On rule pruning using fuzzy neural networks

Nikhil R. Pal[a,*], Tandra Pal[b]

[a] *Machine Intelligence Unit, Indian Statistical Institute, 203 B.T. Road, India*
[b] *Department of Computer Science, Regional Engineering College, Durgapur, W.B., India*

Received July 1996; received in revised form August 1997

Abstract

Shann and Fu (SF) proposed a fuzzy neural network (FNN) for rule pruning in a fuzzy controller. In this paper we first analyze the FNN of SF and discuss some of its limitations. SF attempted to eliminate redundant rules interpreting some of the connection weights as certainty factors of rules. In their strategy the connection weights are unrestricted in sign and hence their interpretation as certainty factors introduces some inconsistencies into the scheme. We propose a modification of this FNN, which eliminates these inconsistencies. Moreover, we also propose a pruning scheme which, unlike the scheme of SF, always produces a compatible rule set. Superiority of the modified FNN is established using the inverted pendulum problem

Keywords: Fuzzy control; Fuzzy neural networks; Rule pruning; Certainty factors

1. Introduction

Fuzzy logic provides an effective means to capture the approximate, inexact nature of the real world. One of its most important applications is the fuzzy logic controller (FLC) [5, 8, 9]. FLC converts the linguistic control strategy based on expert's knowledge into an automatic control strategy [3]. Experience shows that FLC yields results sometimes superior to those obtained by conventional control algorithms. In particular FLC appears very useful when the processes is too complex to analyze by conventional quantitative technique or when the available sources of information are interpreted qualitatively, inexactly or imprecisely. Thus, fuzzy logic control falls in between the conventional precise mathematical control and expert-like decision making.

The performance of FLC mostly relies on two important factors: sound technique of knowledge acquisition and the availability of human experts. These two factors may restrict the application domain of FLCs. Thus, extraction of an appropriate set of rules or selection of an optimal subset of rules from the set of all possible rules is an important problem. Moreover, given a set of rules there is a great need of learning or tuning the rules to achieve a desired level of controller performance.

Several attempts [1, 2, 4, 10–15] have been made to integrate fuzzy systems and neural networks, with a view to achieving in the same system, the human style inferencing and natural language description of fuzzy systems with the learning and parallel processing abilities of neural networks.

Lee et al. [10] proposed an algorithm for adjusting (tuning) the membership functions of antecedent linguistic values of the rule set by error backpropagation (EBP), where the consequent parts were considered fixed. Thus the extracted fuzzy rules after tuning retain the same linguistic description as the initial rules. Li and Wu [12] proposed a five-layer neuro-fuzzy hierarchical system with if–then rules for pattern classification problem. A five-layer fuzzy neural network is also presented by Yao et al. [15]. The parameters of the net are identified using evolutionary programming and the tuned network is then pruned to extract a small set of rules.

Lin and Lee [11] presented a multilayer feedforward connectionist model designed for FLCs and decision-making systems. A hybrid two-step learning scheme that combined self-organized (unsupervised) and supervised learning algorithms for selection of fuzzy rules and tuning of membership functions were developed. Lin and Lee used Kohonen's self-organized feature map [6] for finding the centers of the membership functions. After the selection of the rule set, i.e., when the network architecture is established, the second step of supervised learning begins. Some heuristic guidelines for rule reduction and combination were also provided.

Shann and Fu [14] presented a layer-structured fuzzy neural network (FNN) for selection of rules. Initially, the FNN was constructed to contain all possible fuzzy rules. After the EBP training, redundant rules were deleted by a rule pruning process for obtaining a concise fuzzy rule-base. The architecture of Shann and Fu's network is similar to that of Lin and Lee [11] in several respects.

In this paper we first discuss some conceptual problems associated with the FNN of Shann and Fu and then propose a modification of the same. The superiority of the modified FNN is established using the inverted pendulum problem. In our subsequent discussion the FNN of Shann and Fu will be referred as FNNO (O for original) and the modified form as FNNM.

The organization of the rest of the paper is as follows. Section 2 discusses Shann and Fu's FNN and its limitations, and Section 3 presents the proposed modification. Details of implementation and results are discussed in Section 4. We end our report with conclusions in Section 5.

2. FNN of Shann and Fu (FNNO)

The fuzzy neural network of Shann and Fu has five layers, as shown in Fig. 1. Consecutive layers are constructed according to the consecutive steps of a fuzzy logic controller. We use indices i, j, k, and l for nodes in layers 2, 3, 4, and 5, respectively. The output from the nth node of layer m will be denoted by y_n^m.

A node in layer 1 represents an input linguistic variable which takes a crisp value from outside and delivers it to the next layer, the membership function layer. Each node is connected to only those nodes of layer 2 which represent the linguistic values of corresponding linguistic variable.

Nodes at layer 2 act as membership functions to represent the terms of the respective linguistic variable. All link weights between layers 1 and 2 are set to unity.

Each node in layer 3 represents a possible IF-part of a fuzzy rule. The weights of the links are set to unity. The nodes in this layer perform the AND operation. The output of the node j is computed as

$$y_j^3 = \min_{i \in I_j}(y_i^2). \tag{1}$$

Here I_j is the set of indices of the nodes in layer 2 that are connected to node j in layer 3, and y_i^2 is the output of node i in layer 2.

A node in layer 4 represents a possible THEN-part of a fuzzy rule and performs the OR operation to inte-

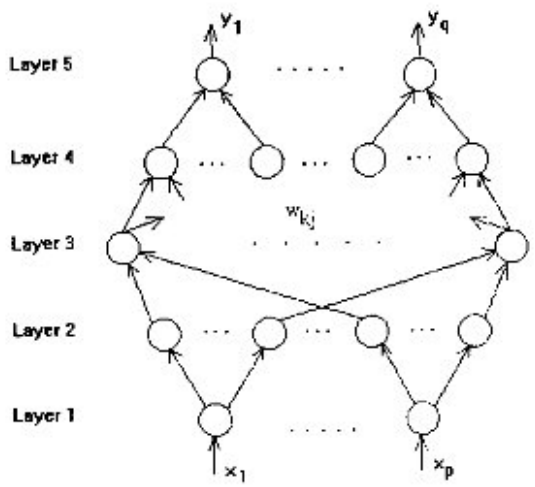

Fig. 1. The fuzzy neural network.

grate the fired rules which have the same consequent clause. The nodes of layers 3 and 4 are fully connected. The weight w_{kj} of the link connecting the node k in layer 4 to the node j in layer 3 represents the certainty factor of the corresponding fuzzy rule. The output of the node k in this layer is:

$$y_k^4 = \max_{j \in I_k}(y_j^3 w_{kj}), \tag{2}$$

where I_k is the set of indices of the nodes in layer 3 that are connected to the node k in layer 4.

The output linguistic variables are represented by nodes in layer 5. A node (say the lth node) in this layer computes the defuzzified value as

$$y_l^5 = \sum_{k \in I_l}(y_k^4 a_{lk} c_{lk}) \Big/ \sum_{k \in I_l}(y_k^4 a_{lk}). \tag{3}$$

Here I_l is the set of indices of the nodes in layer 4 which are connected to the node l in layer 5. a_{lk} and c_{lk} are respectively the area and centroid of the membership function of the output linguistic value represented by the k in layer 4. The weights of the links from the nodes in layer 5 to the nodes in layer 4 are unity. Thus the only learnable weights in the network are w_{kj}'s between layers 3 and 4.

The weights w_{kj} of the FNNO are learned using the error backpropagation algorithm based on a set of training data. Let d_l be the target output of the node l in layer 5 for an input vector $X = (x_1, x_2, \ldots, x_p)$. The error for the data point X is

$$E_X = \sum_{l=1}^{q}(d_l - y_l^5)^2, \tag{4}$$

where q is the number of nodes in layer 5. For notational simplicity we drop the subscript X from E_X in the following discussion.

Let the weight w_{kj} after t steps of update be denoted by $w_{kj}(t)$, then $w_{kj}(t)$ is modified according to

$$w_{kj}(t+1) = w_{kj}(t) - \eta \frac{\partial E}{\partial w_{kj}(t)}, \tag{5}$$

where η is the learning rate, and

$$\frac{\partial E}{\partial w_{kj}} = \begin{cases} -(d_l - y_l)\dfrac{a_{lk}(c_{lk} - y_l^5)}{\sum_{k' \in P_l} y_{k'}^4 a_{lk'}} y_j^3 & \text{if } j = r, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

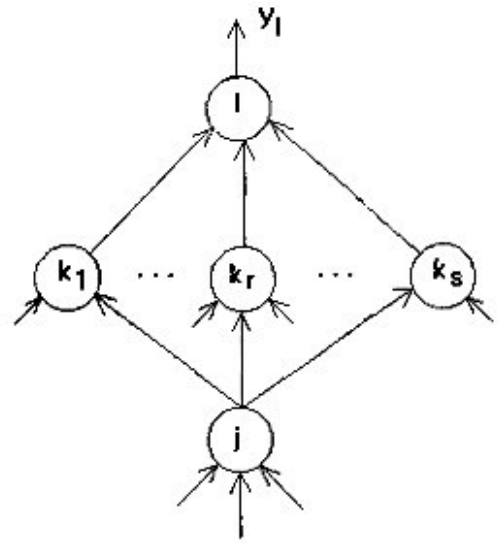

Fig. 2. The diagram of the possible fuzzy rules with identical antecedent for an output linguistic variable y_l.

Here $r = Arg\max_j(y_j^3 w_{kj})$, and P_l is the set of indices of the nodes in layer 4 which are connected to exactly one node l in layer 5.

2.1. Rule pruning of Shann and Fu

Initially, all possible fuzzy rules are considered. If there are s linguistic values for an output linguistic variable, then there are s rules with the same antecedent part but different consequents, which are inherently inconsistent. Let us consider the subnet in Fig. 2 which shows only the connections used for selecting the most relevant rule corresponding to the antecedent clause (IF-part) represented by the node j in layer 3. These rules are called incompatible rules. Fig. 2 corresponds to the following incompatible rules:

If $(antecedent)_j$ then y_l is $T_{r,s}(w_{k,j})$, $r = 1, 2, \ldots, s$.

$(antecedent)_j$ is the antecedent represented by the node j of layer 3. The certainty factors $w_{k,j}$ of the rules are shown in parenthesis.

For rule pruning the centroid of the set of incompatible rules is calculated as the output of node l in layer 5 considering only the connections shown in Fig. 2:

$$y_{l,j}^5 = \frac{\sum_{k \in I} y_k^4 a_{lk} c_{lk}}{\sum_{k \in I} y_k^4 a_{lk}} = c_{lj}, \tag{7}$$

where $y_k^4 = y_j^3 w_{kj}$, and I is the set of nodes in layer 4 connected to node l in layer 5 and node j in layer 3, i.e., the connections shown in Fig. 2.

Hence,

$$c_{lj} = \frac{\sum_{k\in I} y_j^3 w_{kj} a_{lk} c_{lk}}{\sum_{k\in I} y_j^3 w_{kj} a_{lk}} = \frac{\sum_{k\in I} w_{kj} a_{lk} c_{lk}}{\sum_{k\in I} w_{kj} a_{lk}}. \tag{8}$$

If the centroid c_{lj} is in the support of a linguistic value T_{r,y_i}, then the rules whose THEN-part contain T_{r,y_i} remains in the rule-base after pruning, and all other rules are eliminated. Thus, if the output fuzzy sets are defined on *nonoverlapped* intervals, the pruned rule-base will not have any incompatible rules. Such a rule-base with no incompatible rule is called a sound rule-base. When the output fuzzy sets are overlapped, for the same antecedent clause, Shann and Fu's scheme may suggest more than one consequents; i.e., more than one rule with the same antecedent but different consequents (incompatible rules). This is contradictory to the basic design philosophy of fuzzy systems. For smooth outputs, we need overlapped output fuzzy sets and in the present case this can result in incompatible fuzzy rules. If the reduced rule-base is sound, the connections corresponding to the pruned rules are set to unity, otherwise the reduced set is further trained.

2.2. Remarks on Shann and Fu model

The FNN model of Shann and Fu combines fuzzy logic and neural network in a systematic manner. The working process of a fuzzy logic control system is embedded in the layered structure of the network. Layers 3 and 4 along with their connecting links function as a connectionist inference engine, which avoids the rule-matching process. After the training, a rule pruning process is executed to delete redundant rules and a much smaller rule-base is obtained. After pruning the size of the network is also reduced by disconnecting the redundant links. Authors of [14] claimed that besides rule learning this network is flexible and extendible for learning other design parameters such as membership functions, fuzzy operators, etc.

Shann and Fu interpreted the link weights between layers 3 and 4 as the certainty factors of the associated rules, i.e., each of the rules is activated to a certain degree represented by the weight values. According to this interpretation, the link weights should always have *positive* values, and during the training phase, the link weights between layers 3 and 4 should be learnable *nonnegative* real numbers. In this regard, authors in [14] mentioned that during the training phase, the link weights w_{kj} in layer 4 are learnable nonnegative real numbers. But the EBP algorithm based on gradient descent search does NOT guarantee that the weights will always be adjusted to nonnegative real numbers even if the weights are initialized to nonnegative real values. This happens because of the dynamic nature of gradient descent learning which updates the weights keeping in view only the reduction of residual square error; it does not care what numerical values different connection weights take. Therefore, it can result in significantly large negative values even for majority of the connection weights. Our experiments indeed show that there are some weights with significant negative values. It appears that authors of [14] are also aware of this fact as they mentioned that *most* of the weights after training have nonzero positive values. Thus, as such the connection weights in FNNO *cannot* be interpreted as certainty factors of the associated rules.

The function of a node l in layer 5 is claimed to be based on the correlation-product inference and the fuzzy centroid defuzzification scheme. The equation for evaluating the centroid of the incompatible fuzzy rules is defined as:

$$c_{lj} = \sum_{k\in I} w_{kj} a_{lk} c_{lk} \Big/ \sum_{k\in I} w_{kj} a_{lk}. \tag{9}$$

This calculation is based on the function performed by the node l of layer 5. Since w_{kj} is unrestricted, the denominator of Eq. (9) may be even zero. Moreover, Eq. (9) may not be interpreted as the fuzzy centroid defuzzification scheme [7]. We shall discuss this later. However, it is a reasonable defuzzification scheme provided $w_{kj} > 0$.

After rule pruning, the fuzzy rule-base obtained may not be sound, i.e., may contain incompatible rules. The FNNO pruning scheme is such that the reduced rule-base will necessarily be sound only when the intervals defined for the linguistic values of each output linguistic variable are nonoverlapped. Thus, to ensure a sound rule-base in their scheme one has to choose the intervals for the linguistic values of each output linguistic variable in such a way that they do not overlap but cover the entire space of the corresponding output

linguistic variable. This appears to be an unnecessary restriction on system design and may hinder the performance of the system. As mentioned earlier, this is contradictory to the basic design philosophy of fuzzy systems, which demands overlapped output fuzzy sets for smooth output of the system. Consequently, in [14] authors used the fuzzy sets which are overlapped. But for the purpose of rule pruning they defined (by some means) a nonoverlapped partition for the output linguistic values.

Referring to Eq. (9) we find that $a_{lk} > 0$; c_{lk} unrestricted (could be positive, zero, or negative) and w_{kj} is also unrestricted in sign. Writing $w_{kj}a_{lk}$ as, say, q_{lk} the Eq. (9) becomes

$$c_{lj} = \sum_{k \in I} q_{lk} c_{lk} \Big/ \sum_{k \in I} q_{lk} = \sum_{k \in I} p_{lk} c_{lk}, \tag{10}$$

where $p_{lk} = q_{lk} / \sum_{k \in I} q_{lk}$. Now if c_{lk} is positive then a rule with the corresponding fuzzy set as the output variable should bias the defuzzified output towards c_{lk} when the rule is fired. At least the rule should have a positive contribution. But under the present scenario since w_{kj} could be negative also and if it is negative then p_{lk} will be negative and hence, the contribution of the corresponding rule becomes negative. In other words, it will bias the resultant centroid in the wrong direction. It is claimed [14] that since the knowledge of fuzzy rules learned by EBP algorithm is distributed over the adjustable weights, most of the weights after training should have nonzero *positive* values. In this regard, we emphasize that the EBP algorithm does not guarantee that most of the weights after training will have positive values. We already explained that most of the connection weights even may assume significantly large negative values. Shann and Fu obtained an encouraging result possibly because in their simulation they got most of the weights positive and the input output membership functions were tuned to the problem.

3. Proposed scheme – the FNNM

We used the same fuzzy neural network structure of FNNO. The functions of layers 1–3 are also the same. The functions of layers 4 and 5 are different from those of FNNO and are described next:

The output of node k in layer 4 is

$$y_k^4 = \max_{j \in I_k} (y_j^3 w_{kj}^2), \tag{11}$$

where I_k is the set of indices of the nodes in layer 3 that are connected to node k in layer 4. Here we took the square of the weights w_{kj} because it is always positive and we can interpret w_{kj}^2 as the strength of the rule whose IF-part is represented by the node j in layer 3 and THEN-part is represented by the node k in layer 4.

The output of node l in layer 5 is computed by

$$y_l^5 = \frac{\sum_{k \in I_l} y_k^4 a_{lk} c_{lk}}{\sum_{k \in I_l} y_k^4 a_{lk}}. \tag{12}$$

I_l, a_{lk} and c_{lk} are as defined in Section 2. Here $y_k^4 = \max_{j \in I_k} (y_j^3 w_{kj}^2)$. y_k^4 is always nonnegative real number, because y_j^3 is always nonnegative real number (it is a membership value). Thus FNNM eliminates the problems associated with FNNO, the original method of Shann and Fu. In other words, the counter intuitive characteristics of the defuzzification scheme adopted in [14] no longer exist. Note that, since nodes in layer 4 use w_{kj}^2 and hence nodes in layer 5 (i.e., the defuzzification scheme) also use w_{kj}^2, the network does not see the sign of w_{kj}. Even if w_{kj} attains a large negative value, as far as functioning of the net is concerned it sees only a large positive value. This is equivalent to constraining the search space for the connection weights of the learning algorithm to R^+. Thus the learning algorithm will look for the optimal solution in R^+. It may happen that the unconstrained weights (as in FNNO) can generate a lower value of the residual square error. But if we allow negative connection weights (note that for FNNM the connection weight is w_{kj}^2), the network will be like a black-box type function approximator and will lose the logical structure of the fuzzy reasoning system because of negative certainty factors.

3.1. EBP learning algorithm of FNNM

We now derive the learning algorithm for the FNNM with node functions defined in Section 3 using

backpropagation to minimize the error function,

$$e = \sum_X E = \frac{1}{2}\sum_X \sum_{l=1}^{q} (d_l - y_l^5)^2, \tag{13}$$

where q is the number of nodes in layer 5 and d_l and y_l are the target and actual outputs of the node l in layer 5 for an input X.

The weights w_{kj} (of the links from layer 3 to 4) are adjusted using gradient descent as

$$w_{kj}(t+1) = w_{kj}(t) - \eta\left(\frac{\partial E}{\partial w_{kj}}\right), \tag{14}$$

where $\eta > 0$ is the learning rate. We can write,

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial y_k^4}\frac{\partial y_k^4}{\partial w_{kj}} = \frac{\partial E}{\partial y_l^5}\frac{\partial y_l^5}{\partial y_k^4}\frac{\partial y_k^4}{\partial w_{kj}}. \tag{15}$$

From Eq. (13) we get,

$$\partial E/\partial y_l^5 = -(d_l - y_l^5). \tag{16}$$

Recalling Eq. (12), $y_l^5 = \sum(y_k^4 a_{lk} c_{lk})/\sum(y_k^4 a_{lk})$, we get

$$\begin{aligned}\frac{\partial y_l^5}{\partial y_k^4} &= \frac{(\sum_{k'} y_{k'}^4 a_{lk'})a_{lk}c_{lk} - (\sum_{k'} y_{k'}^4 a_{lk'}c_{lk'})a_{lk}}{(\sum_{k'} y_{k'}^4 a_{lk'})^2} \\ &= \frac{a_{lk}(c_{lk}(\sum_{k'} y_{k'}^4 a_{lk'}) - (\sum_{k'} y_{k'}^4 a_{lk'}c_{lk'}))}{(\sum_{k'} y_{k'}^4 a_{lk'})^2} \\ &= \frac{a_{lk}(c_{lk}(\sum_{k'} y_{k'}^4 a_{lk'}) - y_l^5(\sum_{k'} y_{k'}^4 a_{lk'}))}{(\sum_{k'} y_{k'}^4 a_{lk'})^2} \\ &= \frac{a_{lk}(c_{lk} - y_l^5)}{\sum_{k'} y_{k'}^4 a_{lk'}}.\end{aligned} \tag{17}$$

Here k' is the indices of the nodes in layer 4 which are connected to the node l in layer 5.

Considering Eq. (11), let

$$r = Arg\max_j(y_j^3 w_{kj}^2). \tag{18}$$

Then,

$$\frac{\partial y_k^4}{\partial w_{kj}} = \begin{cases} 2y_j^3 w_{kj} & \text{if } j = r, \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

Using Eqs. (16), (17), and (19), we can write Eq. (15) as

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial y_l^5}\frac{\partial y_l^5}{\partial y_k^4}\frac{\partial y_k^4}{\partial w_{kj}} =$$

$$\begin{cases} -(d_l - y_l^5)\dfrac{a_{lk}(c_{lk} - y_l^5)}{\sum_{k'} y_{k'}^4 a_{lk'}} 2y_j^3 w_{kj} & \text{if } j = r, \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

Hence weight update rule (14) becomes

$$w_{kj}(t+1) =$$

$$\begin{cases} w_{kj}(t) + \eta(d_l - y_l^5)\dfrac{a_{lk}(c_{lk} - y_l^5)}{\sum_{k'} y_{k'}^4 a_{lk'}} y_j^3 w_{kj} \\ \quad \text{if } j = r, \\ 0 \quad \text{otherwise.} \end{cases} \tag{21}$$

3.2. Rule pruning for FNNM

For the rule pruning we use a method similar (but not identical) to that of Shann and Fu. For every antecedent clause we compute the centroid of the corresponding set of s incompatible rules. Considering the subnet in Fig. 2 we compute the output of node l as

$$y_{l,j}^5 = \sum_{k \in I_{lj}} y_k^4 a_{lk} c_{lk} \Big/ \sum_{k \in I_{lj}} y_k^4 a_{lk}. \tag{22}$$

Since $y_k^4 = y_j^3 w_{kj}^2$,

$$\begin{aligned} y_{l,j}^5 &= \sum_{k \in I_l} y_j^3 w_{kj}^2 a_{lk} c_{lk} \Big/ \sum_{k \in I_l} y_j^3 w_{kj}^2 a_{lk} \\ &= \sum_{k \in I_l} w_{kj}^2 a_{lk} c_{lk} \Big/ \sum_{k \in I_l} w_{kj}^2 a_{lk}. \end{aligned} \tag{23}$$

$y_{l,j}^5$ in Eq. (23) can be viewed as the centroid c_{lj} of the set of incompatible fuzzy rules which corresponds to Fig. 2 with certainty factor w_{kj}^2 for the rule with antecedent node j and consequent node k. This centroid c_{lj} can now be used for rule reduction.

If c_{lj} falls in the support set or interval of t ($t > 0$) output linguistic values, then following the concept

of Shann and Fu, corresponding t rules may be retained. In this case the pruned rule set will have incompatible rules. To get around this, we suggest a different scheme for pruning. In this scheme we find the membership values of c_{jk} in all consequent fuzzy sets of the incompatible rules. Then the rule whose consequent has the highest membership value for c_{jk} is selected and the rest are deleted. We used 50% overlap (Fig. 4) between two consecutive fuzzy sets defined on the output linguistic variable. Hence, the centroid c_{ij} will be in the interval (support set) of two linguistic values T_{r,y_i} and T_{r+1,y_i}. If the membership of c_{ij} to T_{r,y_i} is more than that to T_{r+1,y_i} then we select the rule corresponding to T_{r,y_i}; otherwise, the rule with T_{r-1,y_i} is selected. Thus, in this scheme the pruned rule set will never have incompatible rules; i.e., will always be sound.

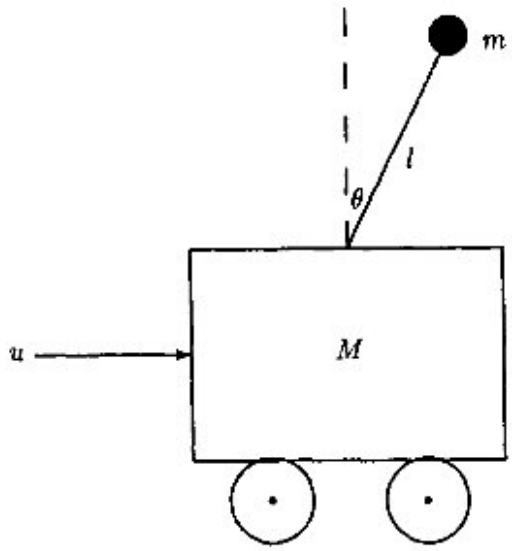

Fig. 3. The inverted pendulum.

4. Implementation and results

4.1. The problem of inverted pendulum

In this investigation we used the simple inverted pendulum (Fig. 3) as the control system because of its simplicity for computer simulation. The inverted pendulum is a rod of mass m supported through a hinge by a cart of mass M, where the rod motion is constrained to be on a vertical plane and the cart motion is constrained to be along the horizontal X direction. When the cart is at rest, the stick is in the vertical position and the force u is zero then the system is in equilibrium. This equilibrium position is unstable in the sense that with any perturbation from this position, no matter how small, the stick will fall down. For this system we have two types of fuzzy control rules – one for controlling the pole and the other for the cart. For simplicity, in this investigation we have considered only the pole balancing part. A typical rule for this has the form:

If θ is PB and $\dot{\theta}$ is PB then u is PB.

Here θ is the angular displacement, $\dot{\theta}$ is the angular velocity, u is the force applied. θ and $\dot{\theta}$ are the input linguistic variables and u is the output linguistic variable.

4.2. Computational protocols

We have used the following computational protocols: pole length 0.5 m; pole mass 0.1 kg; cart mass 2 kg; learning rate $\eta = 0.002$; the maximum allowable angular deviation is 0.18 rad and angular velocity is 1.8 rad/s. θ, $\dot{\theta}$, and u, each of these linguistic variables has seven linguistic values: NB, NM, NS, Z, PS, PM, and PB. The membership functions of the linguistic values of the linguistic variables are given in Fig. 4. In the absence of any preferential guidelines about the base and peak of different membership functions we used the most natural choice – isosceles triangles with 50% overlap with the neighboring membership functions. Each membership function has equal base-length. Thus our choice is the most unbiased choice for membership functions.

The number of possible fuzzy rules is 343 ($=7\times7\times7$). A target fuzzy rule-base, referred to as *standard rule-matrix*, and the corresponding control surface are shown in Fig. 5. This rule-matrix is used as a standard to compare the performance of FNNO and FNNM. The control surface produced by the membership functions in Fig. 4 and this rule-matrix, is regarded as the ideal control surface (Fig. 5(b)) and was used to generate the training samples. We

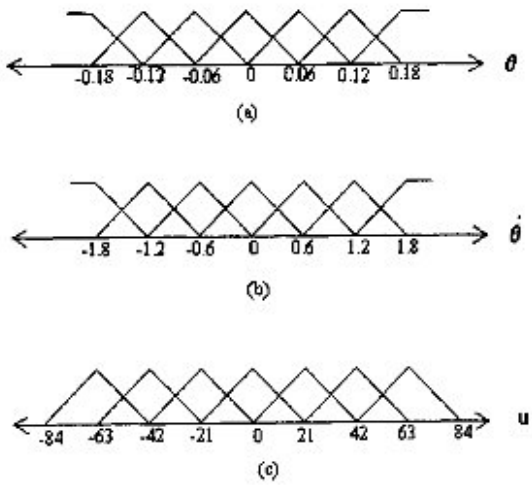

Fig. 4. Fuzzy membership functions of the linguistic values associated with (a) input linguistic variable θ (b) input linguistic variable $\dot{\theta}$ and (c) output linguistic variable u.

$\dot{\theta}$ / θ	NB	NM	NS	Z	PS	PM	PB
NB	NB	NM	NM	NS	NS	NS	NS
NM	NM	NM	NM	NS	NS	Z	Z
NS	NM	NM	NM	NS	Z	PS	PS
Z	NS	NS	Z	Z	Z	PS	PS
PS	NS	NS	Z	PS	PM	PM	PM
PM	Z	Z	PS	PS	PM	PM	PM
PB	PS	PS	PS	PS	PM	PM	PB

Fig. 5(a). The standard rule-matrix.

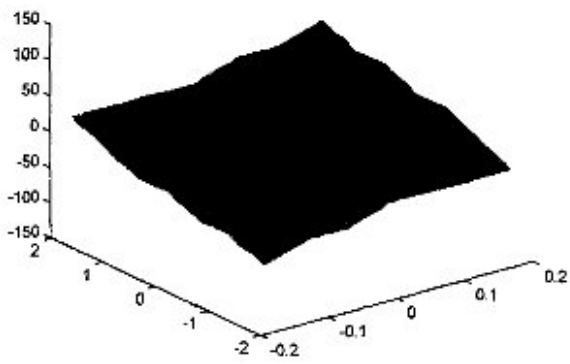

Fig. 5(b). Control surface of the standard matrix in Fig. 5(a).

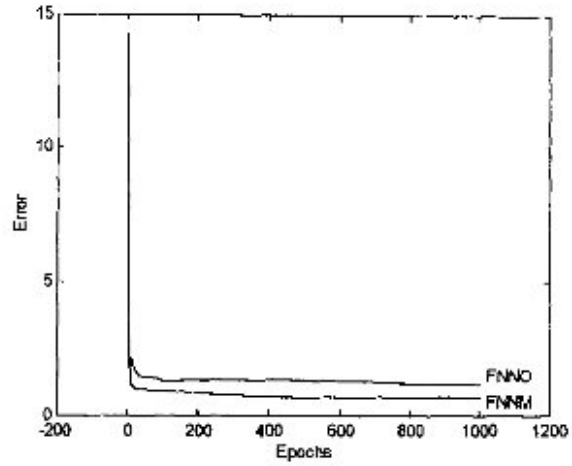

Fig. 6. Plot of error as a function of epochs (iterations) for FNNM and FNNO for the standard matrix in Fig. 5(a).

generated 1369 ($=37\times37$) training samples uniformly over the product space $\theta\times\dot{\theta}$. Hence, in the fuzzy neural network, for the inverted pendulum, there are 2 nodes (1 for θ and 1 for $\dot{\theta}$) in layer 1, 14 (7 for linguistic values of θ and 7 for linguistic values of $\dot{\theta}$) nodes in layer 2, 49 (for 7×7 different IF-part of rule base) nodes in layer 3, 7 (for 7 linguistic values of u) in layer 4, and 1 (for 1 output linguistic variable) in layer 5. There are 343 ($=49\times7$) links between layers 3 and 4 before pruning. Each link represents a possible fuzzy rule. After pruning the number of links becomes 49. For all results reported, the initial weights of the links between layers 3 and 4 are randomly selected in $[-1,+1]$.

4.3. Results

Fig. 6 depicts a typical illustration of the variation of training error with iteration for both FNNO and FNNM with the same learning rate $\eta=0.002$ and the same initialization of the network link weights. Fig. 6 reveals that, in the present case, the training error for FNNM reduces faster than that in FNNO. The steady-state error of our model is also much less than that for FNNO.

After training the rule pruning process was performed. Only 49 of the 343 initial fuzzy rules remained after pruning. For comparison of the two models, we generated pruned rule-matrix at different

θ \ $\dot{\theta}$	NB	NM	NS	Z	PS	PM	PB
NB	NM	NM	NM	NS	NS	Z	NS
NM	NM	NM	NM	NS	NS	Z	Z
NS	NM	NM	NM	NS	Z	PS	PS
Z	NS	NS	Z	Z	Z	PS	PS
PM	Z	Z	PS	PS	PM	PM	PM
PB	PS	PS	PS	PS	PM	PM	PB

Fig. 7(a). The rule-matrix generated by FNNM and standard rule-matrix after 5 training epochs.

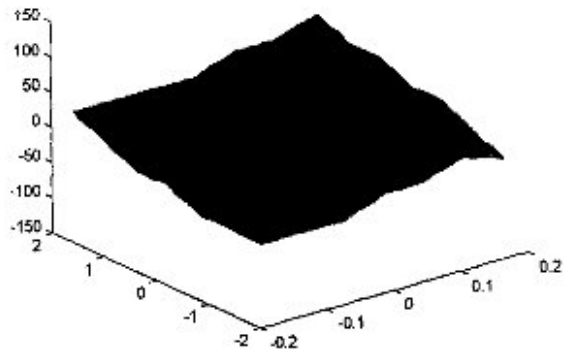

Fig. 7(b). Control surface produced by FNNM after 5 epochs for the standard matrix.

stages of training. Our proposed model reproduces exactly the *standard rule-matrix* after only 11 training epochs, whereas the model of Shann and Fu could not produce the *standard rule-matrix* even after 1000 training epochs. Fig. 7(a) shows the resulting rule-matrix generated by FNNM only after 5 epochs and the corresponding control surface is depicted in Fig. 7(b). Comparing rule matrix in Fig. 7(a) with the target, i.e., *standard rule-matrix* we find that there is not much difference between the two. This is also evident from the comparison of the control surface in Fig. 7(b) with that in Fig. 5(b). The rule-matrices produced by the pruning algorithm for FNNO after 5, 11, 100, and 1000 training epochs are shown in Fig. 8. The rule-matrix generated by FNNO even after 1000 training epochs mismatches the *standard rule-matrix* in six positions in which one significant position ($\theta = \text{PB}$, $\dot{\theta} = \text{PB}$) is highly different from the target. The target value is PB while FNNO suggested NB! To compare FNNO with FNNM, in Fig. 9 we have shown the plot of difference between the control

θ \ $\dot{\theta}$	NB	NM	NS	Z	PS	PM	PB
NB	NB	NM	NB	NS	NS	NS	NM
NM	NM	NM	NM	NS	Z	Z	Z
NS	NM	NM	NM	NS	Z	PS	PS
Z	NS	NS	Z	Z	Z	PS	PS
PS	NS	NS	Z	PS	PM	PM	PB
PM	PS	Z	PS	PS	PM	PM	PB
PB	PS	PM	PS	PS	PM	PB	NB

(a)

θ \ $\dot{\theta}$	NB	NM	NS	Z	PS	PM	PB
NB	NB	NM	NB	NS	NS	NS	NM
NM	NM	NM	NM	NS	NS	Z	Z
NS	NM	NM	NM	NS	Z	PS	PS
Z	NS	NS	Z	Z	Z	PS	PS
PS	NS	NS	Z	PS	PM	PM	PB
PM	PS	Z	PS	PS	PM	PM	PB
PB	PS	PM	PS	PS	PM	PB	NB

(b)

θ \ $\dot{\theta}$	NB	NM	NS	Z	PS	PM	PB
NB	NB	NM	NM	NS	NS	NS	NS
NM	NM	NM	NM	NS	Z	Z	PS
NS	NM	NM	NM	NS	Z	PS	PS
Z	NS	NS	Z	Z	Z	PS	PS
PS	NS	Z	Z	PS	PM	PM	PB
PM	PS	NS	PM	PS	PM	PM	NB
PB	PS	PM	PS	PS	PM	PB	NB

(c)

θ \ $\dot{\theta}$	NB	NM	NS	Z	PS	PM	PB
NB	NB	NM	NB	NS	NS	NS	NS
NM	NM	NM	NM	NS	NS	Z	Z
NS	NM	NM	NM	NS	Z	PS	PS
Z	NS	NS	Z	Z	Z	PS	PS
PS	NS	NS	Z	PS	PM	PM	PB
PM	Z	Z	PM	PS	PM	PM	PB
PB	PS	PM	PS	PS	PM	PB	NB

(d)

Fig. 8. The rule-matrices generated by FNNO and standard rule-matrix after (a) 5 training epochs. (b) 11 training epochs. (c) 100 training epochs and (d) 1000 training epochs.

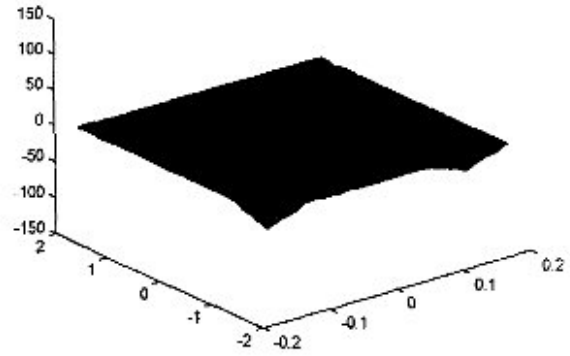

Fig. 9(ai). Difference surface produced by FNNM after 5 epochs for the standard matrix.

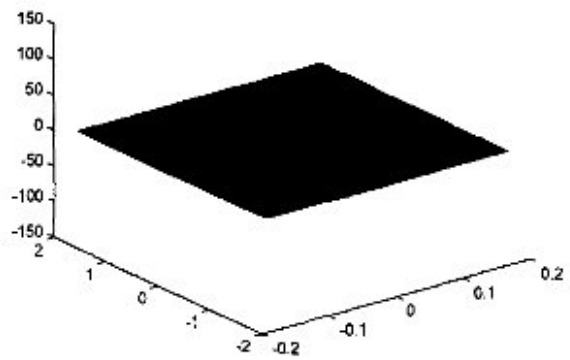

Fig. 9(aii). Difference surface produced by FNNM after 11 epochs for the standard matrix.

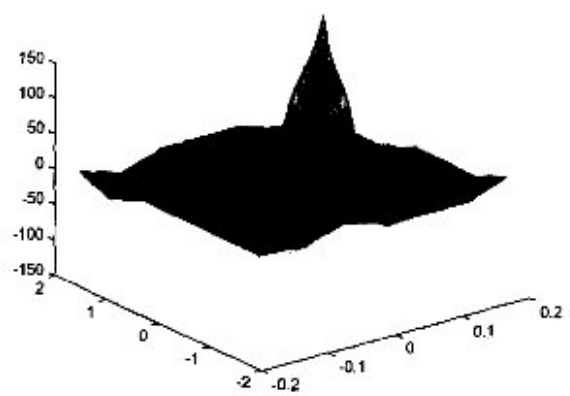

Fig. 9(bi). Difference surface produced by FNNO after 5 epochs for the standard matrix.

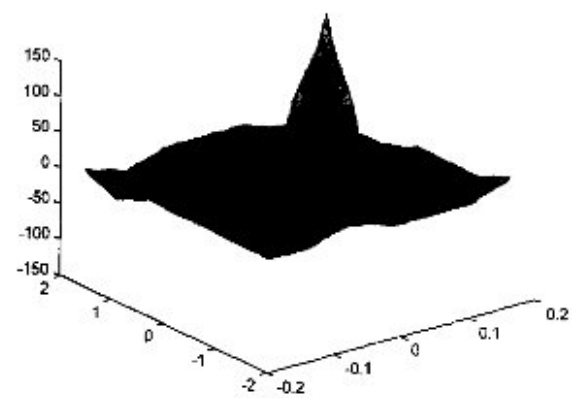

Fig. 9(bii). Difference surface produced by FNNO after 11 epochs for the standard matrix.

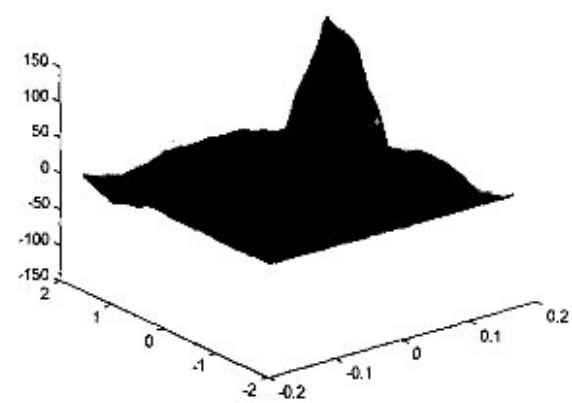

Fig. 9(biii). Difference surface produced by FNNO after 100 epochs for the standard matrix.

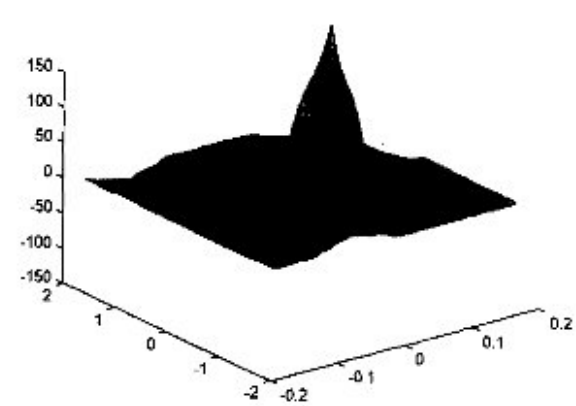

Fig. 9(biv). Difference surface produced by FNNO after 1000 epochs for the standard matrix.

$\dot{\theta}$ / θ	NB	NM	NS	Z	PS	PM	PB
NB	NB	NB	NM	NS	NS	Z	Z
NM	NB	NB	NM	NS	NS	Z	Z
NS	NM	NM	NM	NS	Z	PS	PS
Z	NS	NS	NS	Z	PS	PS	PS
PS	NS	NS	Z	PS	PM	PM	PM
PM	Z	Z	PS	PS	PM	PB	PB
PB	Z	Z	PS	PS	PM	PB	PB

Fig. 10(a). The close-matrix.

$\dot{\theta}$ / θ	NB	NM	NS	Z	PS	PM	PB
NB	PS	Z	NM	PB	Z	NS	Z
NM	NS	PM	PS	NB	Z	PB	PM
NS	PS	PB	Z	NB	NM	NS	NB
Z	NS	NM	PS	PB	NS	PS	PM
PS	PM	PS	Z	NM	PB	PM	PB
PM	PB	NB	NM	NS	PM	Z	NB
PB	NM	NS	PB	PM	Z	PS	Z

Fig. 10(b). The random-matrix.

$\dot{\theta}$ / θ	NB	NM	NS	Z	PS	PM	PB
NB	NB	NB	NS	NS	NS	Z	Z
NM	NB	NB	NM	NS	NS	Z	Z
NS	NM	NM	NM	NS	Z	PS	PS
Z	NS	NS	NS	Z	PM	PS	PS
PS	NS	Z	Z	PM	PM	PM	PB
PM	Z	Z	PS	PS	PM	PB	NB
PB	Z	Z	PS	PM	NB	NB	NB

Fig. 11. The rule-matrix generated by FNNO and close-matrix after 500 training epochs.

surfaces generated by (a) FNNM and the *standard rule-matrix*, and (b) FNNO and the *standard rule-matrix* after different epochs of training. Note that for FNNM we have shown the difference surfaces only with 5 and 11 epochs as FNNM can produce the target rule-matrix with 11 iterations. Fig. 9 clearly shows that FNNM outperforms FNNO for the standard target rule-matrix.

In order to see the ability and effectiveness of the proposed model for other target rule-matrices besides the *standard rule-matrix* just discussed, we trained

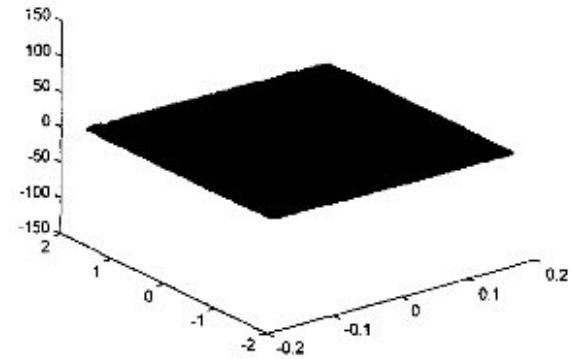

Fig. 12(a). Difference surface produced by FNNM after 10 epochs for the close matrix.

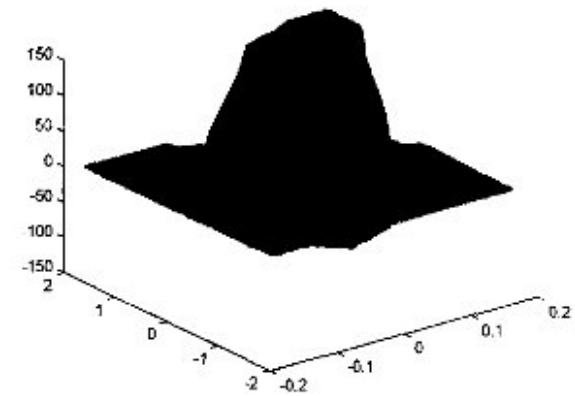

Fig. 12(b). Difference surface produced by FNNO after 500 epochs for the close matrix.

both FNNO and FNNM with the same training samples generated by the following two other target rule-matrices: (1) a rule-matrix (Fig. 10(a)) very close to the *standard rule-matrix*. We call this rule-matrix as *close-matrix*. (2) a very inconsistent rule-matrix produced by randomly selecting the consequent fuzzy sets of each rule. This rule-matrix is not able to balance the inverted pendulum from any initial position. We call this matrix as *random-matrix*. It is shown in Fig. 10(b).

FNNM is found to reproduce the *close-matrix* just with 10 iterations; while FNNO cannot produce the target rule-matrix even with 1000 iterations. Fig. 12(a) depicts the difference surface for FNNM after only 10 epochs, while Fig. 12(b) shows the same for the rule-matrix (Fig. 11) selected by FNNO after 500 epochs of training.

θ \ $\dot{\theta}$	NB	NM	NS	Z	PS	PM	PB
NB	PS	Z	NS	PM	Z	NS	Z
NM	NS	PM	PS	NB	Z	PM	PM
NS	PS	PB	Z	NB	NM	NS	NB
Z	NS	NS	PS	PM	NS	PS	PM
PS	PM	PS	Z	NM	PB	PM	PB
PM	PB	NM	NM	NS	PM	Z	NB
PB	NM	NS	PB	PM	Z	PS	Z

Fig. 13(a). The rule-matrix generated by FNNM and random-matrix after 500 training epochs.

θ \ $\dot{\theta}$	NB	NM	NS	Z	PS	PM	PB
NB	PS	Z	NM	PB	Z	NS	Z
NM	NS	PM	PS	NB	Z	PB	PM
NS	PS	PB	Z	NB	NM	NS	NB
Z	NS	NM	PS	PB	NS	PS	PM
PS	PM	PS	Z	NM	PB	PM	PB
PM	NB	NB	NM	NS	PM	Z	NB
PB	NM	NS	NB	PM	Z	PS	Z

Fig. 13(b). The rule-matrix generated by FNNO and random-matrix after 500 training epochs.

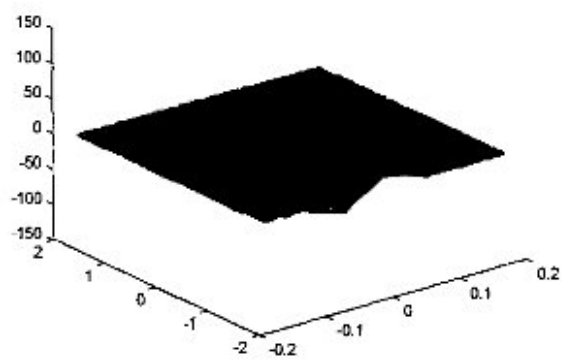

Fig. 14(a). Difference surface produced by FNNM after 500 epochs for the random matrix.

In case of *random-matrix*, our model takes more training epochs compared to other two cases discussed earlier. After 500 training epochs we get the rule-matrix shown in Fig. 13(a). Though the number of positions in which mismatches occur are six, the values

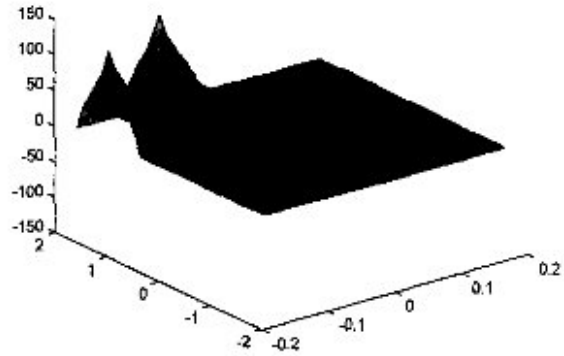

Fig. 14(b). Difference surface produced by FNNO after 500 epochs for the random matrix.

are slightly different from the target *random-matrix*. On the other hand, the rule-matrix produced by FNNO in the same situation mismatches the target *random-matrix* only at two positions (Fig. 13(b)), but the difference in values at these two positions are maximum (in both cases the target fuzzy set was PB and FNNO selects NB). Consequently, the control surfaces are expected to be significantly different. Fig. 14 clearly shows that this is indeed the case. Fig. 14(a) shows the difference surface for the rule-matrix, selected by FNNM after 500 epochs of training, and Fig. 14(b) depicts the same for the rule-matrix, selected by FNNO after the same number of training iterations.

5. Conclusions

Shann and Fu proposed a fuzzy neural network for rule selection. In this paper, we first addressed different limitations of the FNNO and provided solutions for the same. In FNNO the trainable connection weights could be negative and hence its use as certainty factors led to a counter-intuitive defuzzification scheme. We have avoided these problems by using square of the connection weights as certainty factors and incorporating that into the defuzzification as well as in the learning and pruning schemes. The required learning rules have also been derived. We also proposed a rule pruning scheme, which unlike FNNO, will always produce a set of compatible rules. The superiority of FNNM over FNNO has been established for the inverted pendulum problem.

Acknowledgements

We gratefully acknowledge Mr. R.K. De, Mr. S. Sinha and Mr. B. Umashankar for their help and co-operation. We are also grateful to the anonymous referees for their valuable suggestions.

References

[1] S. Abe, M.S. Lan, A method for fuzzy rules extraction directly from numerical data and its application to pattern recognition, IEEE Trans. Fuzzy Systems 3 (1) (1995) 18–28.

[2] J.M. Benitez, A. Blanco, I. Requena, An empirical procedure to obtain fuzzy rules using neural networks, in: Proc. VII IFSA World Congr., 95, Sao Paulo, Brazil, 1995, pp. 663–666.

[3] D. Driankov, H. Hellendoorn, M. Reinfrank, An Introduction to Fuzzy Control, Springer, New York, 1993.

[4] H. Ishibuchi, R. Fujioka, H. Tanaka, Neural networks that learn from fuzzy if then rules, IEEE Trans. Fuzzy Systems 1 (2) (1993) 85–97.

[5] G.J. Klir, B. Yuan, Fuzzy Sets and Fuzzy Logic – Theory and Applications, Prentice-Hall, Englewood Cliffs, NJ, 1995.

[6] T. Kohonen, Self-organization and Associative Memory, Springer, Berlin, 1998.

[7] B. Kosko, Neural Networks and Fuzzy Systems, Prentice-Hall of India, New Delhi, 1994.

[8] C.C. Lee, Fuzzy logic in control system: fuzzy logic controller – part i, IEEE Trans. System Man Cybernet. SMC-20 (1990) 404–418.

[9] C.C. Lee, Fuzzy logic in control system: fuzzy logic controller – part ii, IEEE Trans. System Man Cybernet. SMC-20 (1990) 419–435.

[10] K. Lee, D. Kwang, H.L. Wang, A fuzzy neural network model for fuzzy inference and rule tuning, Internat. J. Uncertainty, Fuzziness Knowledge-Based Systems 2 (3) (1994) 265–277.

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

[12] C.C. Li, C.J. Wu, Generating fuzzy rules for a neural fuzzy classifier, in: Proc. 3rd IEEE Internat. Conf. on Fuzzy Systems FUZZ IEEE '94, Orlando, 1994, pp. 1719–1724.

[13] S.K. Pal, N.R. Pal, Soft computing: goals, tools and feasibility, J. IETE 42 (4–5) (1996) 195–204.

[14] J.J. Shann, H.C. Fu, A fuzzy neural network for rule acquiring on fuzzy control system, Fuzzy Sets and Systems 71 (1995) 345–357.

[15] S. Yao, C. Wei, Z. He, Evolving fuzzy neural networks for extracting rules, in: Proc. 5th IEEE Internat. Conf. on Fuzzy Systems FUZZ IEEE '96, New Orleans, 1996, pp. 361–367.