Rule Extraction Using a Novel Class of Fuzzy Degraded Hyperellipsoidal Composite Neural Networks

Mu-Chun Su, Chien-Jen Kao, Kai-Ming Liu, and Chi-Yeh Liu

Department of Electrical Engineering, Tamkang University, Taiwan, R.O.C.

Abstract :

In this paper, we present an innovative approach to the rule extraction directly from experimental numerical data for system identification. We discuss how to use a novel class of fuzzy degraded hyperellipsoidal composite neural networks (FDHECNN's) to extract fuzzy if-then rules. The fuzzy rules are defined by hyperellipsoids of which principal axes are parallel to the coordinates of the input space. These rules are extracted from the parameters of the trained FDHECNN's. Based on a special learning scheme, the FDHECNN's can involve automatically to acquire a set of fuzzy rules for approximating the input/output functions of considered systems. A highly nonlinear system is used to test the proposed neuro-fuzzy systems.

I. Introduction

Recently, neural networks and fuzzy systems have been widely applied to many fields, yet in practice each has its own advantages and disadvantages. One of the most appealing aspect of neural networks is that they can inductively learn concepts from given experimental data. Furthermore, a two-layer backpropagation network with sufficient hidden nodes has been proven to be a universal approximator. Nevertheless, backpropagation networks suffer from lengthy training time. Besides, it is very difficult to interpret, in physically meaningful way, trained networks. That is , the knowledge is encoded in the parameters. This kind of knowledge representation is in a sense not acceptable to human users. On the other hand, the knowledge acquisition is the bottleneck of implementing fuzzy systems. Conventional approaches to fuzzy system designs either presume that fuzzy rules be given by human experts or involve dividing the input and the output space into fuzzy regions. However, in many applications, the initial linguistic rules are too crude for engineering purposes. On the other hand, the major restriction of the second approach is that the number of division of each input and output variables must be defined in advance. Besides, it is very difficult to apply such fuzzy systems to problems in which the number of input variables is large. Therefore, to overcome the bottleneck of the knowledge acquisition, several methods have been proposed for extracting fuzzy rules directly from numerical data [1], [2], [3].

One approach uses back-propagation networks to extract fuzzy rules [1]. The antecedent and consequent of the fuzzy if-then rules are represented by two separate back-propagation networks. The knowledge is encoded in the parameters of the trained networks. Thus such approach does not give a meaningful expression of the qualitative aspects of human reasoning. Another approach uses Gaussion potential functions for the construction of the input membership functions. Then the least mean squared (LMS) algorithm or back-propagation algorithm is used to train such fuzzy neural networks [2], [3]. The major disadvantage of this approach is that the initialization of weights. The initial values of the parameters are set either by the k-means algorithm or in such a way that the membership functions along each axis can cover the operating range totally with sufficient overlapping with each other. Input data clustered by the k-means algorithm may exhibit very different output responses. The initial membership functions set by the second method are decided too heuristically. Therefore, such initialization may not really speed up the learning process.

This paper is organized as follows. Section 2 introduces the novel class of fuzzy degraded hyperellipsoidal composite neural networks (FDHECNN's) and the hyperrectangular composite neural networks (HRCNN's). The training algorithm is given in section 3. The simulation results are demonstrated in section 4. Finally some concluding remarks are given in section 5.

II. Fuzzy Degraded Hyperellipsoidal Composite Neural Network

Ordinarily, fuzzy system designers often assume input variables are independent, therefore, the membership function is assigned variable by variable. Fig. 1 shows an example of conventional fuzzy rule partitions. Most of the membership functions are assumed to be triangular, trapezoidal or bell-shaped. In fact, there is no straightforward method for choosing membership functions. Either trial-and-error tuning or backpropagation algorithm is used to find the right number of division of inputs

0-7803-2461-7/95/$4.00 © 1995 IEEE

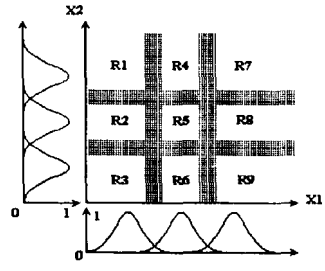

Fig. 1 An example of traditional fuzzy rule partitions

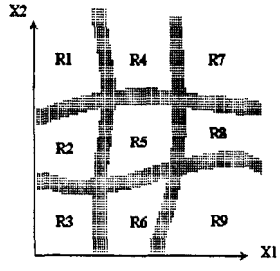

Fig. 2 An example of modified fuzzy rule partitions

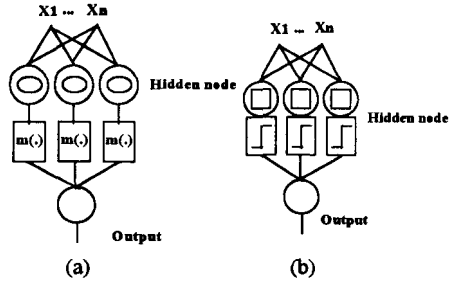

Fig. 3 The symbolic representations of (a) a two-layer FDHECNN and (b) a two-layer HRCNN

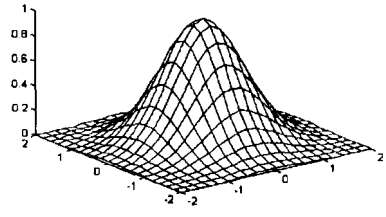

Fig. 4 An example of $m_j(\underline{x})$ when $d_j=0$

and appropriate membership functions. Since it is likely that there exist correlations among input variables, in this case, the fuzzy rule partitions should not be divided variable by variable. The fuzzy regions should be arbitrarily shaped, as shown in Fig.2 . A neural network approach to fuzzy rule partitions was proposed in [1]. However, the extracted knowledge is still too implicit. Therefor, we propose to use the aggregations of hyperellipsoids to approximate arbitrary fuzzy rule partitions. This new scheme is implemented as an two-layer FHRCNN.

The novel class of FDHECNN's is a modified version of the class of neural networks with quadratic junctions. The inherent architectural efficiency of this kind of neural networks was demonstrated in [4], [5], [6]. A two-layer feedforward fuzzy degraded hyperellipsoidal composite neural network (FDHECNN) shown in Fig.3(a) is described by

$$net_j(\underline{x}) = d_j^2 - \sum_{i=1}^{n}(w_{ji}x_i + b_{ji})^2, \quad (1)$$

$$m_j(\underline{x}) = \exp\{-s_j^2[\max(d_j^2, d_j^2 - net_j(\underline{x})) - d_j^2]\}, \quad (2)$$

and

$$Out = \sum_{j=1}^{J} w_j m_j(\underline{x}) + \theta \quad (3)$$

where b_{ji} , w_{ji} and d_j are adaptive weights, s_j is the slope value which regulates how fast $m_j(\underline{x})$ goes down, w_j is the connection weight from hidden node j to the output, and θ is a small real number. If we assume d_j be zero then the FDHECNN's are isomorphic to the radial basis function (RBF) networks. The RBF networks using Gaussion potential functions have been proven to be universal approximators [7]. The essential difference between the Gaussion function and the $m_j(\underline{x})$ can be illustrated in Fig.4 and 5 .

Obviously, the $m_j(\underline{x})$offers more flexibility. The functionality of the $m_j(\underline{x})$ can change from the Gaussion function depicted in Fig. 4 to the steplike depicted in Fig. 5. Therefore the FDHECNN's offer significantly more flexiblility in approximating functions.

Most fuzzy systems employ "center average defuzzifier"

$$out = \sum_{j=1}^{J} out_j mem_j(\underline{x}) / \sum_{j=1}^{J} mem_j(\underline{x}) \quad (4)$$

However each fuzzy rule plays a different role in contributing to the computation of the crispy final output. Therefore, we propose the "weighted center average defuzzifier"

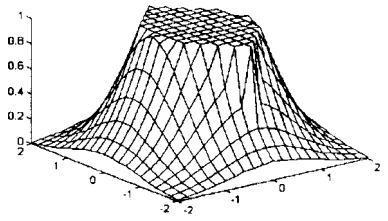

Fig. 5 An example of $m_j(\underline{x})$ when d_j is a small number

$$out = \sum_{j=1} cf_j out_j m_j(\underline{x}) + \theta = \sum_{j=1}^{J} w_j m_j(\underline{x}) + \theta \qquad (5)$$

where cf_j is the certainty factor of the jth fuzzy rule. The weighted center average defuzzifier in a sense takes the linear combination of the firing strengths of J fuzzy rules. A two-layer hyperrectangular composite neural network (HRCNN) shown in Fig.3(b) is described by

$$net_j(\underline{x}) = \sum_{i=1}^{n} f((M_{ji} - x_i)(x_i - m_{ji})) - n, \qquad (6)$$

$$Out_j(\underline{x}) = f(net_j), \qquad (7)$$

and

$$Out = \sum_{j=1}^{J} Out_j \qquad (8)$$

where

$$f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \qquad (9)$$

M_{ji} and $m_{ji} \in$ R are adjustable weights, $\underline{x}(=[x_1, x_2, \ldots, x_n]^t)$ is an input pattern, and $out_j(\underline{x}) : R^n \rightarrow \{0,1\}$ is an output function for a neural node with hyperrectangular neural-type junctions. The classification rule extracted from a trained HRCNN with k hidden nodes is expressed in the form of

IF($\underline{x} \in [m_{11}, M_{11}]x[m_{12}, M_{12}]x...x[m_{1n}, M_{1n}]$)
THEN *Out* is 1;
...
IF($\underline{x} \in [m_{k1}, M_{k1}]x[m_{k2}, M_{k2}]x...x[m_{kn}, M_{kn}]$)
THEN *Out* is 1. (10)

The domain defined by the antecedent is a hyperrectangle $[m_1, M_1]x...x[m_n, M_n]$. The supervised decision-directed learning (SDDL) algorithm is utilized to train HRCNN's. The algorithm generates a two-layer feedforward HRCNN in a sequential manner by adding hidden nodes as needed. A detailed description of the SDDL algorithm is given in [4], [8].

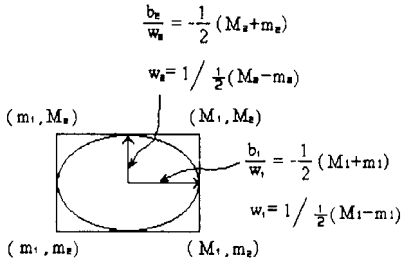

Fig. 6 Parameters initialization procedure

III.Hybrid Training Algorithm for FDHECNN's

Basically, the FDHECNN's provide more flexible local functions (the output functions of hidden nodes) than the RBF networks do. Therefore, it can be shown to be a universal approximator. The proof is given elsewhere. The performance of a FDHECNN depends on the chosen fuzzy hyperellipsoids defined by $d_j^2 \geq \sum_{i=1}^{n}(w_{ji}x_i + b_{ji})^2$ and the connection weights. The fuzzy hyperellipsoids should suitably sample the input domain and reflect the data distribution. Traditional approaches used the k-means algorithm to cluster input data directly on the input space. They do not take account for the output response of each input pattern. Besides, in the process of system identification, input data are often chosen to scatter evenly over the input space. The k-means algorithm does not reflect any important data distribution in these cases. Based on above discussions, a hybrid learning algorithm consists of the following steps:

Step 1. Partition the Output Space into Fuzzy Regions

Divide the output into regions (the length of these divided intervals can be equal or unequal), denoted by SN (Small N), ..., S1 (Small 1), CE (Center), L1 (Large 1),..., LN (Large N). At this time, we do not need to assign a fuzzy membership function to each region since the defuzzifier employes the "weighted center average defuzzifier".

Step 2. Transform a Function Approximation Problem into a Pattern Recognition Problem

The original input-output pairs are then transformed into quantized input-output pairs, that is, a function approximation problem is converted to a (2N+1)-class pattern recognition problem. By the SDDL algorithm, (2N+1) two-layer HRCNN's (hyperrectangular composite neural networks) are

created to recognize these patterns. After sufficient training, (2N+1) sets of hyperrectangles are created. Assume there be m_i^e hidden nodes in the ith trained HRCNN, i=1,.., 2N+1. The m_i^e hidden nodes are then ranked in the order of respective representativeness.

Step 3. Initialize and Update Weights

For output class i, the first n_i out of m_i^e hidden nodes are selected to initialize the weights (b_{jl}, d_j, and w_{jl}) of a FDHECNN with ($n_1+n_2+...+n_{2N+1}$) hidden nodes. Fig.6 illustrates such initialization scheme. The connection weights, $w_j's$, are initialized to be the mean values of the (2N+1) segmented intervals, respectively.

There are two approaches to adjust the weights. The first one is to fix the weights, b_{jl},d_j, and w_{jl}, and to update only the connection weights, w_j, in real-time using the recursive least square error algorithm. However, it is advantageous to update all weights, b_{jl}, d_{jl} ,w_{jl} and w_j, simultaneously because this will significantly improve both the modeling capability. The reason why the modeling capability can be improved is because the boundary of hyperellipsoid can be adjusted to efficiently sample the data distribution.

Assuming the given training data set has p entries, we can define an error function as

$$E=\sum_p E_p=\sum_p \tfrac{1}{2}(Out_p-t_p)^2 \tag{11}$$

where t_p is the pth desired output and Out_p is the actual output of the FHRCNN with respect to the pth input. By use of the recursive LMS algorithm or back-propagation algorithm, the weights can be adjusted to minimize the total error E. All weights are adapted according to the following equations:

$$\frac{\partial E}{\partial d_j}=\frac{\partial E}{\partial Out_p}\frac{\partial Out_p}{\partial m_j}\frac{\partial m_j}{\partial d_j} \tag{12}$$

$$=\begin{cases} 0 & z_j>0 \\ \sum_p (Out_p-t_p)w_j s_j^2 2d_j e^{s_j^2 z_j} & o.w. \end{cases}$$

$$\frac{\partial E}{\partial \theta}=\sum_p \frac{\partial E_p}{\partial Out_p}\frac{\partial Out_p}{\partial \theta}$$

$$=\sum_p (Out_p-t_p) \tag{13}$$

$$\frac{\partial E}{\partial w_j}=\sum_p \frac{\partial E_p}{\partial Out_p}\frac{\partial Out_p}{\partial w_j}$$

$$=\sum_p (Out_p-t_p)m_j(\underline{x}) \tag{14}$$

$$\frac{\partial e}{\partial b_{jl}}=\frac{\partial E}{\partial Out_p}\frac{\partial Out_p}{\partial m_j}\frac{\partial m_j}{\partial b_{jl}} \tag{15}$$

$$=\begin{cases} 0 & z_j>0 \\ \sum_p (Out_p-t_p)w_j(-2)s_j^2(w_{jl}x_l+b_{jl})e^{s_j^2 z_j} & o.w. \end{cases}$$

$$\frac{\partial E}{\partial w_{jl}}=\frac{\partial E}{\partial Out_p}\frac{\partial Out_p}{\partial m_j}\frac{\partial m_j}{\partial w_{jl}} \tag{16}$$

$$=\begin{cases} 0 & z_j>0 \\ \sum_p (Out_p-t_p)w_j s_j^2(-2x_l)(w_{jl}x_l+b_{jl})e^{s_j^2 z_j} & o.w. \end{cases}$$

Step 4. Interpret the trained FDHECNN's

Let HE_k be the hyperellipsoid, $\sum_{i=1}^{n}(w_{ki}x_i+b_{ki})^2 \le d_k^2$. From the trained FHRCNN, a set of fuzzy rules can be extracted and represented as

Rule1: IF($\underline{x}$ is *near* HE_1)
THEN the system output is LN
.
.
. (17)
Rule 2N+1: ELSE IF($\underline{x}$ is *near* HE_{2N+1})
THEN the system output is LP

IV. Simulation Results

The plant to be identified is governed by the differential equation

$$\begin{aligned} y(t+1)=&(0.8-0.5\exp(-y^2(t)))y(t) \\ &-(0.3+0.9\exp(-y^2(t)))u(t) \\ &+0.1\sin(3.1415y(t)) \end{aligned} \tag{18}$$

where u and y are the input and output of the plant, respectively. The training patterns are in two-dimensional input space (y(k) and u(k)) and one-dimensional output space, as shown in Fig.7. The input parts of the training data are chosen to scatter evenly over a square region [-2,2]x[-2,2]. The total number of training data is 441.

First, we partition the output variables into five regions, [-2.45, -1.47](large negative), [-1.47, -0.49](small negative), [-0.49, 0.49](near zero), [0.49, 1.47](small positive) and [1.47, 2.45](large positive). In the following step, five HRCNN's were trained to recognize the 5-class pattern recognition problem.

After training, five HRCNN's with 7, 13, 16, 15 and 8 hidden nodes, corresponding to 5 different classes, were created. A FDHECNN with (1+2+2+2+3) hidden nodes was trained to identify the plant. Fig.8 depicts the three dimensional representatives learned by the FHRCNN's using the back-propagation algorithm. The error curve is shown

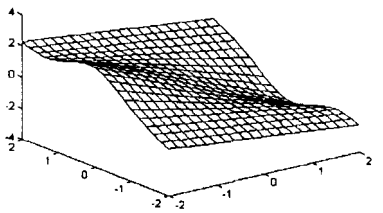

Fig. 7 The training patterns presented to the FDHECNN for learning

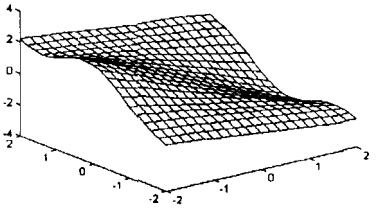

Fig. 8 Three dimensional representatives learned by the FDHECNN

in Fig.9. The ten extracted fuzzy rules are represented as

IF (y(k),u(k)) is near the ellipse :
$[(y(k)-3.25)^2/1.00^2]+ [(u(k)+0.10)^2/0.52^2]=1$
THEN y(k+1) is large negative;

IF (y(k),u(k)) is near the ellipse :
$[(y(k)-3.97)^2/2.76^2]+ [(u(k)+3.38)^2/0.66^2]=1$
THEN y(k+1) is small negative.

IF (y(k),u(k)) is near the ellipse :
$[(y(k)+0.11)^2/0.63^2]+ [(u(k)-1.20)^2/0.17^2]=1$
THEN y(k+1) is small negative.

IF (y(k),u(k)) is near the ellipse :
$[(y(k)+0.7)^2/0.43^2]+ [(u(k)+0.88)^2/0.18^2]=1$
THEN y(k+1) is near zero.

IF (y(k),u(k)) is near the ellipse :
$[(y(k)+1.40)^2/0.23^2]+ [(u(k)+1.10)^2/0.19^2]=1$
THEN y(k+1) is near zero.

IF (y(k),u(k)) is near the ellipse :
$[(y(k)+0.44)^2/0.39^2]+ [(u(k)-2.80)^2/0.76^2]=1$
THEN y(k+1) is small positive;

IF (y(k),u(k)) is near the ellipse :
$[(y(k)-1.35)^2/0.76^2]+ [(u(k)-2.97)^2/0.74^2]=1$
THEN y(k+1) is small positive.

IF (y(k),u(k)) is near the ellipse :
$[(y(k)+2.76)^2/0.86^2]+ [(u(k)-0.14)^2/0.52^2]=1$
THEN y(k+1) is large positive.

IF (y(k),u(k)) is near the ellipse :
$[(y(k)+2.27)^2/1.02^2]+ [(u(k)-2.62)^2/0.73^2]=1$
THEN y(k+1) is large positive.

IF (y(k),u(k)) is near the ellipse :
$[(y(k)+1.29)^2/0.21^2]+ [(u(k)+67.60)^2/62.49^2]=1$
THEN y(k+1) is large positive.

V. Concluding Remarks

We have proposed a novel approach to the rule extraction directly from experimental numerical data without interviewing domain experts. This approach employs the fuzzy degraded hyperellipsoidal composite neural networks as building blocks and the LMS or the backpropagation algorithm as a training procedure. A special preprocessing scheme, transforming a function approximation problem into a pattern recognition problem, is proposed to find a set of good initializations in order to speed up the learning steps. Most of all, the correlations among input features have been considered. The fuzzy rule partitions are no longer parallel to input features. The aggregation of a set of hyperellipsoids are utilized to represent a fuzzy rule.

References

[1]H. Takagi and I. Hayashi, "NN-driven fuzzy reasoning", Int. J. Approximate Reasoning, vol.5, No3, PP. 191-212, May, 1991.

[2]L. X. Wang, Adaptive Fuzzy Systems and Control: Design and Stability Analysis, Prentice Hall, Inc. New Jersey, 1994.

[3]C. T. Lin C. S. G. Lee, Neural network-based fuzzy logic control and decision system, "IEEE Trans. Computers, vol.40, No.12, PP.1320-1336, 1991.

Fig. 7 Evolution of mean squared error

[4]M. C. Su, A Neural Network Approach to Knowledge Acquisition, Ph.D. thesis, University of Maryland, August, 1993.

[5]N. DeClaris and M. C. Su, "A novel class of neural networks with quadratic junctions, "IEEE Int. Conf. on systems, Man, and Cybernetics, PP. 1557-1562, 1991 (Received the 1992 Franklin V. Taylor Award).

[6]N. DeClaris and M. C. Su, "Introduction to the theory and application of neural networks with

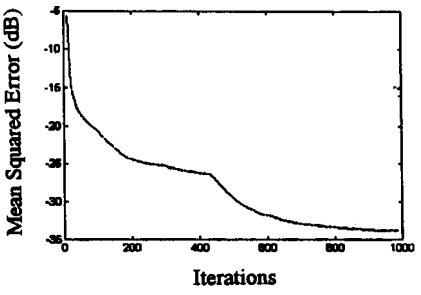

Fig. 9 Evolution of the mean squared error

quadratic junctions, "IEEE Int. Conf. on systems, Man, and Cybernetics, PP. 1320-1325, 1992.

[7]T. Poggio and F. Girosi, "Networks for approximation and learning ," Proc of the IEEE, vol.78, No.9, PP.1481-1497, 1990.

[8]C. Hsieh, M. C. Su, and C. T. Tseng, "A Mandarin recognition system based on a novel class of hyperrectangular composite neural networks, "to be published in IASTED Int. Conf. Modeling, Simulation and Identification, 1994.