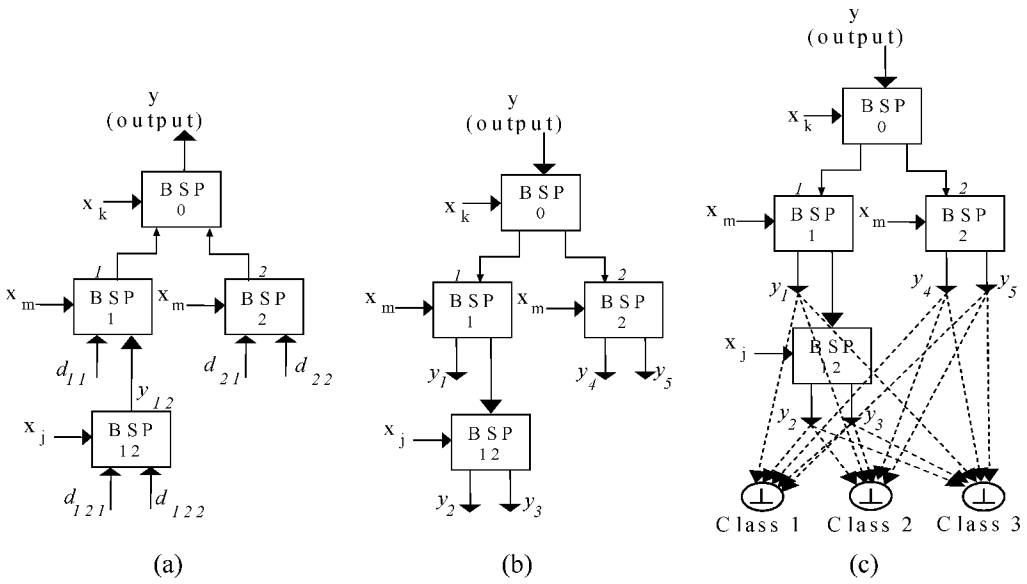

Fig. 6. (a) Original HNFB architecture obtained from a database containing three classes. (b) Inversion of the architecture shown in Fig. 5(a). (c) Connection of the inverted architecture to T-conorm cells.

B. Basic Inverted-HNFB Cell

Similarly to the original HNFB model, a basic inverted-HNFB cell is a neuro-fuzzy minisystem that performs fuzzy binary partitioning in a particular space according to the membership functions described by (1) and (2). However, after a defuzzification process, the inverted-HNFB cell generates two crisp outputs instead of just one. Fig. 5(a) shows the basic representation of the inverted-HNFB cell, and Fig. 5(b) illustrates the interior of the inverted-HNFB cell.

By considering that membership functions μ (*high*) and ρ (*low*) are complementary, the (crisp) outputs of an HNFB^{-1} cell are given by

$$y_1 = \beta * \rho(x) \tag{10}$$

$$y_2 = \beta * \mu(x) \tag{11}$$

where β corresponds to one of the following two possible cases:

1) β *is the input of the first cell*—the case in which $\beta = 1$, where the value '1" represents the entire input space, that is, the entire universe of discourse of the variable x_i that is being used as the cell's input;
2) β *is the output of a cell of a previous level*—the case in which $\beta = y_j$, where y_j represents one of the two outputs of a generic "j" cell, whose value is also calculated by (10) or (11).

C. Inverted-HNFB Architecture

As mentioned previously, the HNFB^{-1} model employs the learning algorithm of the original HNFB model to obtain its structure. Afterwards, the generated structure is inverted, in order to perform the classification task. Fig. 6(a) presents an example of the original HNFB architecture obtained during the training phase of a database containing three distinct classes, while Fig. 6(b) shows how the HNFB^{-1} model is obtained, after the inversion process.

In the HNFB^{-1} architecture shown in Fig. 6(b), it may be observed that the classification system has several outputs (y_1 to y_5), one for each existing leaf in the original HNFB architecture. The outputs of the leaf cells of the system in Fig. 6(b) (y_1 to y_5) are calculated by means of the following equations (considering the use of complementary membership functions $(\rho + \mu = 1)$):

$$y_1 = \rho_0 . \rho_1 \tag{12}$$

$$y_2 = \rho_0 . \mu_1 . \rho_{12} \tag{13}$$

$$y_3 = \rho_0 . \mu_1 . \mu_{12} \tag{14}$$

$$y_4 = \mu_0 . \rho_2 \tag{15}$$

$$y_5 = \mu_0 . \mu_2 \tag{16}$$

where ρ_i and μ_j follow the same definitions described in Section II-B.

Once the output of each leaf cell of the HNFB^{-1} system has been calculated, these cells are linked to the T-conorm neurons [see Fig. 6(c)] so that the final output of the pattern classification system may be obtained. This procedure is described hereafter.

D. Determining HNFB^{-1} System Outputs

After the inversion has been performed, the outputs are connected to T-conorm cells (the OR operator) that define the classes [see Fig. 6(c)]. The T-conorm cell (in this case, *class1, class2*, or *class3*) with the highest output value defines the class to which the pattern presented to the system belongs. The initial procedure for linking the leaf cells to the T-conorm neurons consists of connecting all the leaf cells with all the T-conorm neurons, according to the number of classes in which the database is organized. Once this connection has been made, it is necessary to establish weights for these links. For the purpose of assigning these weights, a learning method based on the LMS [2] has been employed.

After the weights have been determined, it is necessary to define which T-conorm operators will be used for obtaining the final output of the HNFB^{-1} model. In the example of Fig. 6(c),

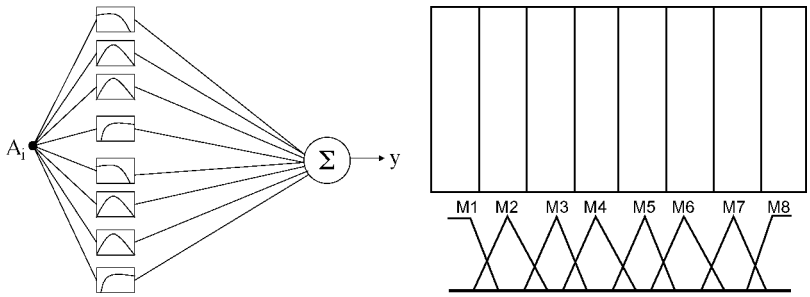

Fig. 7. (a) ANFIS minisystem (with one input) for selecting variables for the HNFB^{-1} model. (b) ANFIS partitioning of one input with eight fuzzy sets.

the outputs of the three T-conorm neurons are calculated according to the following:

$$\text{T-conorm class 1} = y_1 * w_{11} \oplus y_2 * w_{21} \oplus y_3 * w_{31} \oplus y_4 * w_{41} \oplus y_5 * w_{51} \quad (17)$$

$$\text{T-conorm class 2} = y_1 * w_{12} \oplus y_2 * w_{22} \oplus y_3 * w_{32} \oplus y_4 * w_{42} \oplus y_5 * w_{52} \quad (18)$$

$$\text{T-conorm class 3} = y_1 * w_{13} \oplus y_2 * w_{23} \oplus y_3 * w_{33} \oplus y_4 * w_{43} \oplus y_5 * w_{53} \quad (19)$$

where y_i, $i \in \{1,2,3,4,5\}$ are the outputs of the leaf cells; w_{ij}, $i \in \{1,2,3,4,5\}$, and $j \in \{1,2,3\}$ are the weight of the link between the leaf cell i and the T-conorm neuron j; and $\oplus$ is the T-conorm operation used for processing the output of the neuron.

In this paper, the limited-sum T-conorm operator [12] has been used. This operator is the most appropriated in this case, since it considers all inputs in the output calculation. Another T-conorm operator that is very popular in the literature, the *max*, only takes the maximum membership value into account, ignoring the membership values of other inputs.

The final output of the HNFB^{-1} system is specified by the highest output obtained among all the T-conorm neurons, determining the class to which the input pattern belongs.

E. Variable Selection for Each HNFB^{-1} Cell

In most pattern classification applications, the databases contain a large number of attributes, many of which are irrelevant or redundant. The strategy for selecting the variable employed by each HNFB^{-1} cell consists of ordering the attributes of the database in a decreasing order of relevance from the information point of view [13]. In the particular case of the HNFB and HNFB^{-1} models, selecting the relevant attribute for each cell avoids unnecessary partitioning and yields more compact BSP tree structures, which result in better generalization, fewer rules, and a higher level of interpretability.

In the HNFB^{-1}, the strategy employed for selecting variables is based on the Adaptive Network-Based Fuzzy Interence System (ANFIS) neuro-fuzzy model proposed by Jang [4]. The HNFB^{-1} makes use of ANFIS minisystems with only one input divided into eight fuzzy sets, as may be observed in Fig. 7(a). In this case, the input space is divided into eight partitions, as shown in Fig. 7(b).

The variables selection algorithm chooses an attribute (A_i) from the database and trains the ANFIS minisystem during a specified number of cycles. Next, the classification error for that attribute is calculated. Another attribute is then chosen, and the system is trained with the same number of cycles. After all attributes have been chosen, they are ordered according to their error (lowest training error first). Once the attributes have been ordered according to their relevance, each one is used as input for each level of the BSP tree during the learning process and construction of the HNFB^{-1} architecture. The same input (attribute) is used for all the nodes of the same level.

F. Fuzzy Rule Extraction

One of the main advantages of the HNFB^{-1} model is that it is capable of generating interpretable rules for the purpose of extracting information from a specific database. Differently from the original HNFB, the rules extracted in this model are of the following type: If x is high and y is big and... and w is hot, then class k.

In order to perform the rule extraction, the HNFB^{-1} model makes use of the inverted BSP structure before it is connected to the T-conorm neurons [see Fig. 6(b)]. In this approach, each partition of the input space (leaf node) will have an associated rule. However, it is important to point out that since the Inverted HNFB is a fuzzy system, the elements of each partition belong to all the existing k classes, with different membership levels. Therefore, each partition of the resulting BSP structure generates k subrules, each of which has an evaluation degree that is determined by the fuzzy accuracy and fuzzy coverage (see Section III-F-2)).

This new rule extraction process consists of the following steps:

1) routing on the BSP tree, calculating the firing level of each rule in each partitioning;
2) evaluation of the rules with the use of fuzzy accuracy and coverage.

These steps are described in the following subsections.

1) Routing of the BSP Tree: In order to illustrate the HNFB^1 rule extraction methodology, let us consider the hypothetical database shown in Table I that consists of eight patterns (A to H) of two attributes (age and weight) and two classes {0, 1}.

In the case of crisp trees, each pattern either belongs or does not belong to a partition, whereas in the fuzzy case, all the patterns in Table I (A, B, C, ..., H) are present to a different degree in all partitions. Therefore, the first step in the HNFB^{-1} model's rule extraction process consists of calculating each pattern's membership level in all the existing partitions (leaf nodes). This calculation is performed by routing on the BSP tree and making

TABLE I
DATABASE USED AS AN EXAMPLE IN THE RULE EXTRACTION METHOD

	Age	Weight	Class
A	10	30	1
B	12	30	1
C	40	90	1
D	55	45	0
E	55	55	0
F	60	92	0
G	62	95	0
H	75	40	1

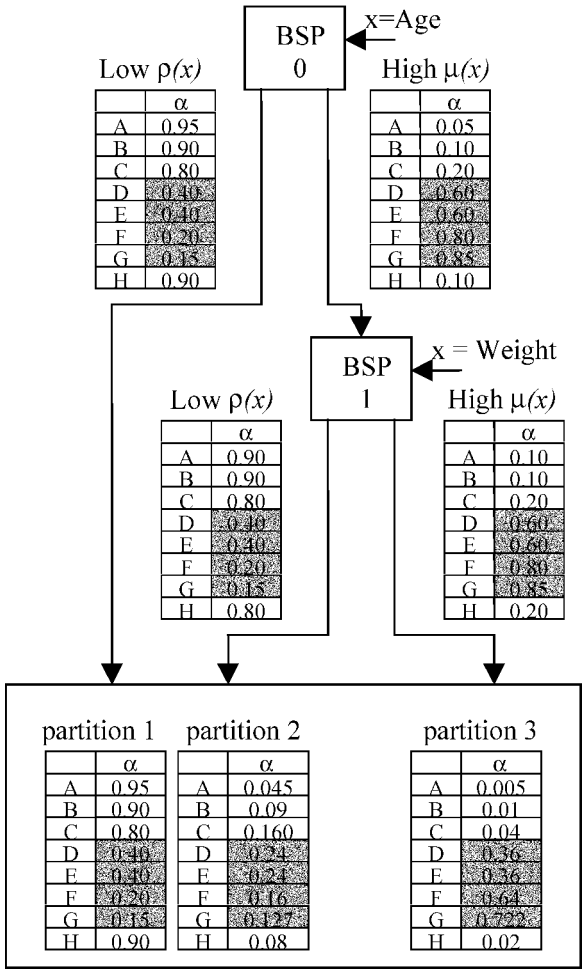

Fig. 8. Firing level (α) of each rule (partition) using product as T-norm operator.

the intersection (T-norm operator) of the membership degrees presented by each pattern in the membership functions of each level in the tree.

Fig. 8 presents, for the database in Table I, the result of each pattern's firing level (α) in each partition (tree node), which is calculated with the use of the "product" T-norm. Patterns that belong to class 0 are highlighted.

As may be observed in Fig. 8, all patterns are present in all partitions, to a greater or lesser extent, regardless of the class to which they belong. Therefore, it may be noticed that each routing process results in an antecedent that classifies all classes that exist in the database and generates k subrules for each partition. Each subrule's degree of suitability is defined by the fuzzy accuracy and coverage, as described in the next section. It must be noticed, however, that the resultant firing levels indicate the degree each pattern belongs to each partition. Therefore, Fig. 8 illustrates that patterns belonging to class 1 (in white) are better represented by partition 1, since they have higher membership degrees to this specific partition.

2) Rule Evaluation: In order to evaluate the subrules generated in each partition, two fuzzy evaluation measures were defined: fuzzy accuracy and fuzzy coverage.

a) Fuzzy accuracy: The accuracy of a rule measures how well it is applied to the data [14]. In order to determine how suitable a particular fuzzy rule describes a specific class k, the fuzzy accuracy measurement was created, as shown in

$$\text{Fuzzy\_Accuracy}_k^i = \frac{\sum_{j=1}^{P_k} \alpha_{k,j}^i}{\sum_{j=1}^{P_i} \alpha_j^i} \tag{20}$$

where $\text{Fuzzy\_Accuracy}_k^i$ is the accuracy of the rule for class k in partition i; $\alpha_{k,j}^i$ is the membership level of pattern j of class k in partition i; α_j^i is the membership level of pattern j in partition i (regardless of the class); P_k is the total number of patterns of class k; and P_i is the total number of patterns in partition i.

In the case of databases with different numbers of patterns per class, the correction factor (W_k^i) must be applied to compensate for nonuniform pattern distribution. The corrected fuzzy accuracy ($\text{Fuzzy}^*\text{\_Accuracy}_k^i$) is calculated by

$$\text{Fuzzy}^*\text{\_Accuracy}_k^i = \text{Fuzzy\_Accuracy}_k^i . W_k^i \tag{21}$$

$$W_k^i = \frac{1}{P_k \sum_{j=1}^{N_c} \frac{\text{Fuzzy\_Accuracy}_j^i}{P_j}} \tag{22}$$

where

W_k^i correction factor for the accuracy of class k in partition i;
N_c total number of classes;
P_j number of patterns of class j;
P_k number of patterns of class k.

In the case of the example presented in Table I, the patterns are distributed uniformly in each class (four patterns for each class); therefore, the value of W_k^i is 1, and therefore, the Fuzzy_Accuracy for classes 0 and 1 in partition 1 are the following (see Fig. 8):

$$\begin{aligned}
&\text{Fuzzy_Accuracy}_0^1 \\
&\quad = \frac{0.40+0.40+0.20+0.15}{0.95+0.90+0.80+0.40+0.40+0.20+0.15+0.90} \\
&\quad = \frac{1.15}{4.7} \\
&\quad = 0.245
\end{aligned}$$

$$\begin{aligned}
&\text{Fuzzy_Accuracy}_1^1 \\
&\quad = \frac{0.95+0.90+0.80+0.90}{0.95+0.90+0.80+0.40+0.40+0.20+0.15+0.90} \\
&\quad = \frac{3.55}{4.7} \\
&\quad = 0.755.
\end{aligned}$$

TABLE II
DESCRIPTION OF THE FEATURES OF THE DATABASES EMPLOYED

	IRIS DATASET	*WINE DATASET*	*PIMA INDIANS DIABETES DATABASE*	*BUPA LIVER DISORDERS*	*HEART DISEASE*
Number of Patterns	150	178	768	345	270
Number of Attributes	4	13	8	6	13
Number of Classes	3	3	2	2	2
Number of Patterns in Class 1	50	78	500	145	150
Number of Patterns in Class 2	50	59	268	200	120
Number of Patterns in Class 3	50	48			

From these results, it can be concluded that patterns belonging to partition 1 are more related to class 1, since the accuracies for this partition are 0.755 and 0.245 for classes 1 and 0, respectively. In other words, the partition's subrule associated with class 1 is more precise and adequate to describe the specific database.

The sum of the accuracy of all classes in a specific partition is always equal to 1, since each node's membership functions are complementary, and each pattern's level of presence in each partition is calculated by the intersection of all the firing levels in each node with the use of the product operator.

The sum of each pattern's membership levels in all the partitions is also equal to 1 on account of the features mentioned previously (complementary functions and product operator).

b) Fuzzy coverage: Fuzzy coverage supplies a measure of how comprehensive a rule is in relation to the total number of patterns in the rule base, i.e., it measures "how many" patterns are affected by the rule at hand. The definition of fuzzy coverage is given by

$$\text{Fuzzy}^i\text{\_Coverage} = \frac{\sum_{j=1}^{P_i} \alpha_j^i}{P} \tag{23}$$

where $\text{Fuzzy}^i\text{\_Coverage} =$ fuzzy coverage of partition i; $P =$ total number of patterns in the database; α_j^i is the membership level of pattern j in partition i; and P_i is the number of patterns in partition i.

Due to the aforementioned features (complementary membership functions and product operator), the composition of all the rules covers the total number of patterns. In other words, the sum of the fuzzy coverage of all partitions is equal to 1.

Again, using Fig. 8, the Fuzzy_Coverage for all three partitions are calculated as follows:

$$\text{Fuzzy}^1\text{\_Coverage} = \frac{0.95+0.90+0.80+0.40+0.40+0.20+0.15+0.90}{8} = 0.59$$

$$\text{Fuzzy}^2\text{\_Coverage} = \frac{0.045+0.09+0.16+0.24+0.24+0.16+0.127+0.08}{8} = 0.14$$

$$\text{Fuzzy}^3\text{\_Coverage} = \frac{0.005+0.01+0.04+0.36+0.36+0.64+0.722+0.02}{8} = 0.27.$$

These results indicate that, from the three partitions created, partition 1 (and its associated rule) encompasses (covers) more patterns than partitions 2 and 3. It must be emphasized, however, that as the hierarchical structure increases, the fuzzy coverage associated with each partition decreases, as bigger structures result in more partitions and, consequently, less patterns associated with each partitioning.

IV. CASE STUDIES

In order to evaluate the performance of the HNFB^{-1} model, five benchmark classification databases were selected among those most frequently employed in the area of neural nets and of neuro-fuzzy systems: Iris Dataset, Wine Data, Pima Indians Diabetes Database, Bupa Liver Disorders and Heart Disease. All these databases can be found in ftp://ftp.ics.uci.edu/pub/machine-learning-databases/

- *Iris Dataset*: a classification data set based on characteristics of a plant species (length and thickness of its petal and sepal) divided into three distinct classes (*Iris Setosa*, *Iris Versicolor* and *Iris Virginica*);
- *Wine Data*: data set resulting from chemical analyses performed on three types of wine produced in Italy from grapevines cultivated by different owners in one specific region;
- *Pima Indians Diabetes Database*: data set related to the diagnosis of diabetes (with or without the disease) in an Indian population that lives near the city of Phoenix, Arizona;
- *Bupa Liver Disorders*: data set related to the diagnosis of liver disorders and created by *BUPA Medical Research, Ltd.*;
- *Heart Disease*: data set related to diagnoses of people with heart problems.

Table II presents a summary of the main features of each database that has been used in this study.

A. Classification Performance

In the case of the Iris Dataset, Wine Data, Pima Indians Diabetes Database and Bupa Liver Disorders, in order to generate the training and test sets, the total set of patterns was randomly divided into two equal parts (database1.dat and database2.dat). Each of these two sets was alternately used either as a training set or as a test set. Table III below summarizes the results obtained in the classification of these four data sets with the use of the HNFB^{-1} model.

TABLE III
RESULTS OBTAINED WITH THE HNFB^{-1} MODEL FOR THE CLASSIFICATION OF FOUR DATABASES: IRIS DATASET, WINE DATA, PIMA INDIANS DIABETES DATABASE, AND BUPA LIVER DISORDERS

Data set used for testing	Hit percentage in the training set	Hit percentage in the test set	Number of rules generated	Decomposition Rate
iris1.dat	98.67 %	98.67 %	19	0.04
iris2.dat	98.67 %	98.67 %	19	0.04
Average IRIS	**98.67 %**	**98.67 %**		
wine1.dat	98.88 %	100.00 %	31	0.03
wine2.dat	98.88 %	98.88 %	23	0.04
Average WINE	**98.88 %**	**94.44%**		
pima1.dat	80.99 %	78.39 %	98	0.007
pima2.dat	79.17 %	78.13 %	55	0.01
Average PIMA	**80.08%**	**78.26%**		
liver1.dat	84.39 %	73.26 %	229	0.006
liver2.dat	75.00 %	73.41 %	55	0.007
Average LIVER	**79.69%**	**73.33%**		

TABLE IV
COST MATRIX OF THE RESULTS

Real Classification	Model Classification	
	Class 1 (Absence)	*Class 2 (Presence)*
Class 1 (Absence)	0	1
Class 2 (Presence)	5	0

As may be observed from Table III, there is a tradeoff between the hit percentage attained for classification and the number of rules generated. By decreasing the decomposition rate, it is possible to obtain better classification results, but the number of rules is increased, which reduces the model's interpretability. The average values (in bold type) of each application will be used for comparisons with other classification methods later on in Section IV-B (see Table VI).

As for the Heart Disease database (extracted from the StatLog project [15]), the tests were carried out with the use of the 9-Fold Cross Validation methodology [15], the same approach used by all algorithms that were analyzed by the StatLog project. This method consists of partitioning the database into nine subsets (heart1.dat, heart2.dat, heart3.dat, heart4.dat, heart5.dat, heart6.dat, heart7.dat, heart8.dat, and heart9.dat), where eight subsets are used for training and the remaining subset is used for testing (validation). The process is repeated nine times in such a way that each time a different subset of data is used for testing.

Thus, the database was randomly segmented into nine subsets with 30 elements each. Each subset contains about 56% of Class 1 records (without heart disease) and 44% of Class 2 records (with heart disease).

The methodology also makes use of a cost matrix, which is described in Table IV. The purpose of such a matrix is to penalize wrongly classified records in different ways, depending on the class. The weight of the penalty for Class 2 records that are classified as Class 1 records is 5, while the weight of the penalty for Class 1 records that are classified as Class 2 records is 1.

Therefore, the cost of wrongly classifying the patterns in the training and test data sets is given by (24) and (25), respectively, as follows:

$$C_{Tr} = \frac{P_1 * 5 + P_2 * 1}{P_{Tr}} \tag{24}$$

$$C_{Te} = \frac{P_1 * 5 + P_2 * 1}{P_{Te}} \tag{25}$$

where

- C_{Tr} cost in the training set;
- C_{Te} cost in the test set;
- P_1 number of patterns that were wrongly classified as belonging to Class 1;
- P_2 number of patterns that were wrongly classified as belonging to Class 2;
- P_{Tr} total number of patterns in the training set;
- P_{Ts} total number of patterns in the test set.

Table V presents the errors and costs of the training and test sets along with the number of rules and the decomposition rates for the HNFB^{-1} model.

Upon closer inspection of Table V, it may be observed that the configuration of the heart8.dat database as a test subset obtained lower errors and consequently a lower cost for the HNFB^{-1} model.

Fig. 9 illustrates the tree structure obtained by the HNFB^{-1} model for the eighth test (in bold type in Table V). The T-conorm links are not shown in this diagram because they would make it more difficult to understand the rule extraction process.

It may be observed in Fig. 9 that only nine of the 13 attributes of the database were used for achieving a satisfactory performance (x_1, x_4, x_5, and x_6 were not used). By means of tree routing, it is possible to extract the rules that describe the Heart Disease database. Some of the fuzzy rules extracted for Class 1 from the tree structure in Fig. 9 are listed hereafter. (The left side of each HNFB^{-1} cell is considered low partition and the right side the high partition.)

```
Sub_rule 1:
If X12 is low and If X13 is low and If X3
  is low
  then Class = 1
  [Accuracy: 0,7688/Coverage: 0,1314]
Sub_rule 2:
If X12 is low and If X13 is low and If X3
is high and If X2 is low and If X11 is low
  then Class = 1
```

TABLE V
RESULTS OBTAINED BY THE HNFB^{-1} MODEL FOR THE CLASSIFICATION OF THE HEART DISEASE DATABASE

Data set used for testing	Errors in the training set		Errors in the test set		Cost in the training set	Cost in the test set	Decomposition rate	Number of Rules
	Class1	Class2	Class1	Class2				
heart1.dat	11/134	17/106	4/16	1/14	0.4	0.3	0.01	50
heart2.dat	20/134	40/106	1/16	3/14	0.583	0.533	0.05	8
heart3.dat	11/134	24/106	2/16	3/14	0.545	0.566	0.03	22
heart4dat	11/133	19/107	1/17	1/13	0.441	0.2	0.01	47
heart5.dat	25/133	40/107	5/17	0/13	0.937	0.166	0.05	2
heart6.dat	7/133	22/107	1/17	3/13	0.487	0.533	0.01	36
heart7.dat	20/133	25/107	5/17	0/13	0.604	0.166	0.05	6
heart8.dat	**13/133**	**21/107**	**3/17**	**0/13**	**0.491**	**0.1**	**0.01**	**31**
heart9.dat	28/133	36/107	2/17	4/13	0.866	0.733	0.05	2
Average					0.594	0.366		

[Accuracy: 0,7704/Coverage: 0,4695]
Sub_rule 3:
If X_{12} is low and If X_{13} is low and If X_3 is high and If X_2 is low and If X_{11} is high
then $Class = 1$
[Accuracy: 0,7084/Coverage: 0,02969]

B. Comparison With Other Models

The results obtained for the Iris Dataset, Wine Data, Pima Indians Diabetes Database, and Bupa Liver Disorders data sets were compared with the results described in [16] where the performance of several models is presented: NN (nearest neighbor), kNN (k nearest neighbor, FSS (nearest neighbor with forward sequential selection of feature), and BSS (nearest neighbor with backward sequential selection of feature). In addition, HNFB^{-1} model has also been compared with other methods such as MFS (multiple feature subsets) [17], CART (CART decision tree) [18], C4.5 (C4.5 decision tree) [19], FID3.1 (FID3.1 decision tree) [20], MLP (multilayer perceptron) [2], and NEFCLASS [21]. Finally, the performance of HNFB^{-1} was compared with the original hierarchical neuro-fuzzy BSP models (see Table VI) HNFB, HNFB_fixed (which is the HNFB model with the same variable for all cells in the same level), HNFB_adaptive (the HNFB model with different variables for cells in the same level), and the Hierarchical Neuro-Fuzzy Quadtree (NFHQ) model [9], which uses the Quadtree partition of the input space [22].

Table VI presents a summary of the results obtained by the various different systems. The best performance for each data set, measured in terms of each model's hit percentage, is highlighted in bold type.

The classification results found for the Heart Disease data set were compared with the results found in the StatLog project [15]. According to the StatLog project methodology, comparison consists of calculating the average cost produced by the nine data subsets used for validation. Table VII presents the average cost for the nine training and test subsets. The result of the HNFB^{-1} model, which obtained the best performance (lowest average training and test costs) among all the systems presented [15], is highlighted in bold.

As can be seen from Table VII, the result obtained with the test set is better than with the training set. This uncommon result might be due to the fuzzy hierarchical recursive partitioning employed in the HNFB^{-1}, where more partitions are created in regions where the function complexity is more accentuated. In addition, due to the fuzzy aspect of the model, the generalization capability is enhanced when compared with more rigid classification methods. Therefore, the combined features of hierarchical partitioning and fuzzy mapping result in a more accurate model.

V. CONCLUSION AND FUTURE STUDIES

This paper has introduced and evaluated the Inverted Hierarchical Neuro-Fuzzy BSP System (HNFB^{-1}), a new neuro-fuzzy model that was specially created for the task of pattern classification and fuzzy rule extraction. The HNFB^{-1} model belongs to the new class of neuro-fuzzy systems, called Hierarchical Neuro-Fuzzy Systems [7]–[9], which allow a greater number of inputs and are able to generate their own structure, thus creating automatically their rule base. These models employ a recursive partitioning method that successively divides the input space into two regions.

The new model was tested on several benchmark applications that are common in the area of computational intelligence. The case studies demonstrated that the HNFB^{-1} model performs the pattern classification task quite well. In most cases, the results obtained with the HNFB^{-1} model proved to be as good as or better than the best results found by the other models and algorithms with which it was compared. The performance of the HNFB^{-1} models is remarkable in terms of processing time. For all the databases described in the case studies, the models converged in an order of magnitude of less than one minute of processing time on a Pentium III 500 MHz computer.

The number and quality of the rules that were extracted proved to be suitable for knowledge extraction applications. The fuzzy rules obtained after the learning phase were evaluated by means of fuzzy accuracy and fuzzy coverage measurements. These measurements attempt to supply clear information on the "crispness" and "coverage" of each rule.

The model is being modified so as to allow the implementation of Mamdani fuzzy inference systems [23], generating a new model called HNFB_Mamdani. Preliminary results have been

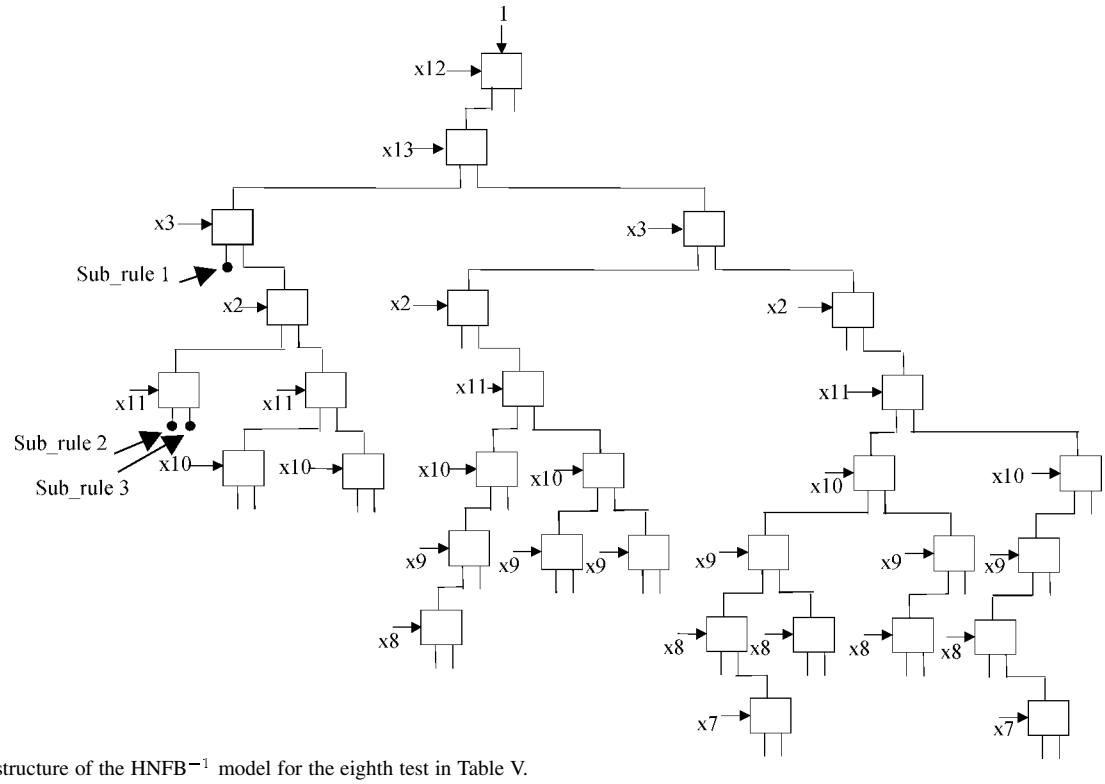

Fig. 9. Tree structure of the HNFB^{-1} model for the eighth test in Table V.

TABLE VI
COMPARISON OF THE AVERAGE PERFORMANCE OF SEVERAL CLASSIFICATION SYSTEMS

	Iris Dataset	Wine Data	Pima Indians Diabetes Database	Bupa Liver Disorders
NN		95.2 %	65.1 %	60.4 %
KNN		96.7%	69.7 %	61.3 %
FSS		92.8 %	73.6 %	56.8 %
BSS		94.8 %	67.7 %	60.0 %
MFS1		97.6 %	68.5 %	65.4 %
MFS2		97.9 %	72.5 %	64.4 %
CART			74.5 %	
C4.5	94.0 %		74.7 %	
FID3.1	96.0 %		75.9 %	
MLP			75.2 %	
NEFCLASS	96.0 %			
HNFB	**98.67 %**	98.31 %	77.08 %	74.49 %
HNFB fixed	**98.67 %**	97.8 %	78.0 %	
HNFB adaptive	**98.67 %**	97.8 %	**78.6 %**	
HNFQ	**98.67 %**	**98.88 %**	77.08 %	**75.07 %**
HNFB^{-1}	**98.67 %**	**99.44 %**	78.26 %	73.33 %

TABLE VII
TABLE COMPARING THE AVERAGE COST IN THE TRAINING AND TEST SET OF SEVERAL CLASSIFICATION SYSTEMS EVALUATED FOR THE HEART DISEASE DATABASE

Algorithm	Cost in Test	Cost in Training
HNFB^{-1}	***0.366***	***0.594***
Bayes	0.374	0.351
Discrim	0.393	0.315
LogDisc	0.396	0.271
Alloc80	0.407	0.394
QuaDisc	0.422	0.274
Castle	0.441	0.374
Ca15	0.444	0.330
Cart	0.452	0.436
Cascade	0.467	0.207
KNN	0.478	0
Smart	0.478	0.264
Dipol92	0.507	0.429
Itrule	0.515	-
BayTree	0.526	0.111
Default	0.560	0.560
BackProp	0.574	0.381
LVQ	0.600	0.140
IndCart	0.630	0.261
Kohonen	0.693	0.429
Ac2	0.744	0
Cn2	0.767	0.206
Radial	0.781	0.303
C4.5	0.781	0.439

reported in [24]. In this case, the output T-conorm operators are replaced by fuzzy sets, yielding rules of the following type: If x is high and y is big and ... and w is hot then z is low.

The variable selection methodology is also being altered to include two new model-free methods, called least-squares estimator (LSE) [25] and single-input effectiveness (SIE) [26]. Although the approach employed in this work provided good results, the one-dimension mini-ANFIS system has two disadvantages: it is a model-based strategy, where the good (or bad) performance might be associated with the model itself, not with the variable under analysis; and it does not take into consideration any dependency or correlation that might exist between vari-

ables. The two new methods were selected to overcome these drawbacks, and preliminary results [27] have demonstrated their superior performance.

In addition, in order to create potentially smaller trees, the strategy that has been employed for selecting variables, in which the same attribute is presented as input for all the nodes that belong to the same level in the hierarchy (fixed strategy), will be modified so as to allow a different variable to be selected for each tree node, regardless of its hierarchical level (adaptive strategy). The fixed strategy tends to generate unnecessary partitioning due to the fact that the selected variable is not always the most suitable one for all the nodes of that level. The advantage of the fixed strategy is its low computational cost, since feature selection is performed only once, before the learning process.

REFERENCES

[1] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Englewood Cliffs, NJ: Prenctice-Hall, 1997.

[2] S. Haykin, *Neural Networks—A Comprehensive Foundation*. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[3] J. M. Mendel, "Fuzzy logic systems for engineering: a tutorial," *Proc. IEEE*, vol. 83, no. 3, pp. 345–377, Mar. 1995.

[4] J. S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 665–685, May–Jun. 1993.

[5] R. Kruse and D. Nauck, "NEFCLASS—A neuro-fuzzy approach for the classification of data," in *Proc. 1995 ACM Symp. Applied Computing*, Nashville, TN, 1995, pp. 461–465.

[6] P. Vuorimaa, "Fuzzy self-organizing map," *Fuzzy Sets Syst.*, vol. 66, no. 2, pp. 223–231, 1994.

[7] F. J. de Souza, M. M. B. R. Vellasco, and M. A. C. Pacheco, "Load forecasting with the hierarchical neuro-fuzzy binary space partitioning model," *Int. J. Comput. Syst. Signals*, vol. 3, no. 2, pp. 1–15, 2002.

[8] M. M. B. R. Vellasco, M. A. C. Pacheco, L. S. Ribeiro Neto, and F. J. de Souza, "Electric load forecasting: evaluating the novel hierarchical neuro-fuzzy BSP model," *Int. J. Electr. Power Energy Syst.*, vol. 26, no. 2, pp. 131–142, Feb. 2004.

[9] F. J. de Souza, M. M. B. R. Vellasco, and M. A. C. Pacheco, "Hierarchical neuro-fuzzy quadtree models," *Fuzzy Sets Syst.*, vol. 130/2, pp. 189–205, Aug. 2002.

[10] M. Brown, K. M. Bossley, D. J. Mills, and C. J. Harris, "High dimensional neurofuzzy systems: Overcoming the curse of dimensionality," in *Proc. Int. Joint Conf. Fuzzy Systems 2nd Int. Fuzzy Eng. Symp.*, 1995, pp. 2139–2146.

[11] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, no. 1, pp. 116–132, 1985.

[12] E. Cox, *The Fuzzy Systems Handbook: A Practitioner's Guide to Building, Using, and Maintaining Fuzzy Systems*. Cambridge, MA: AP Professional, 1994.

[13] M. Dash and H. Liu. (1997) Feature selection for classification. *Intelligent Data Analysis* [Online], vol (3). Available: http://www.tnt.uni-hannover.de/org/whois/wissmit/rost/muster_erk/article.htm

[14] W. Klösgen and J. M. Zytkow, *Knowledge Discovery in Databases Terminology. Advances in Knowledge Discovery and Data Mining*, U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds., 1996, pp. 573–592.

[15] Heart Disease Dataset. [Online]. Available: http://www.ncc.up.pt/liacc/ML/statlog/datasets/heart/heart.doc.html

[16] D. W. Aha and R. L. Bankert, "Feature selection for case-based classification of cloud types: An empirical comparison," in *Proc. Amer. Assn. for Artificial Intelligence (AAAI-94)—Workshop Case-Based Reasonings*, 1994, pp. 106–112.

[17] S. D. Bay, "Nearest neighbor classification from multiple feature subsets," *Intell. Data Analysis*, vol. 3, no. 3, pp. 191–209, 1999.

[18] (1996) Pattern Recognition and Neural Networks. [Online]. Available: http://129.186.1.21/~dicook/stat501/97/lectures/4.17.html

[19] J. R. Quinlan, "Improved use of continuous attributes in C4.5," *J. Artif. Intell. Res.*, vol. 4, pp. 77–90, 1996.

[20] C. Z. Janikow and M. Faifer, "Fuzzy partitioning with FID3.1," in *Proc. IEEE 18th Int. Conf. North American Fuzzy Information Processing Soc.*, 1999, pp. 467–471.

[21] F. Klawonn, D. Nauck, and R. Kruse, "Generating rules from data by fuzzy and neuro-fuzzy methods," in *Proc. Fuzzy-Neuro-Systeme 1995*, 1995, pp. 223–230.

[22] R. A. Finkel and J. L. Bentley, "Quad trees, a data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, pp. 1–9, 1974.

[23] J. S. R. Jang and C.-T. Sun, "Neuro-fuzzy modeling and control," *Proc. IEEE*, vol. 83, no. 3, pp. 378–406, 1995.

[24] R. A. M. Bezerra, M. M. B. R. Vellasco, and R. Tanscheit. Modelo neuro-fuzzy hierárquico BSP Mamdani. presented at *VI Brazilian Symp. Intelligent Automation—VI SBAI—Simpósio Brasileiro de Automação Inteligente*. [CD-ROM](in Portuguese)

[25] F.-L. Chung and J.-C. Duan, "On multistage fuzzy neural network modeling," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 2, pp. 125–142, Apr. 2000.

[26] C. Yi and R. Diane, "An input pre-screening technique for control structure selection," *Comput. Chem. Eng.*, vol. 21, no. 6, pp. 563–5369, 1997.

[27] R. J. Contreras, M. M. B. R. Vellasco, and R. Tanscheit. Técnicas de seleção de características aplicadas a modelos neuro-fuzzy hierárquicos BSP. presented at *VI Brazilian Symposium on Intelligent Automation—VI SBAI—Simpósio Brasileiro de Automação Inteligente*. [CD-ROM](in Portuguese)

Laercio Brito Gonçalves received the B.S. degree in electrical and computer engineering from Universidade do Estado do Rio de Janeiro (UERJ), Rio de Janeiro, Brazil, in 1998, and the M.Sc. degree in electrical engineering from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro, Brazil, in 2001. He is currently working toward the Ph.D. degree in electrical engineering at PUC-Rio.

He is currently a collaborator in the Research Centre of Applied Computational Intelligence (ICA) of PUC-Rio. His current research interests include artificial neural networks, fuzzy systems, neuro-fuzzy systems, evolutionary computation, artificial immunologic systems, data mining, and time-series forecasting.

Marley Maria Bernardes Rebuzzi Vellasco (M'89) received the B.Sc. and M.Sc. degrees in electrical engineering from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro, Brazil, in 1984 and 1987, respectively, and the Ph.D. degree in computer science from the University College London (UCL), London, U.K., in 1992.

She is currently an Assistant Professor at the Electrical Engineering Department of PUC-Rio and jointly heads the Applied Computational Intelligence Laboratory (ICA) of PUC-Rio with Dr. M. Pacheco. She is also an Assistant Professor at the Computing and Systems Engineering Department, State University of Rio de Janiero (UERJ). She is the author of over 100 papers in books, professional journals, and conference proceedings in the area of soft computing. Her research interests include neural networks, fuzzy logic, neuro-fuzzy systems and evolutionary computation for decision support systems, pattern classification, time-series forecasting, control, optimization, knowledge discovery databases, and data mining.

Marco Aurélio Cavalcanti Pacheco received the B.Sc. and M.Sc. degrees in electrical engineering from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro, Brazil, in 1976 and 1980, respectively, and the Ph.D. degree in computer science from the University College London (UCL), London, U.K., in 1991.

He is currently an Assistant Professor at the Electrical Engineering Department of PUC-Rio and jointly heads the Applied Computational Intelligence Laboratory (ICA) of PUC-Rio with Dr. M. Vellasco. He is the author of over 100 papers in books, professional journals, and conference proceedings in the area of soft computing. His research interests include evolutionary computation, evolvable hardware, nanotechnology, neural networks, fuzzy systems, applied computational intelligence, knowledge discovery databases, and data mining.

Flavio Joaquim de Souza received the B.Sc. degree from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro, Brazil, in 1979; the M.Sc. degree from the Federal University of Rio de Janeiro (UFRJ) in 1987; and the Ph.D. degree in electrical engineering from PUC-Rio in 1979.

He is currently an Assistant Professor at the Computing and Systems Engineering Department, State University of Rio de Janeiro (UERJ), Rio de Janeiro, Brazil. His research interests are intelligent systems and geoprocessing.