Design of Fuzzy Neural Network for Function Approximation and Classification

Amit Mishra, Zaheeruddin *†

Abstract— **A hybrid Fuzzy Neural Network (FNN) system is presented in this paper. The proposed FNN can handle numeric and fuzzy inputs simultaneously. The numeric inputs are fuzzified by input nodes upon presentation to the network while the fuzzy inputs do not require this translation. The connections between input to hidden nodes represent rule antecedents and hidden to output nodes represent rule consequents. All the connections are represented by Gaussian fuzzy sets. The mutual subsethood measure for fuzzy sets that indicates the degree to which the two fuzzy sets are equal and is used as a method of activation spread in the network. A volume based defuzzification method is used to compute the numeric output of the network. The training of the network is done using gradient descent learning procedure. The model has been tested on three benchmark problems i.e. sine−cosine and Narazaki Ralescu's function for approximation and Iris flower data for classification. Results are also compared with existing schemes and the proposed model shows its natural capability as a function approximator, and classifier.**

Keywords: ***Cardinality, classifier, function approximation, fuzzy neural system, mutual subsethood***

1 Introduction

The conventional approaches to system modeling that are based on mathematical tools (i.e. difference equations) perform poorly in dealing with complex and uncertain systems. The basic reason is that, most of the time; it is very difficult to find a global function or analytical structure for a nonlinear system. In contrast, fuzzy logic provides an inference morphology that enables approximate human reasoning capability to be applied in a fuzzy inference system. Therefore, a fuzzy inference system employing fuzzy logical rules can model the quantitative aspects of human knowledge and reasoning processes without employing precise quantitative analysis.

*Jaypee Institute of Engineering and Technology, A.B. Road, Raghogarh, Distt.Guna, Madhya Pradesh, India, PIN-473226. Email: amitutk@ gmail.com

†Jamia Millia Islamia (A Central University), Department of Electrical Engineering, Jamia Nagar, New Delhi, India, PIN-110 025. Email: zaheer_ 2k@ hotmail.com

In recent past, artificial neural network has also played an important role in solving many engineering problems [1], [2]. Neural network has advantages such as learning, adaption, fault tolerance, parallelism, and generalization. The fuzzy system utilizing the learning capability of neural networks can successfully construct the input output mapping for many applications [3], [4]. The benefits of combining fuzzy logic and neural network have been explored extensively in the literature [5], [6], [7], [8], [9].
The term neuro-fuzzy system (also neuro-fuzzy methods or models) refers to combinations of techniques from neural networks and fuzzy system [10], [11], [12], [13], [14]. This does not mean that a neural network and a fuzzy system are used in some kind of combination, but a fuzzy system is created from data by some kind of (heuristic) learning method, motivated by learning procedures used in neural networks. The neuro-fuzzy methods are usually applied, if a fuzzy system is required to solve a problem of function approximation−or a special case of it, like, control or classification [15], [16], [17], [18], [19]−and the otherwise manual design process should be supported and replaced by an automatic learning process.
In this paper, the attention has been focused on the function approximation and classification capabilities of the subsethood based fuzzy neural model (subsethood based FNN). This model can handle simultaneous admission of fuzzy or numeric inputs along with the integration of a fuzzy mutual subsethood measure for activity propagation. A product aggregation operator computes the strength of firing of a rule as a fuzzy inner product and works in conjunction with volume defuzzification to generate numeric outputs. A gradient descent algorithm allows the model to fine tune rules with the help of numeric data.
The organization of the paper is as follows: Section 2 presents the architectural and operational detail of the model. Section 3 describes the gradient descent learning procedure for training the model. Section 4 and Section 5 shows the experiment results for three benchmark problems based on function approximation and classification. Finally, the Section 6 concludes the paper.

2 Fuzzy Neural Network system

The proposed architecture of subsethood based Fuzzy neural network is shown in Fig. 1. Here x_1 to x_m and

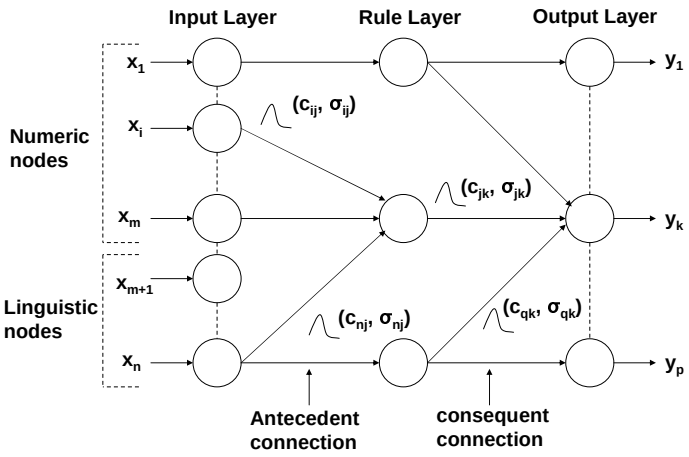

Figure 1: Architecture of subsethood based FNN model.

x_{m+1} to x_n are numeric and linguistic inputs respectively. Each hidden node represents a rule, and input-hidden node connection represents fuzzy rule antecedent. Each hidden-output node connection represents a fuzzy rule consequent. Fuzzy set corresponding to linguistic levels of fuzzy *if-then* rules are defined on input and output UODs and are represented by symmetric Gaussian membership functions specified by a center c and spread σ. The center and spread of fuzzy weights w_{ij} from input nodes i to rule nodes j are shown as c_{ij} and σ_{ij} of a Gaussian fuzzy set and denoted by $w_{ij} = (c_{ij}, \sigma_{ij})$. In a similar way, consequent fuzzy weights from rule nodes j to output nodes k are denoted by $v_{jk} = (c_{jk}, \sigma_{jk})$. Here y_1 to $y_k \ldots y_p$ are the outputs of the subsethood based FNN model.

2.1 Signal Transmission at Input Nodes

In the proposed FNN the input features $x_1, ..., x_n$ can be either linguistic or numeric or the combination of both. Therefore two kinds of nodes may present in the input layer of the network corresponding to the nature of input features.

Linguistic nodes accept the linguistic inputs represented by a fuzzy sets with a Gaussian membership function and modeled by a center c_i and spread σ_i. These linguistic inputs can be drawn from pre-specified fuzzy sets as shown in Fig. 2, where three Gaussian fuzzy sets have been defined on the universe of discourse (UODs) [-1,1]. Thus, a linguistic input feature x_i is represented by the pair of center and spread (c_i, σ_i). No transformation of inputs takes place at linguistic nodes in the input layer. They merely transmit the fuzzy input forward along antecedent weights.

Numeric nodes accept numeric inputs and fuzzify them into Gaussian fuzzy sets. The numeric input is fuzzified by treating it as the centre of a Gaussian membership function with a heuristically chosen spread. An example of this fuzzification process is shown in Fig. 3, where a numeric feature value of 0.3 has been fuzzified

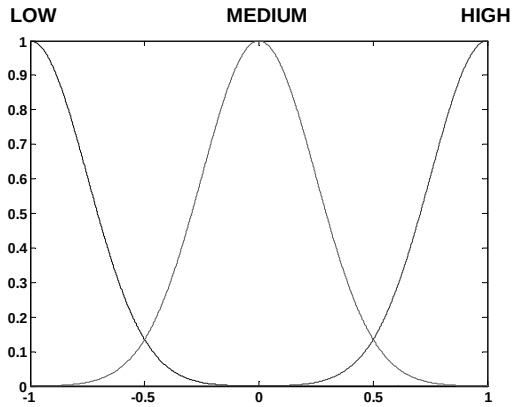

Figure 2: Fuzzy sets for fuzzy inputs.

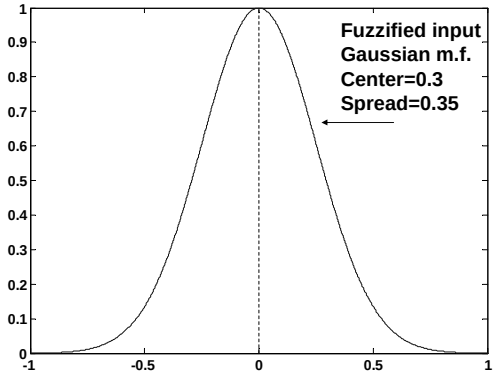

Figure 3: Fuzzification of numeric input.

into a Gaussian membership function centered at 0.3 with spread 0.35. The Gaussian shape is chosen to match the Gaussian shape of weight fuzzy sets since this facilitates subsethood calculations detailed in section 2.2.

Therefore, the signal from a numeric node of the input layer is represented by the pair (c_i, σ_i). Antecedent connections uniformly receive signals of the form (c_i, σ_i). Signals $(S(x_i) = (c_i, \sigma_i))$ are transmitted to hidden rule nodes through fuzzy weights w_{ij} also of the form (c_{ij}, σ_{ij}), where single subscript notation has been adopted for the input sets and the double subscript for the weight sets.

2.2 Signal Transmission from Input to Rule Nodes (Mutual Subsethood Method)

Since both the signal and the weight are fuzzy sets, being represented by Gaussian membership function, there is a need to quantify the net value of the signal transmitted along the weight by the extent of overlap between the two fuzzy sets. This is measured by their *mutual subsethood* [20]. Consider two fuzzy sets A and B with centers c_1, c_2 and spreads σ_1, σ_2 respectively. These sets are expressed by their membership functions as:

$$a(x) = e^{-((x-c_1)/\sigma_1)^2}. \quad (1)$$

$$b(x) = e^{-((x-c_2)/\sigma_2)^2}. \quad (2)$$

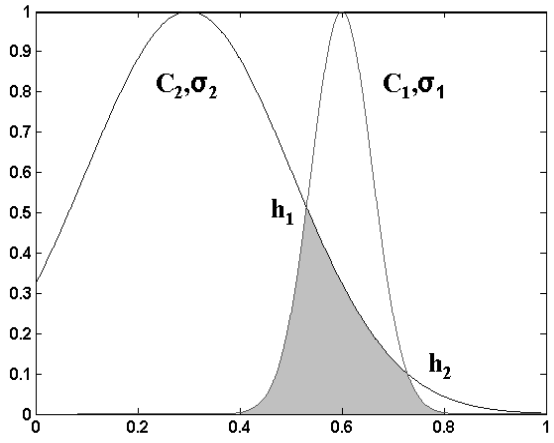

Figure 4: Example of overlapping: $c_1 > c_2$ and $\sigma_1 < \sigma_2$.

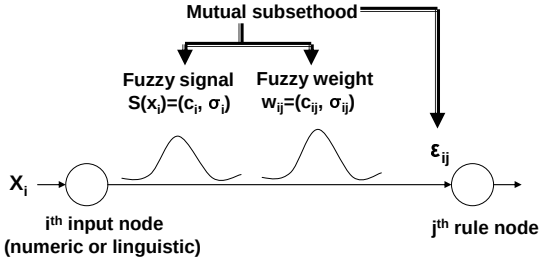

Figure 5: Fuzzy signal transmission.

The cardinality $C(A)$ of fuzzy set A is defined by

$$C(A) = \int_{-\infty}^{\infty} a(x)dx = \int_{-\infty}^{\infty} e^{-((x-c_1)/\sigma_1)^2}dx. \quad (3)$$

Then the mutual subsethood $\mathcal{E}(A,B)$ of fuzzy sets A and B measures the extent to which fuzzy set A equals fuzzy set B can be evaluated as:

$$\mathcal{E}(A,B) = \frac{C(A \cap B)}{C(A) + C(B) - C(A \cap B)}. \quad (4)$$

Further detail on the mutual subsethood measure can be found in [20]. Depending upon the relative values of centers and spreads of fuzzy sets A and B (nature of overlap), the four possible different cases are as follows:
case 1: $c_1 = c_2$ having any values of σ_1 and σ_2.
case 2: $c_1 \neq c_2$ and $\sigma_1 = \sigma_2$.
case 3: $c_1 \neq c_2$ and $\sigma_1 > \sigma_2$.
case 4: $c_1 \neq c_2$ and $\sigma_1 < \sigma_2$.
In case 1, the two fuzzy sets do not cross over−either one fuzzy set belongs completely to the other or two fuzzy sets are identical. In case 2, there is exactly one cross over point, whereas in cases 3 and 4, there are exactly two crossover points. An example of case 4 type overlap is shown in Fig. 4. To calculate the crossover points, by setting $a(x) = b(x)$, the two cross over points h_1 and h_2 yield as,

$$h_1 = \frac{c_1 + \frac{\sigma_1}{\sigma_2}c_2}{1 + \frac{\sigma_1}{\sigma_2}}, \quad (5)$$

$$h_2 = \frac{c_1 - \frac{\sigma_1}{\sigma_2}c_2}{1 - \frac{\sigma_1}{\sigma_2}}. \quad (6)$$

These values of h_1 and h_2 are used to calculate the mutual subsethood $\mathcal{E}(A,B)$ based on $C(A \cap B)$, as defined in (4). Symbolically, for a signal $s_i = S(x_i) = (c_i, \sigma_i)$ and fuzzy weight $w_{ij} = (c_{ij}, \sigma_{ij})$, the mutual subsethood is

$$\mathcal{E}_{ij} = \mathcal{E}(s_i, w_{ij}) = \frac{C(s_i \cap w_{ij})}{C(s_i) + C(w_{ij}) - C(s_i \cap w_{ij})}. \quad (7)$$

As shown in Fig. 5, in the subsethood based FNN model, a fuzzy input signal is transmitted along a fuzzy weight that represents an antecedent connection. The transmitted signal is quantified by $\mathcal{E}_{ij}$, which denotes the mutual subsethood between the fuzzy signal $S(x_i)$ and fuzzy weight (c_{ij}, σ_{ij}) and can be computed using (4).
The expression for cardinality can be evaluated for each of the four cases in terms of standard error function $erf(x)$ represented as (8).

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}dt. \quad (8)$$

The expressions for $C(s_i \cap w_{ij})$ for all the four cases identified above are evaluated in Appendix (A) seperately.

2.3 Activity Aggregation at Rule Nodes (Product Operator)

The net activation z_j of the rule node j is a product of all mutual subsethoods known as the fuzzy inner product can be evaluated as

$$z_j = \prod_{i=1}^{n} \mathcal{E}_{ij} = \prod_{i=1}^{n} \mathcal{E}(S(x_i), w_{ij}) \quad (9)$$

The inner product operator in (9) exhibits following properties: it is bounded between 0 and 1; monotonic increasing; continuous and symmetric.
The signal function for the rule node is linear

$$S(z_j) = z_j. \quad (10)$$

Numeric activation values are transmitted unchanged to consequent connections.

2.4 Output Layer Signal Computation (Volume Defuzzification)

The signal of each output node is determined using standard volume based centroid defuzzification [20]. The activation of the output node is y_k, and V_{jk}'s denote consequent set volumes, then the general expression of defuzzification is

$$y_k = \frac{\sum_{j=1}^{q} z_j c_{jk} V_{jk}}{\sum_{j=1}^{q} z_j V_{jk}}. \quad (11)$$

The volume V_{jk} is simply the area of consequent fuzzy sets which are represented by Gaussian membership function. From (11), the output can be evaluated as

$$y_k = \frac{\sum_{j=1}^{q} z_j c_{jk} \sigma_{jk}}{\sum_{j=1}^{q} z_j \sigma_{jk}}. \tag{12}$$

The signal of output node k is linear i.e. $S(y_k) = y_k$.

3 Supervised learning (Gradient descent algorithm)

The subsethood based linguistic network is trained by supervised learning. This involves repeated presentation of a set of input patterns drawn from the training set. The output of the network is compared with the desired value to obtain the error, and network weights are changed on the basis of an error minimization criterion. Once the network is trained to the desired level of error, it is tested by presenting a new set of input patterns drawn from the testing set.

3.1 Update Equations for Free Parameters

Learning is incorporated into the subsethood−linguistic model using the gradient descent method [15], [21]. A squared error criterion is used as a training performance parameter. The squared error e^t at iteration t is computed in the standard way

$$e^t = \frac{1}{2} \sum_{k=1}^{p} (d_k^t - S(y_k^t))^2. \tag{13}$$

where d_k^t is the desired value at output node k, and the error evaluated over all p outputs for a specific pattern k. Both the centers and spreads $c_{ij}, c_{jk}, \sigma_{ij}$ and σ_{jk} of antecedents and consequent connections are modified on the basis of update equations given as follows:

$$c_{ij}^{t+1} = c_{ij}^t - \eta \frac{\partial e^t}{\partial c_{ij}^t} + \alpha \triangle c_{ij}^{t-1}. \tag{14}$$

where η is the learning rate, α is the momentum parameter, and

$$\triangle c_{ij}^{t-1} = c_{ij}^t - c_{ij}^{t-1}. \tag{15}$$

3.2 Partial Derivatives Evaluation

The expressions of partial derivatives required in these update equations are derived as follows:
For the error derivative with respect to consequent centers

$$\frac{\partial e}{\partial c_{jk}} = \frac{\partial e}{\partial y_k} \frac{\partial y_k}{\partial c_{jk}} = -(d_k - y_k) \frac{z_j \sigma_{jk}}{\sum_{j=1}^{q} z_j \sigma_{jk}} \tag{16}$$

and the error derivative with respect to the consequent spreads

$$\begin{aligned} \frac{\partial e}{\partial \sigma_{jk}} &= \frac{\partial e}{\partial y_k} \frac{\partial y_k}{\partial \sigma_{jk}} \\ &= -(d_k - y_k) \\ &\quad \left\{ \frac{z_j c_{jk} \sum_{j=1}^{q} z_j \sigma_{jk} - z_j \sum_{j=1}^{q} z_j c_{jk} \sigma_{jk}}{(\sum_{j=1}^{q} z_j \sigma_{jk})^2} \right\}. \end{aligned} \tag{17}$$

The error derivatives with respect to antecedent centers and spreads involve subsethood derivatives in the chain and are somewhat more involved to evaluate. Specifically, the error derivative chains with respect to antecedent centers and spreads are given as following,

$$\begin{aligned} \frac{\partial e}{\partial c_{ij}} &= \sum_{k=1}^{p} \frac{\partial e}{\partial y_k} \frac{\partial y_k}{\partial z_j} \frac{\partial z_j}{\partial \mathcal{E}_{ij}} \frac{\partial \mathcal{E}_{ij}}{\partial c_{ij}} \\ &= \sum_{k=1}^{p} -(d_k - y_k) \frac{\partial y_k}{\partial z_j} \frac{\partial z_j}{\partial \mathcal{E}_{ij}} \frac{\partial \mathcal{E}_{ij}}{\partial c_{ij}}, \end{aligned} \tag{18}$$

$$\begin{aligned} \frac{\partial e}{\partial \sigma_{ij}} &= \sum_{k=1}^{p} \frac{\partial e}{\partial y_k} \frac{\partial y_k}{\partial z_j} \frac{\partial z_j}{\partial \mathcal{E}_{ij}} \frac{\partial \mathcal{E}_{ij}}{\partial \sigma_{ij}} \\ &= \sum_{k=1}^{p} -(d_k - y_k) \frac{\partial y_k}{\partial z_j} \frac{\partial z_j}{\partial \mathcal{E}_{ij}} \frac{\partial \mathcal{E}_{ij}}{\partial \sigma_{ij}}, \end{aligned} \tag{19}$$

and the error derivative chains with respect to input feature spread is evaluated as

$$\begin{aligned} \frac{\partial e}{\partial \sigma_i} &= \sum_{j=1}^{q} \sum_{k=1}^{p} \frac{\partial e}{\partial y_k} \frac{\partial y_k}{\partial z_j} \frac{\partial z_j}{\partial \mathcal{E}_{ij}} \frac{\partial \mathcal{E}_{ij}}{\partial \sigma_i} \\ &= \sum_{j=1}^{q} \sum_{k=1}^{p} -(d_k - y_k) \frac{\partial y_k}{\partial z_j} \frac{\partial z_j}{\partial \mathcal{E}_{ij}} \frac{\partial \mathcal{E}_{ij}}{\partial \sigma_i}. \end{aligned} \tag{20}$$

where

$$\begin{aligned} \frac{\partial y_k}{\partial z_j} &= \frac{\sigma_{jk} c_{jk} \sum_{j=1}^{q} z_j \sigma_{jk} - \sigma_{jk} \sum_{j=1}^{q} z_j c_{jk} \sigma_{jk}}{(\sum_{j=1}^{q} z_j \sigma_{jk})^2} \\ &= \sigma_{jk} \left[\frac{c_{jk} \sum_{j=1}^{q} z_j \sigma_{jk} - \sum_{j=1}^{q} z_j c_{jk} \sigma_{jk}}{(\sum_{j=1}^{q} z_j \sigma_{jk})^2} \right] \\ &= \frac{\sigma_{jk}(c_{jk} - y_k)}{\sum_{j=1}^{q} z_j \sigma_{jk}} \end{aligned} \tag{21}$$

and

$$\frac{\partial z_j}{\partial \mathcal{E}_{ij}} = \prod_{i=1, i \neq j}^{n} \mathcal{E}_{ij}. \tag{22}$$

The expressions for antecedent connection, mutual subsethood partial derivatives $\frac{\partial \mathcal{E}_{ij}}{\partial c_{ij}}$ and $\frac{\partial \mathcal{E}_{ij}}{\partial \sigma_{ij}}$ are obtained by differentiating (7) with respect to c_{ij}, σ_{ij} and σ_i as in (23), (24) and (25). In these equations, the calculation of $\partial C(s_i \cap w_{ij})/\partial c_{ij}$ and $\partial C(s_i \cap w_{ij})/\partial \sigma_{ij}$ depends on the nature of overlap of the input feature fuzzy set and weight fuzzy set, i.e. upon the values of c_{ij}, c_i, σ_{ij} and σ_i.

$$\frac{\partial \mathcal{E}_{ij}}{\partial c_{ij}} = \left[\frac{\left(\frac{\partial C(s_i \cap w_{ij})}{\partial c_{ij}} \left(\sqrt{\pi}(\sigma_i + \sigma_{ij}) - C(s_i \cap w_{ij}) \right) \right)}{\left(\sqrt{\pi}(\sigma_i + \sigma_{ij}) - C(s_i \cap w_{ij}) \right)^2} \right.$$

$$- \frac{-\left(\frac{-\partial C(s_i \cap w_{ij})}{\partial c_{ij}} C(s_i \cap w_{ij})\right)}{\left(\sqrt{\pi}(\sigma_i + \sigma_{ij}) - C(s_i \cap w_{ij})\right)^2} \Bigg], \quad (23)$$

$$\frac{\partial \mathcal{E}_{ij}}{\partial \sigma_{ij}} = \Bigg[\frac{\left(\frac{\partial C(s_i \cap w_{ij})}{\partial \sigma_{ij}} \left(\sqrt{\pi}(\sigma_i + \sigma_{ij}) - C(s_i \cap w_{ij})\right)\right)}{\left(\sqrt{\pi}(\sigma_i + \sigma_{ij}) - C(s_i \cap w_{ij})\right)^2} - \frac{\left(\left(\sqrt{\pi} - \frac{\partial C(s_i \cap w_{ij})}{\partial \sigma_{ij}}\right) C(s_i \cap w_{ij})\right)}{\left(\sqrt{\pi}(\sigma_i + \sigma_{ij}) - C(s_i \cap w_{ij})\right)^2} \Bigg], \quad (24)$$

and

$$\frac{\partial \mathcal{E}_{ij}}{\partial \sigma_i} = \Bigg[\frac{\left(\frac{\partial C(s_i \cap w_{ij})}{\partial \sigma_i} \left(\sqrt{\pi}(\sigma_{ij} + \sigma_i) - C(s_i \cap w_{ij})\right)\right)}{\left(\sqrt{\pi}(\sigma_{ij} + \sigma_i) - C(s_i \cap w_{ij})\right)^2} - \frac{\left(\left(\sqrt{\pi} - \frac{\partial C(s_i \cap w_{ij})}{\partial \sigma_i}\right) C(s_i \cap w_{ij})\right)}{\left(\sqrt{\pi}(\sigma_{ij} + \sigma_i) - C(s_i \cap w_{ij})\right)^2} \Bigg]. \quad (25)$$

In (23), (24) and (25), the calculation of $\partial C(s_i \cap w_{ij})/\partial c_{ij}$, $\partial C(s_i \cap w_{ij})/\partial \sigma_{ij}$ and $\partial C(s_i \cap w_{ij})/\partial \sigma_i$ is required, which depends on the nature of overlap. The case wise expressions of the above terms are evaluated in Appendix (B) seperately.

4 Function approximation

Function approximation involves determining or learning the input-output relations using numeric input-output data. Conventional methods like linear regression are useful in cases where the relation being learnt, is linear or quasi-linear. For nonlinear function approximation multilayer neural networks are well suited to solve the problem but at the same time they also experience the drawback of their black box nature and heuristic decisions regarding the network structure and tunable parameters. Interpretability of learnt knowledge is another severe problem in conventional neural networks.

On the other hand, function approximation by fuzzy system employs the concept of dividing the input space into sub regions, and for each sub region a fuzzy rule is defined thus making the system interpretable. The performance of the fuzzy system depends on the generation of sub regions in input space for a specific problem. The practical limitation arises with fuzzy systems when the input variables are increased and the number of fuzzy rules explodes leading to the problem known as the curse of dimensionality.

Both fuzzy system and neural network are universal function approximators and can approximate functions to any arbitrary degree of accuracy [20], [22]. Fuzzy neural system also has capability of approximating any continuous function or modeling a system [23],[24],[25]. The proposed fuzzy neural network was tested to exploit the advantages of both neural network and fuzzy system seamlessly in the applications like function approximation and classification.

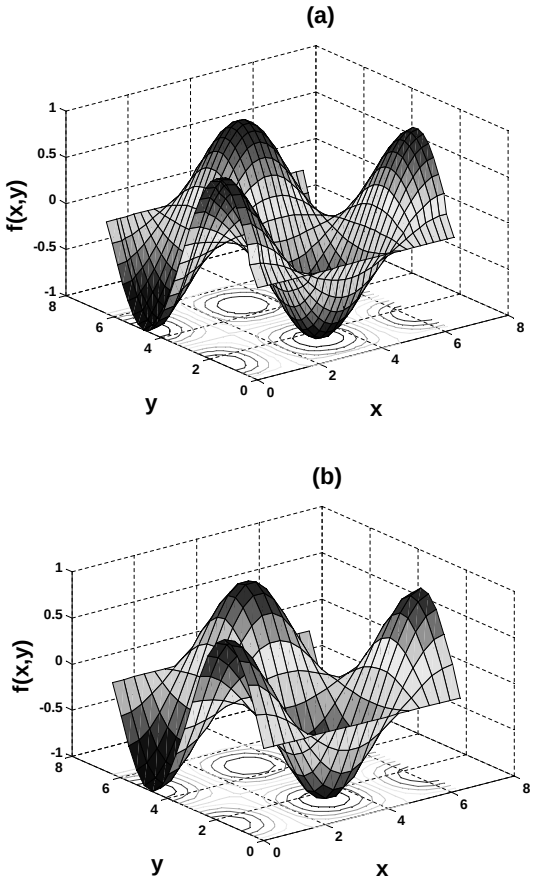

Figure 6: (a) Mesh plot and contours of 900 training patterns. (b) Mesh plot and contours of 400 testing patterns.

Table 1: Details of different learning schedules used for simulation studies

Learning Schedule	Details
LS=0.2	η and α are fixed to 0.2
LS=0.1	η and α are fixed to 0.1
LS=0.01	η and α are fixed to 0.01
LS=0.001	η and α are fixed to 0.001

η-learning rate and α-momentum

4.1 Sine-Cosine Function

The learning capabilities of the proposed model was demonstrated by approximating the *sine-cosine* function given as

$$f(x, y) = sin(x)cos(y). \quad (26)$$

for the purpose of training the network the above function was described by 900 sample points, evenly distributed in a 30x30 grid in the input cross-space $[0, 2\pi]$ x$[0, 2\pi]$. The model was tested by another set of 400 points evenly distributed in a 20x20 grid in the input cross-space $[0, 2\pi]$x$[0, 2\pi]$. The mesh plot of training and testing patterns are shown in Fig. 6.

For training of the model, the centers of fuzzy weights between the input layer and rule layer were initially randomized in the range$[0, 2\pi]$ while the centers of fuzzy weights

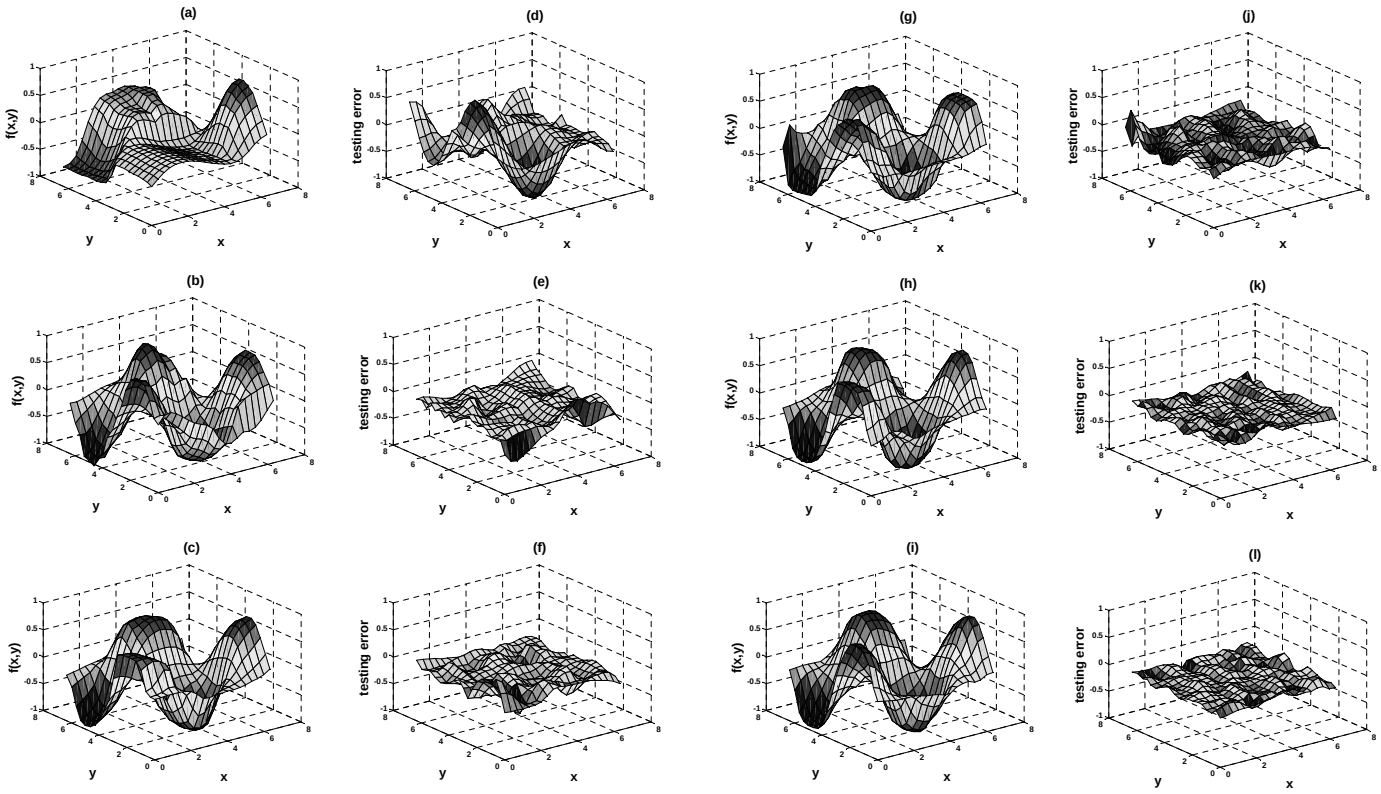

Figure 7: $f(x, y)$ surace plot and their corresponding testing error surface after 250 epochs for different rule counts with learning schedule LS=0.01, (a) $f(x, y)$ surface for 5 rules, (b) $f(x, y)$ surface for 10 rules, (c) $f(x, y)$ surface for 15 rules, (d) error suface for 5 rules, (e) error suface for 10 rules, (f) error suface for 15 rules, (g) $f(x, y)$ surface for 20 rules, (h) $f(x, y)$ surface for 30 rules, (i) $f(x, y)$ surface for 50 rules, (j) error suface for 20 rules, (k) error suface for 30 rules, (l) error suface for 50 rules.

between rule layer and output layer were initially randomized in the range $[-1, 1]$. The spreads of all the fuzzy weights and the spreads of input feature fuzzifiers were initialized randomly in range $[0.2, 0.9]$.

The number of free parameters that subsethood based FNN employs is straightforward to calculate: one spread for each numeric input; a center and a spread for each antecedent and consequent connection of a rule. For this function model employs a 2-r-1 network architecture, where r is the number of rule nodes. Therefore, since each rule has two antecedents and one consequent, an r-rule FNN system will have $6r+2$ free parameters.

Model was trained for different number of rules−5, 10, 15, 20, 30 and 50. To study the effect of learning parameters on the performance of model the simulation were performed with different learning schedules as shown in Table 1 . The root mean square error, evaluated for both training and testing patterns, is given as

$$RMSE_{trn} = \sqrt{\frac{\sum_{training\,patterns}(desired - actual)^2}{number\,of\,training\,patterns}} \tag{27}$$

$$RMSE_{test} = \sqrt{\frac{\sum_{testing\,patterns}(desired - actual)^2}{number\,of\,testing\,patterns}} \tag{28}$$

Table 2: Root mean square errors for different rule count and learning schedules (LS) for 250 epochs

Rules	$LS = 0.2$		$LS = 0.1$	
	$RMSE_{trn}$	$RMSE_{test}$	$RMSE_{trn}$	$RMSE_{test}$
5	0.4306	0.5928	0.3464	0.6210
10	0.1851	0.3144	0.2745	0.3239
15	0.0897	0.1125	0.1250	0.1746
20	0.0631	0.1518	0.0811	0.1026
30	0.0418	0.0522	0.0518	0.0615
50	0.0316	0.0323	0.0219	0.0452
Rules	$LS = 0.01$		$LS = 0.001$	
	$RMSE_{trn}$	$RMSE_{test}$	$RMSE_{trn}$	$RMSE_{test}$
5	0.3352	0.4080	0.3428	0.3567
10	0.1758	0.1997	0.2194	0.2783
15	0.1419	0.1516	0.2771	0.2954
20	0.0972	0.1247	0.1446	0.1432
30	0.0645	0.0735	0.1135	0.1246
50	0.0336	0.0354	0.0336	0.0354

In order to visualize the surface obtained from the test set after training the function f(x,y)=sin(x) cos(y) for 250 epochs the three dimensional plots of the function were generated. Fig. 7 illustrates surface plots of the function and the error surface for different values of rule counts with learning schedule as LS=0.01. It is observed that a model of mere 5 rules seems to be coarsely approximating

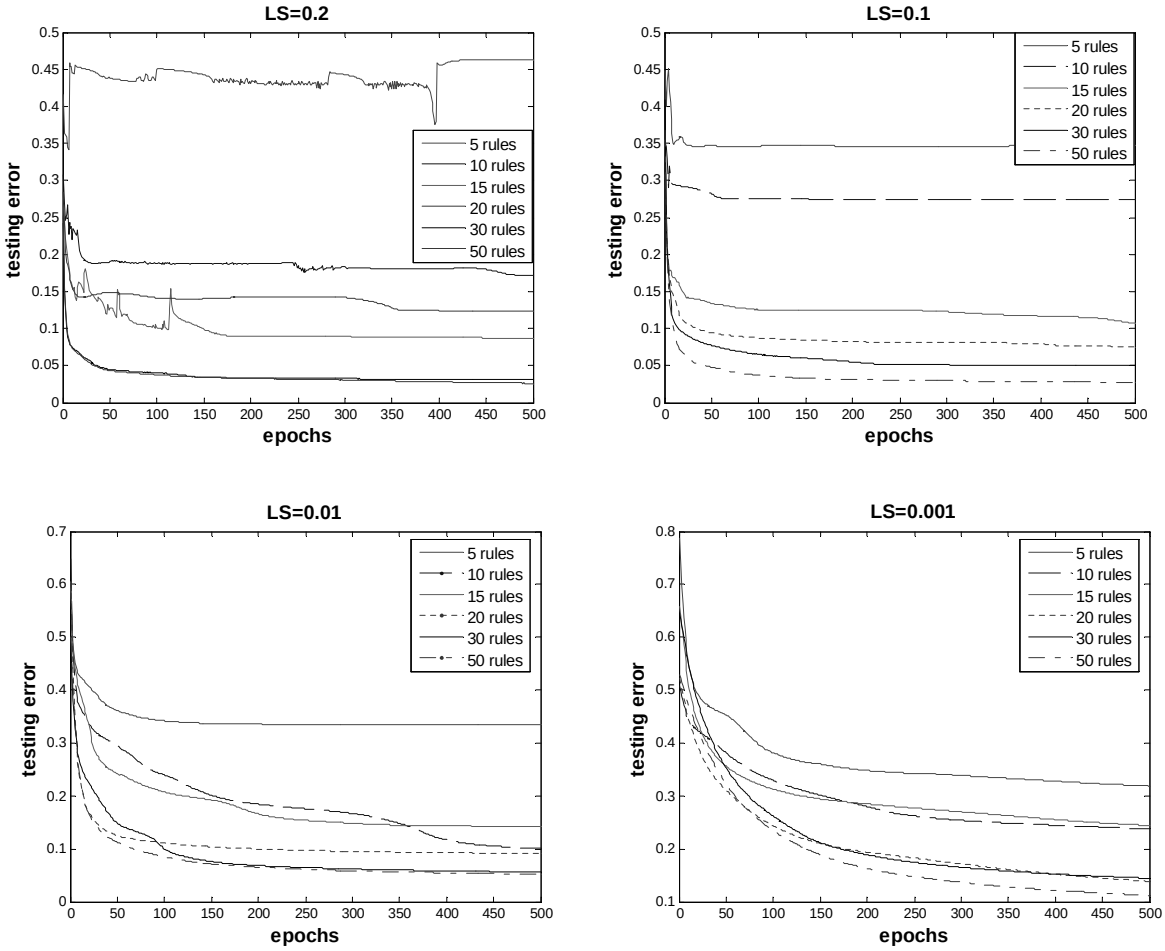

Figure 8: Error trajectories for different rules and learning schedules in $sine - cosine$ function problem.

the given function. The error is more where the slope of the function changes in that region. Thus, increasing the number of rules generates better approximated surface for $f(x, y)$ and the corresponding plots are shown in Fig. 7.

From the results shown in Table 2 it is observed that for a learning schedule LS=0.2 or higher and with small rule count the subsethood model is unable to train, resulting oscillations in error trajectories as shown in Fig. 8. This may occur due to the improper selection of learning parameters (learning rate (η) and momentum (α)) and number of rules. But with the same learning parameters and higher rule counts like 30 and 50 rules model produces good approximation. The observations for fuzzy neuro model drawn in the above experiments can be summarized as following:

1. As the number of rules increases the approximation performance of model improves to a certain limit.
2. For higher learning rates and momentum with lower rule counts the model is unable to learn. In contrast if the learning rate and momentum are kept to small values a smooth decaying trajectory is obtained even for small rule counts.
3. Model works fairly well by keeping the learning rate and momentum fixed to small values.
4. Most of the learning is achieved in a small number of epochs.

4.2 Narazaki and Ralescu function

The function is expressed as follows,

$$y(x) = 0.2 + 0.8(x + 0.7sin(2\pi x)), 0 \leq x \leq 1 \qquad (29)$$

and the plot of the function is shown in Fig.9. The system architecture used for approximating single input-output function is 1-r-1, where r is the number of rule nodes. The tunable parameters that model employs for this application can be calculated same as the $sine - cosine$ function i.e. one spread for each input, and a center and a spread for each antecedent and consequent connection of rule. As each rule has one antecedent and one consequent, r rule architecture will have $4r+1$ free parameters. The model was trained using 21 training patterns. These patterns were generated at intervals of 0.05 in range [0,1]. Thus, the training patterns are of the form:

$$(0, y(0)), (0.05, y(0.05)), ..., (1, y(1)) \qquad (30)$$

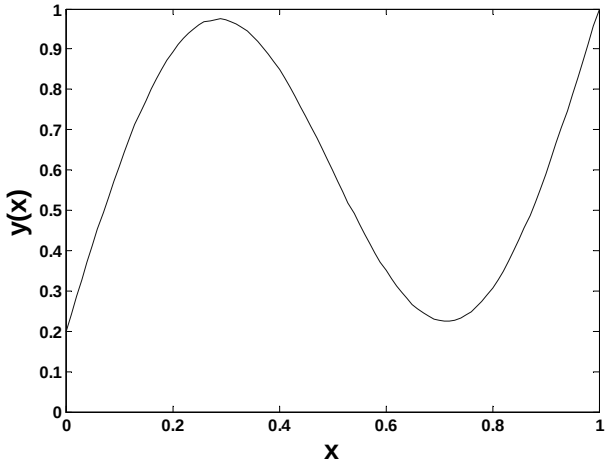

Figure 9: Narazaki-Ralescu function.

The evaluation was done using 101 test data taken at intervals of 0.01. The training and test sets generated were mutually exclusive. Two performance indices J1 and J2 as defined in [26], used for evaluation are given as:

$$J1 = 100\times\frac{1}{21}\sum_{training\,data}\frac{|actual\,output - desired\,output|}{desired\,output} \tag{31}$$

$$J2 = 100\times\frac{1}{101}\sum_{test\,data}\frac{|actual\,output - desired\,output|}{desired\,output} \tag{32}$$

Experiments were conducted for different rule counts, using a learning rate of 0.01 and momentum of 0.01 throughout the learning procedure. Table 3 summarizes the performance of model in terms of indices J1 and J2 for rule counts 3 to 6. It is evident from the performance measures that for 5 or 6 rules the approximation accuracy is much better than that for 3 or 4 rules. In general up to a certain limit, as the number of rules grows, the performance of model improves.
Table 4 compares the test accuracy performance index $J2$ for different models along with the number of rules and tunable parameters used to achieve it. With five rules the proposed model obtained $J1 = 0.9467$ and $J2 = 0.7403$ as better than other schemes. From the above results, it can be infer that subsethood-based FNN shows the ability to approximate function with good accuracy in comparison with other existing models.

Table 3: Subsethood based FNN performance for Narazaki-Ralescu's function

Number of Rules	Trainable Parameter	Training Accuracy (J1%)	Testing Accuracy (J2%)
3	13	2.57	1.7015
4	17	1.022	0.7350
5	21	0.94675	0.7403
6	25	0.6703	0.6595

Table 4: Performance comparison of subsethood based FNN with other methods for Narazaki-Ralescu's function

Methods and reference	Number of Rules	Trainable Parameters	Testing Accuracy (J2%)
FuGeNeSys[31]	5	15	0.856
Lin and Cunningham III [32]	4	16	0.987
Narazaki and Ralescu[26]	na	12	3.19
Subsethood based FNN	3	13	1.7015
Subsethood based FNN	5	21	0.7403

Table 5: Iris data classification results for the subsethood based Fuzzy Neural Network system

Rule Count	Free Parameters	RMSE	number of mis-classifications	resubstitution accuracy (%)
3	46	0.12183	1	99.33
4	60	0.12016	0	100
5	74	0.11453	0	100
6	88	0.11449	0	100
7	102	0.11232	0	100
8	116	0.10927	0	100

5 Classification

In classification problems, the purpose of the proposed network is to assign each pattern to one of a number of classes (or, more generally, to estimate the probability of membership of the case in each class). The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by Sir Ronald Aylmer Fisher as an example of discriminant analysis.

5.1 Iris data Classification

Iris data involves classification of three subspecies of the flower namely, *Iris sestosa*, *iris versicolor* and *Iris virginica* on the basis of four feature measurements of the Iris flower-sepal length, sepal width, petal length and petal width [27]. There are 50 patterns (of four features) for each of the three subspecies of Iris flower. The input pattern set thus comprises 150 four-dimensional patterns. This data can be obtained from UCI repository of machine learning databases through the following link− $http://www.ics.uci.edu/\ mlearn/MLRepository.html$. The six possible scatter plots of Iris data are shown in Fig. 10. It can be observed that classes *Iris versicolor* and *Iris virginica* substantially overlap, while class *Iris sestosa* is well separated from the other two. For this classification problem subsethood based FNN model employs a 4-r-3 network architecture: the input layer consists of four numeric nodes; the output layer comprises three class nodes; and there are r rule nodes in the hidden layer.
To train the network initially the centers of antecedent weight fuzzy sets were randomized in the range of the minimum and maximum values of respective input features of Iris data. Feature-wise, these ranges are (4.3, 7.9), (2.0, 4.4), (1.0, 6.9) and (0.1, 2.5). The centers of hidden-output weight fuzzy sets were randomized in the

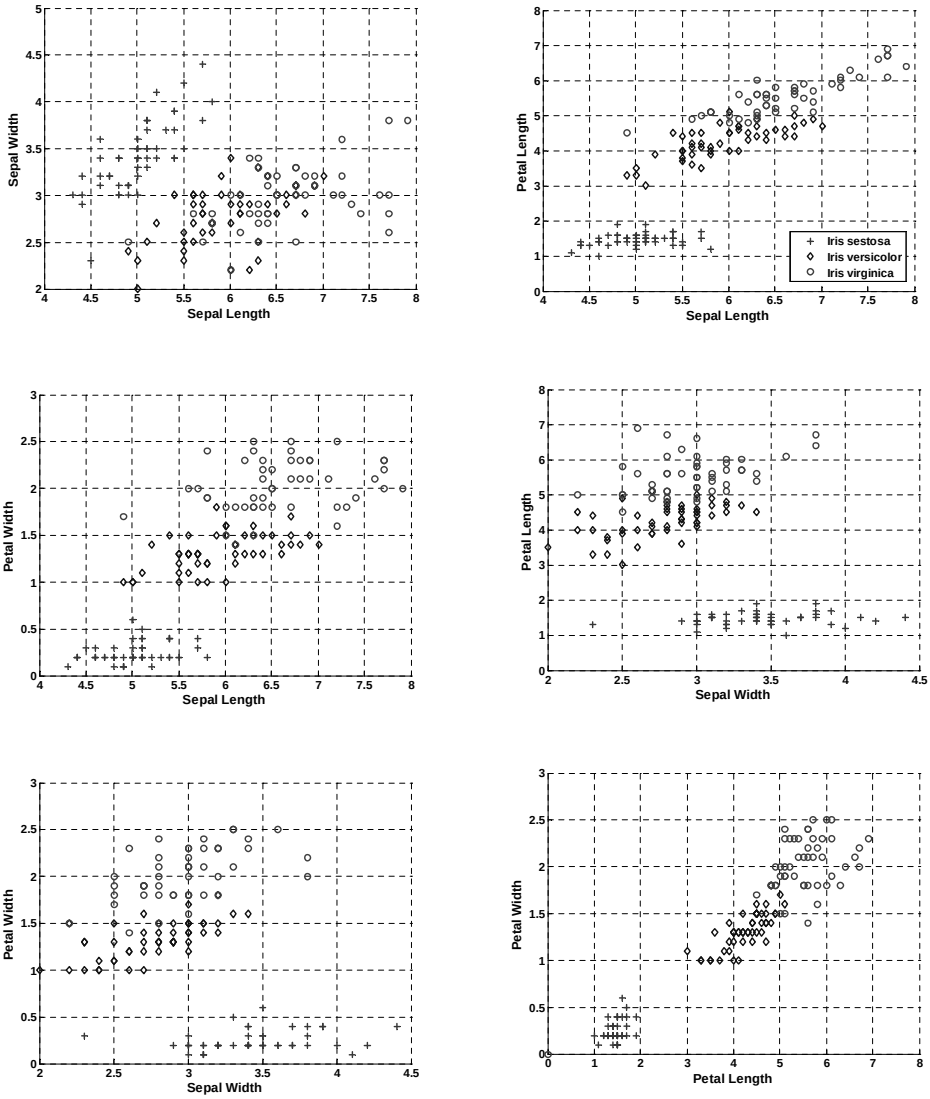

Figure 10: Six projection plots of Iris data.

range (0,1) and the spreads of all fuzzy weights and feature spreads were randomized in the range (0.2, 0.9). All 150 patterns of the Iris data were presented sequentially to the input layer of the network for training. The learning rate and momentum were kept constant at 0.001 during the training process. The test patterns which again comprised all 150 patterns of Iris data were presented to the trained network and the resubstitution error computed.

Simulation experiments were conducted with different numbers of rule nodes to illustrate the performance of the classifier with a variation in the number of rules. Notice that for r rules, the number of connections in the 4-r-3 architecture for Iris data will be $7r$. Because the representation of a fuzzy weight requires two parameters (center and spread), the total number of free parameters to be trained will be $14r+4$.

Table 5 summarizes the performance of proposed model for different rule counts. It is observed that except for rule count 3 subsethood based FNIS model is able to achieve 100 % resubstitution accuracy by classifying all patterns correctly. Thus by merely using 60 parameters for subsethood based FNN model produces no mis-classifications, and using only 46 free parameters 1 mis-classification is obtained. Apart from this, it is also observed from Table 5 that as the numbers of rules increase the training root mean square error (RMSE) decreases.

As an example, the fuzzy weights of the trained network with four rules that produce zero resubstitution error are illustrated in the scatter plot of Iris data in Fig. 11. The rule patches in two dimensions were obtained by finding the rectangular overlapping area produced by the projection of 3σ points of the Gaussian fuzzy sets on different input feature axes of the same rule. The 3σ points were chosen because for 3σ on either side of centers of a Gaussian fuzzy set 99.7 % of the total area of the fuzzy set gets covered.

To solve the Iris data classification problem using other

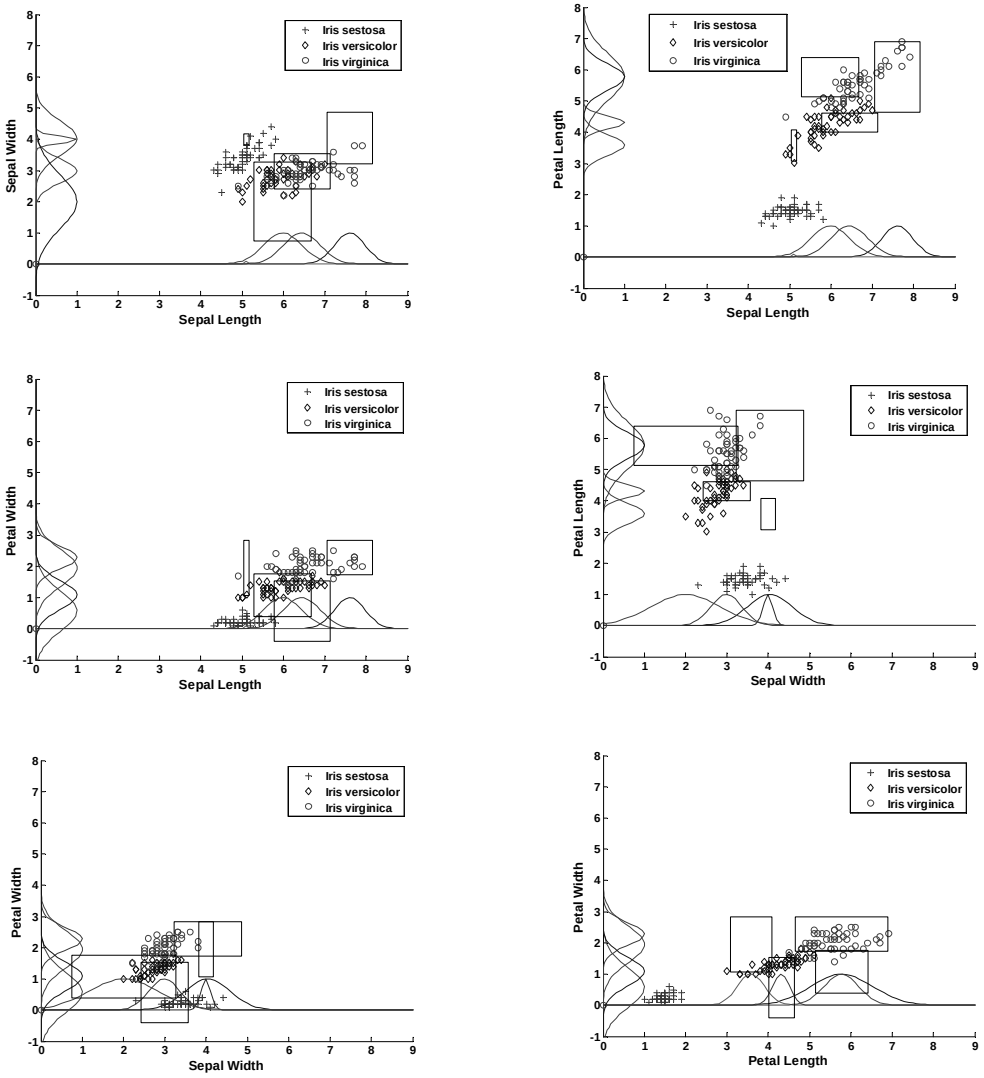

Figure 11: Rule patches learnt by a 4-rule subsethood based FNN network.

techniques like genetic algorithm (GA), learning vector quantization (LVQ) and its family of generalized fuzzy algorithms (GLVQ-F), and random search (RS), several attempts have been reported in the literature [29, 30, 12, 31].

The results obtained from GA, RS, LVQ and GLVQ-F have been adapted from [28] for the purpose of comparison and summarized in Table 6.

Table 6 compares the resubstitution mis-classifications of subsethood based FNN with these techniques. In GA and random search techniques 2 resubstitution misclassification for 3 rules are reported. For 4 rules the GA performance deteriorates with 4 misclassifications in comparison to 2 misclassifications in random search. In comparison, subsethood based FNN has only one resubstitution mis-classification for 3 rules which is less than other methods. Subsethood based FNN produces zero resubstitution mis-classification for any number of rules greater than or equal to 4 as summarized in Table 6.

Table 6: Comparison of number of resubstitution misclassifications for Iris data with different number of prototypes/rules

Prototype/Rule → Model ↓	3	4	5	6	7	8
LVQ*	17	24	14	14	3	4
GLVQ-F*	16	20	19	14	5	3
GA*	2	4	2	2	3	1
RS*	2	2	2	2	1	1
SuPFuNIS [15]	1	1	0	0	0	0
Subsethood based FNN	1	0	0	0	0	0

$*$: result adapted from [28]

6 Conclusion

In this paper the subsethood based Fuzzy Neural Network model is designed and simulated in Matlab 7.1 environment. The three benchmark problems are also discussed to demonstrate the application potential of the proposed fuzzy neural network model. The experiment results show that the proposed model performs better as a universal approximator and a classifier when compared to other existing models. The subsethood based Fuzzy Neural Network model suffers few drawbacks like the use of heuristic approach to select the number of rule nodes to solve a particular problem and unable to accommodate the use of disjunctions of conjunctive antecedents. In future work authors shall investigate the genetic algorithm based evolvable Fuzzy Neural network to overcome the drawbacks of proposed model and also use subsethood based Fuzzy Neural Network model in the field of image compression and control.

Appendix

The expressions for cardinality can be evaluated in terms of the standard function erf(x) as follows

(A) Expressions for $C(s_i \cap w_{ij})$

The expression for cardinality can be evaluated in terms of the standard error function erf (x) given in (8). The case wise expressions for $C(s_i \cap w_{ij})$ for all four possibilities identified in Section (2.2) are as follows.

Case 1– $C_i = C_{ij}$: If $\sigma_i < \sigma_{ij}$, the signal fuzzy set s_i completely belongs to the weight fuzzy set w_{ij}, and the cardinality $C(s_i \cap w_{ij}) = C(s_i)$

$$\begin{aligned} C(s_i \cap w_{ij}) &= C(s_i) = \int_{-\infty}^{\infty} e^{-((x-c_i)/\sigma_i)^2} dx \\ &= \sigma_i \frac{\sqrt{\pi}}{2} \left[erf(\infty) - erf(-\infty) \right] \\ &= \sigma_i \sqrt{\pi}. \end{aligned} \tag{33}$$

Similarly, $C(s_i \cap w_{ij}) = C(w_{ij})$ if $\sigma_i > \sigma_{ij}$ and $C(s_i \cap w_{ij}) = \sigma_{ij}\sqrt{\pi}$. If $\sigma_i = \sigma_{ij}$, the two fuzzy sets are identical. Summarizing these three sub cases, the values of cardinality can be shown as

$$C(s_i \cap w_{ij}) = \begin{cases} C(s_i) = \sigma_i \sqrt{\pi}, & \text{if } \sigma_i < \sigma_{ij} \\ C(w_{ij}) = \sigma_{ij} \sqrt{\pi}, & \text{if } \sigma_i > \sigma_{ij} \\ C(s_i) = C(w_{ij}) = \sigma_i \sqrt{\pi} = \sigma_{ij} \sqrt{\pi}, & \text{if } \sigma_i = \sigma_{ij}. \end{cases} \tag{34}$$

Case 2– $C_i \neq C_{ij}$, $\sigma_i = \sigma_{ij}$: In this case there will be exactly one cross over point h_1. Assuming $c_{ij} > c_i$, the cardinality $C(s_i \cap w_{ij})$ can be evaluated as

$$\begin{aligned} C(s_i \cap w_{ij}) &= \int_{-\infty}^{h_1} e^{-((x-c_{ij})/\sigma_{ij})^2} dx \\ &\quad + \int_{h_1}^{\infty} e^{-((x-c_i)/\sigma_i)^2} dx \\ &= \sigma_i \frac{\sqrt{\pi}}{2} \left[1 + erf\left(\frac{(h_1 - c_{ij})}{\sigma_{ij}} \right) \right] \\ &\quad + \sigma_i \frac{\sqrt{\pi}}{2} \left[1 - erf\left(\frac{(h_1 - c_i)}{\sigma_i} \right) \right] \end{aligned} \tag{35}$$

If $c_{ij} < c_i$, the expression for cardinality $C(s_i \cap w_{ij})$ is

$$\begin{aligned} C(s_i \cap w_{ij}) &= \int_{-\infty}^{h_1} e^{-((x-c_i)/\sigma_i)^2} dx \\ &\quad + \int_{h_1}^{\infty} e^{-((x-c_{ij})/\sigma_{ij})^2} dx \\ &= \sigma_i \frac{\sqrt{\pi}}{2} \left[1 + erf\left(\frac{(h_1 - c_i)}{\sigma_i} \right) \right] \\ &\quad + \sigma_i \frac{\sqrt{\pi}}{2} \left[1 - erf\left(\frac{(h_1 - c_{ij})}{\sigma_{ij}} \right) \right] \end{aligned} \tag{36}$$

Case 3– $C_i \neq C_{ij}$, $\sigma_i < \sigma_{ij}$: In this case, there will be two crossover points h_1 and h_2, as calculated in (5) and (6). Assuming $h_1 < h_2$ and $c_{ij} > c_i$, the cardinality $C(s_i \cap w_{ij})$ can be evaluated as

$$\begin{aligned} C(s_i \cap w_{ij}) &= \int_{-\infty}^{h_1} e^{-((x-c_i)/\sigma_i)^2} dx \\ &\quad + \int_{h_1}^{h_2} e^{-((x-c_{ij})/\sigma_{ij})^2} dx \\ &\quad + \int_{h_2}^{\infty} e^{-((x-c_i)/\sigma_i)^2} dx \\ &= \sigma_i \frac{\sqrt{\pi}}{2} \left[1 + erf\left(\frac{(h_1 - c_i)}{\sigma_i} \right) \right] \\ &\quad + \sigma_i \frac{\sqrt{\pi}}{2} \left[1 - erf\left(\frac{(h_1 - c_i)}{\sigma_i} \right) \right] \\ &\quad + \sigma_{ij} \frac{\sqrt{\pi}}{2} \left[erf\left(\frac{(h_2 - c_{ij})}{\sigma_{ij}} \right) \right. \\ &\quad \left. - erf\left(\frac{(h_1 - c_{ij})}{\sigma_{ij}} \right) \right]. \end{aligned} \tag{37}$$

if $c_{ij} < c_i$, the expression for $C(s_i \cap w_{ij})$ is identical to (37)

Case 4– $C_i \neq C_{ij}$, $\sigma_i > \sigma_{ij}$: This case is similar to case 3, and once again, there will be two crossover points h_1 and h_2, as calculated in (5) and (6). Assuming $h_1 < h_2$ and $c_{ij} > c_i$, the cardinality $C(s_i \cap w_{ij})$ can be evaluated as

$$\begin{aligned} C(s_i \cap w_{ij}) &= \int_{-\infty}^{h_1} e^{-((x-c_{ij})/\sigma_{ij})^2} dx \\ &\quad + \int_{h_1}^{h_2} e^{-((x-c_i)/\sigma_i)^2} dx \end{aligned}$$

$$
\begin{aligned}
&+\int_{h_2}^{\infty} e^{-((x-c_{ij})/\sigma_i)^2}dx \\
=\ & \sigma_{ij}\frac{\sqrt{\pi}}{2}\left[1+erf\left(\frac{(h_1-c_{ij})}{\sigma_{ij}}\right)\right] \\
&+\sigma_{ij}\frac{\sqrt{\pi}}{2}\left[1-erf\left(\frac{(h_1-c_{ij})}{\sigma_{ij}}\right)\right] \\
&+\sigma_i\frac{\sqrt{\pi}}{2}\left[erf\left(\frac{(h_2-c_i)}{\sigma_i}\right)\right. \\
&\left.-erf\left(\frac{(h_1-c_i)}{\sigma_i}\right)\right]. \qquad (38)
\end{aligned}
$$

If $c_{ij} < c_i$ the expression for cardinality is identical to (38).
Corresponding expressions for $\mathcal{E}(s_i \cap w_{ij})$ are obtained by substituting for $C(s_i \cap w_{ij})$ from (34)-(38) in to (7).

(B) Expressions for $\partial C(s_i \cap w_{ij})/\partial c_{ij}$, $\partial C(s_i \cap w_{ij})/\partial \sigma_{ij}$ and $\partial C(s_i \cap w_{ij})/\partial \sigma_i$

As per the discussion in the Section (3.2) that, the calculation of $\partial C(s_i \cap w_{ij})/\partial c_{ij}$, $\partial C(s_i \cap w_{ij})/\partial \sigma_{ij}$ and $\partial C(s_i \cap w_{ij})/\partial \sigma_i$ is required in (23), (24) and (25) which depends on the nature of overlap. Therefore the case wise expressions are given as following:

Case 1– $C_i = C_{ij}$: As is evident from (34), $C(s_i \cap w_{ij})$ is independent of c_{ij}, and therefore,

$$\frac{\partial C(s_i \cap w_{ij})}{\partial c_{ij}} = 0. \qquad (39)$$

similarly the first derivative of (34) with respect to σ_{ij} and σ_i is,

$$\frac{\partial C(s_i \cap w_{ij})}{\partial \sigma_{ij}} = \begin{cases} \sqrt{\pi}, & \text{if } c_{ij} = c_i \text{ and } \sigma_{ij} \le \sigma_i \\ 0, & \text{if } c_{ij} = c_i \text{ and } \sigma_{ij} > \sigma_i. \end{cases} \qquad (40)$$

$$\frac{\partial C(s_i \cap w_{ij})}{\partial \sigma_i} = \begin{cases} 0, & \text{if } c_{ij} = c_i \text{ and } \sigma_{ij} < \sigma_i \\ \sqrt{\pi}, & \text{if } c_{ij} = c_i \text{ and } \sigma_{ij} \ge \sigma_i. \end{cases} \qquad (41)$$

Case 2– $C_i \neq C_{ij}$, $\sigma_i = \sigma_{ij}$: When $c_{ij} > c_i$, $\partial C(s_i \cap w_{ij})/\partial c_{ij}$, $\partial C(s_i \cap w_{ij})/\partial \sigma_{ij}$ and $\partial C(s_i \cap w_{ij})/\partial \sigma_i$ are derived by differentiating (35) as follows :

$$
\begin{aligned}
\frac{\partial C(s_i \cap w_{ij})}{\partial c_{ij}} &= \int_{-\infty}^{h_1} \frac{\partial}{\partial c_{ij}} e^{-((x-c_{ij})/\sigma_{ij})^2}dx \\
&\quad + \int_{h_1}^{\infty} \frac{\partial}{\partial c_{ij}} e^{-((x-c_i)/\sigma_i)^2}dx \\
&= \int_{-\infty}^{h_1} \frac{\partial}{\partial c_{ij}} e^{-((x-c_{ij})/\sigma_{ij})^2}dx \\
&= -e^{-((h_1-c_{ij})/\sigma_{ij})^2} \qquad (42)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial C(s_i \cap w_{ij})}{\partial \sigma_{ij}} &= \int_{-\infty}^{h_1} \frac{\partial}{\partial \sigma_{ij}} e^{-((x-c_{ij})/\sigma_{ij})^2}dx \\
&\quad + \int_{h_1}^{\infty} \frac{\partial}{\partial \sigma_{ij}} e^{-((x-c_i)/\sigma_i)^2}dx \\
&= -\frac{h_1-c_{ij}}{\sigma_{ij}} e^{-((h_1-c_{ij})/\sigma_{ij})^2} \\
&\quad + \frac{\sqrt{\pi}}{2}\left[erf\left(\frac{(h_1-c_{ij})}{\sigma_{ij}}\right)+1\right] \qquad (43)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial C(s_i \cap w_{ij})}{\partial \sigma_i} &= \int_{-\infty}^{h_1} \frac{\partial}{\partial \sigma_i} e^{-((x-c_{ij})/\sigma_{ij})^2}dx \\
&\quad + \int_{h_1}^{\infty} \frac{\partial}{\partial \sigma_i} e^{-((x-c_i)/\sigma_i)^2}dx \\
&= \frac{h_1-c_i}{\sigma_i} e^{-((h_1-c_i)/\sigma_i)^2} \\
&\quad - \frac{\sqrt{\pi}}{2}\left[erf\left(\frac{(h_1-c_i)}{\sigma_i}\right)-1\right] \qquad (44)
\end{aligned}
$$

When $c_{ij} < c_i$, $\partial C(s_i \cap w_{ij})/\partial c_{ij}$, $\partial C(s_i \cap w_{ij})/\partial \sigma_{ij}$ and $\partial C(s_i \cap w_{ij})/\partial \sigma_i$ are derived by differentiating (36) as follows :

$$
\begin{aligned}
\frac{\partial C(s_i \cap w_{ij})}{\partial c_{ij}} &= \int_{-\infty}^{h_1} \frac{\partial}{\partial c_{ij}} e^{-((x-c_i)/\sigma_i)^2}dx \\
&\quad + \int_{h_1}^{\infty} \frac{\partial}{\partial c_{ij}} e^{-((x-c_{ij})/\sigma_{ij})^2}dx \\
&= \int_{h_1}^{\infty} \frac{\partial}{\partial c_{ij}} e^{-((x-c_{ij})/\sigma_{ij})^2}dx \\
&= e^{-((h_1-c_{ij})/\sigma_{ij})^2} \qquad (45)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial C(s_i \cap w_{ij})}{\partial \sigma_{ij}} &= \int_{-\infty}^{h_1} \frac{\partial}{\partial \sigma_{ij}} e^{-((x-c_i)/\sigma_i)^2}dx \\
&\quad + \int_{h_1}^{\infty} \frac{\partial}{\partial \sigma_{ij}} e^{-((x-c_{ij})/\sigma_{ij})^2}dx \\
&= -\frac{h_1-c_{ij}}{\sigma_{ij}} e^{-((h_1-c_{ij})/\sigma_{ij})^2} \\
&\quad - \frac{\sqrt{\pi}}{2}\left[erf\left(\frac{(h_1-c_{ij})}{\sigma_{ij}}\right)-1\right] \qquad (46)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial C(s_i \cap w_{ij})}{\partial \sigma_i} &= \int_{-\infty}^{h_1} \frac{\partial}{\partial \sigma_i} e^{-((x-c_i)/\sigma_i)^2}dx \\
&\quad + \int_{h_1}^{\infty} \frac{\partial}{\partial \sigma_i} e^{-((x-c_{ij})/\sigma_{ij})^2}dx \\
&= -\frac{h_1-c_i}{\sigma_i} e^{-((h_1-c_i)/\sigma_i)^2} \\
&\quad + \frac{\sqrt{\pi}}{2}\left[erf\left(\frac{(h_1-c_i)}{\sigma_i}\right)+1\right] \qquad (47)
\end{aligned}
$$

Case 3– $C_i \neq C_{ij}$, $\sigma_i < \sigma_{ij}$: Once again, two sub cases arise similar to those of Case 2. When $c_{ij} > c_i$, $\partial C(s_i \cap w_{ij})/\partial c_{ij}$, $\partial C(s_i \cap w_{ij})/\partial \sigma_{ij}$ and $\partial C(s_i \cap w_{ij})/\partial \sigma_i$ are derived by differentiating (37).

$$\frac{\partial C(s_i \cap w_{ij})}{\partial c_{ij}} = \int_{-\infty}^{h_1} \frac{\partial}{\partial c_{ij}} e^{-((x-c_i)/\sigma_i)^2}dx$$

$$
\begin{aligned}
&+\int_{h_1}^{h_2}\frac{\partial}{\partial c_{ij}}e^{-((x-c_{ij})/\sigma_{ij})^2}dx\\
&+\int_{h_2}^{\infty}\frac{\partial}{\partial c_{ij}}e^{-((x-c_i)/\sigma_i)^2}dx\\
=&\int_{h_1}^{h_2}\frac{\partial}{\partial c_{ij}}e^{-((x-c_{ij})/\sigma_{ij})^2}dx\\
=&-e^{-((h_2-c_{ij})/\sigma_{ij})^2}\\
&+e^{-((h_1-c_{ij})/\sigma_{ij})^2} \qquad (48)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial C(s_i\cap w_{ij})}{\partial \sigma_{ij}} =&\int_{-\infty}^{h_1}\frac{\partial}{\partial \sigma_{ij}}e^{-((x-c_i)/\sigma_i)^2}dx\\
&+\int_{h_1}^{h_2}\frac{\partial}{\partial \sigma_{ij}}e^{-((x-c_{ij})/\sigma_{ij})^2}dx\\
&+\int_{h_2}^{\infty}\frac{\partial}{\partial \sigma_{ij}}e^{-((x-c_i)/\sigma_i)^2}dx\\
=&\int_{h_1}^{h_2}\frac{\partial}{\partial \sigma_{ij}}e^{-((x-c_{ij})/\sigma_{ij})^2}dx\\
=&\frac{h_1-c_{ij}}{\sigma_{ij}}e^{-((h_1-c_{ij})/\sigma_{ij})^2}\\
&-\frac{h_2-c_{ij}}{\sigma_{ij}}e^{-((h_2-c_{ij})/\sigma_{ij})^2}\\
&+\frac{\sqrt{\pi}}{2}\left[-erf\left(\frac{(h_1-c_{ij})}{\sigma_{ij}}\right)\right.\\
&\left.+\,erf\left(\frac{(h_2-c_{ij})}{\sigma_{ij}}\right)\right] \qquad (49)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial C(s_i\cap w_{ij})}{\partial \sigma_{i}} =&\int_{-\infty}^{h_1}\frac{\partial}{\partial \sigma_{i}}e^{-((x-c_i)/\sigma_i)^2}dx\\
&+\int_{h_1}^{h_2}\frac{\partial}{\partial \sigma_{i}}e^{-((x-c_{ij})/\sigma_{ij})^2}dx\\
&+\int_{h_2}^{\infty}\frac{\partial}{\partial \sigma_{i}}e^{-((x-c_i)/\sigma_i)^2}dx\\
=&\int_{-\infty}^{h_1}\frac{\partial}{\partial \sigma_{i}}e^{-((x-c_i)/\sigma_i)^2}dx\\
&+\int_{h_2}^{\infty}\frac{\partial}{\partial \sigma_{i}}e^{-((x-c_i)/\sigma_i)^2}dx\\
=&-\frac{h_1-c_i}{\sigma_i}e^{-((h_1-c_i)/\sigma_i)^2}\\
&+\frac{h_2-c_i}{\sigma_i}e^{-((h_2-c_i)/\sigma_i)^2}\\
&+\frac{\sqrt{\pi}}{2}\left[\left\{erf\left(\frac{(h_1-c_i)}{\sigma_i}\right)+1\right\}\right.\\
&\left.-\left\{erf\left(\frac{(h_2-c_{ij})}{\sigma_{ij}}\right)-1\right\}\right] \qquad (50)
\end{aligned}
$$

Similarly, if $c_{ij} < c_i$

$$
\begin{aligned}
\frac{\partial C(s_i\cap w_{ij})}{\partial c_{ij}} =&\int_{-\infty}^{h_1}\frac{\partial}{\partial c_{ij}}e^{-((x-c_i)/\sigma_i)^2}dx\\
&+\int_{h_1}^{h_2}\frac{\partial}{\partial c_{ij}}e^{-((x-c_{ij})/\sigma_{ij})^2}dx\\
&+\int_{h_2}^{\infty}\frac{\partial}{\partial c_{ij}}e^{-((x-c_i)/\sigma_i)^2}dx\\
=&\int_{h_1}^{h_2}\frac{\partial}{\partial c_{ij}}e^{-((x-c_{ij})/\sigma_{ij})^2}dx\\
=&-e^{-((h_2-c_{ij})/\sigma_{ij})^2}\\
&+e^{-((h_1-c_{ij})/\sigma_{ij})^2} \qquad (51)
\end{aligned}
$$

Thus for both the cases ($c_{ij} < c_i$ or $c_{ij} > c_i$), identical expressions for $\partial C(s_i\cap w_{ij})/\partial c_{ij}$ are obtained. Similarly, the expressions for $\partial C(s_i\cap w_{ij})/\partial \sigma_{ij}$ and $\partial C(s_i\cap w_{ij})/\partial \sigma_{i}$ also remain the same as (49) and (50) respectively in both the conditions.

Case 4– $C_i \neq C_{ij}$, $\sigma_i > \sigma_{ij}$: When $c_{ij} > c_i$, $\partial C(s_i\cap w_{ij})/\partial c_{ij}$, $\partial C(s_i\cap w_{ij})/\partial \sigma_{ij}$ and $\partial C(s_i\cap w_{ij})/\partial \sigma_{i}$ are derived by differentiating (38) as

$$
\begin{aligned}
\frac{\partial C(s_i\cap w_{ij})}{\partial c_{ij}} =&\int_{-\infty}^{h_1}\frac{\partial}{\partial c_{ij}}e^{-((x-c_{ij})/\sigma_{ij})^2}dx\\
&+\int_{h_1}^{h_2}\frac{\partial}{\partial c_{ij}}e^{-((x-c_i)/\sigma_i)^2}dx\\
=&-e^{-((h_1-c_{ij})/\sigma_{ij})^2}\\
&+e^{-((h_2-c_{ij})/\sigma_{ij})^2} \qquad (52)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial C(s_i\cap w_{ij})}{\partial \sigma_{ij}} =&\int_{-\infty}^{h_1}\frac{\partial}{\partial \sigma_{ij}}e^{-((x-c_{ij})/\sigma_{ij})^2}dx\\
&+\int_{h_1}^{h_2}\frac{\partial}{\partial \sigma_{ij}}e^{-((x-c_i)/\sigma_i)^2}dx\\
&+\int_{h_2}^{\infty}\frac{\partial}{\partial \sigma_{ij}}e^{-((x-c_{ij})/\sigma_{ij})^2}dx\\
=&\frac{h_1-c_{ij}}{\sigma_{ij}}e^{-((h_1-c_{ij})/\sigma_{ij})^2}\\
&-\frac{h_2-c_{ij}}{\sigma_{ij}}e^{-((h_1-c_{ij})/\sigma_{ij})^2}\\
&+\frac{\sqrt{\pi}}{2}\left[2+erf\left(\frac{(h_1-c_{ij})}{\sigma_{ij}}\right)\right.\\
&\left.-\,erf\left(\frac{(h_2-c_{ij})}{\sigma_{ij}}\right)\right] \qquad (53)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial C(s_i\cap w_{ij})}{\partial \sigma_{i}} =&\int_{-\infty}^{h_1}\frac{\partial}{\partial \sigma_{i}}e^{-((x-c_{ij})/\sigma_{ij})^2}dx\\
&+\int_{h_1}^{h_2}\frac{\partial}{\partial \sigma_{i}}e^{-((x-c_i)/\sigma_i)^2}dx\\
&+\int_{h_2}^{\infty}\frac{\partial}{\partial \sigma_{i}}e^{-((x-c_{ij})/\sigma_{ij})^2}dx\\
=&\frac{h_1-c_i}{\sigma_i}e^{-((h_1-c_i)/\sigma_i)^2}\\
&-\frac{h_2-c_i}{\sigma_i}e^{-((h_2-c_i)/\sigma_i)^2}
\end{aligned}
$$

$$+\frac{\sqrt{\pi}}{2}\left[\left\{erf\left(\frac{(h_2-c_i)}{\sigma_i}\right)\right\}\right.$$
$$\left.-\left\{erf\left(\frac{(h_1-c_i)}{\sigma_i}\right)\right\}\right] \quad (54)$$

If $c_{ij} < c_i$, the expressions for $\partial C(s_i \cap w_{ij})/\partial c_{ij}$, $\partial C(s_i \cap w_{ij})/\partial \sigma_{ij}$ and $\partial C(s_i \cap w_{ij})/\partial \sigma_i$ are again the same as (52), (53) and (54) respectively.

References

[1] Ke Meng, Zhao Yang Dong, Dian Hui Wang and Kit Po Wong, "A Self-Adaptive RBF Neural Network Classifier for Transformer Fault Analysis", *IEEE Transaction on Power Systems*, vol. 25, no. 3, pp. 1350-1360, 2010.

[2] H. H. Dam, H. A. Abbas, C. Lokan and Xin Yao, "Neural based Learning Classifier Systems", *IEEE Transaction on Knowledge and Data Engineering*, vol. 20, no. 1, pp. 26-39, 2008.

[3] G. Schaefer and T. Nakashima, "Data Mining of Gene Expression Data by Fuzzy and Hybrid Fuzzy Methods, *IEEE Transaction on Information Technology in Biomedicine*, vol. 14, no. 1, pp. 23-29, 2010.

[4] E. G. Mansoori, M. J. Zolghadri and S. D. Katebi, "Protine Superfamily Classification Using Fuzzy Rule Based Classifier", *IEEE Transaction on NanoBioscience*, vol. 8, no. 1, pp. 92-99, 2009.

[5] S. Horikawa, T. Furuhashi, and Y. Uchikawa, "On fuzzy modeling using fuzzy neural network with the back propagation algorithm", *IEEE trans. Neural Networks*, vol. 3, no. 5, pp. 801-806, 1992.

[6] J.M. Keller, R. R. Yager, and H. Tahani, "Neural network implementation of fuzzy logic", *Fuzzy Sets Syst.*, vol. 45, no. 5, pp. 1-12, 1992.

[7] C. T. Lin, and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision system", *IEEE Trans. Comput.*, vol. C-40, no. 12, pp. 1320-1336, December 1991.

[8] A.V. Nandedkar and P.K. Biswas, "A granular Reflex Fuzzy Min-Max Neural Network for Classification, *IEEE Transaction on Neural Networks*, vol. 20, no. 7, pp. 1117-1134, 2009.

[9] Detlef D. Nauck and N. Andreas, "The Evolution of Neuro-Fuzzy Systems, *In North American Fuzzy Information Processing Society (NAFIPS-05)*, pp. 98-103, 2005.

[10] J. S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system", *IEEE Trans. Syst.*, vol. 23, pp. 665-685, June 1993.

[11] S. Mitra, and S. K. Pal, "Fuzzy multilayer perceptron, inferencing and rule generation", *IEEE Trans. Neural Networks*, vol. 6, pp. 51-63, January 1995.

[12] D. Nauck, and R. Kruse, "A neuro-fuzzy method to learn fuzzy classification rules from data", *Fuzzy Sets Syst.*, vol. 89, pp. 277-288, 1997.

[13] Zhang Pei, Keyun Qin and Yang Xu, "Dynamic adaptive fuzzy neural-network identification and its application", *System, Man and Cybernetics, IEEE conference*, vol. 5, pp. 4974-4979, 2003.

[14] Chia-Feng Juang and Yu-wei Tsao, "A self-Evolving Integrated Type-2 Fuzzy Neural Network with online structure and parameter learning", *IEEE Transactions Fuzzy Systems*, vol. 16, no. 6, pp. 1411-1424, 2008.

[15] Sandeep Paul, and Satish Kumar, "Subsethood based Adaptive Linguistic Networks for Pattern Classification", *IEEE Transaction on System,man and cybernetics-part C:application and reviews*, vol. 33, No. 2, pp. 248-258, May 2003.

[16] P. K. Simpson, "Fuzzy min-max neural networks-part 1: Classification", *IEEE Transactions on Neural Networks*, vol. 3, pp. 776-786, 1992.

[17] P. K. Simpson, "Fuzzy min-max neural networks-part 2: Clustering", *IEEE Transaction on Fuzzy Systems*, vol. 1, pp. 32-45, February 1992.

[18] R. Kumar, R.R. Das, V. N. Mishra and R. Dwivedi, "A Neuro-Fuzzy Classifier-Cum-Quantifier for Analysis of Alcohols and Alcoholic Beverages using Responses of Thick film Oxide Gas Sensor Array", *IEEE Transaction on Sensors*, vol. 10, no. 9 pp. 1461-1468, 2010.

[19] P. Cermak, and P. Chmiel, "Parameter optimization of fuzzy-neural dynamic model", *IEEE conference on Fuzzy information, NAFIPS' 04*, vol. 2, pp. 762-767, 2004.

[20] B. Kosko, *Fuzzy Engineering*, Englewood Cliffs, NJ: Prentice-Hall, 1997.

[21] R. K. Brouwer, "Fuzzy rule extraction from a feed forward neural network by training a representative fuzzy neural network using gradient descent", *IEEE conference on Industrial Technology, IEEE ICIT' 04*, vol. 3, pp. 1168-1172, 2004.

[22] K. Hornik, "Approximation capabilities of multilayer feed forward networks are universal approximators", *IEEE Transaction on Neural Networks*, vol. 2, pp. 359-366, 1989.

[23] B. Kosko, "Fuzzy systems as universal approximators", *IEEE Transactions on computers*, vol. 43, no. 11, pp. 1329-1333, 1994.

[24] C. T. Lin, and Y. C. Lu, "A neural fuzzy system with linguistic teaching signals", *IEEE Transactions Fuzzy Systems*, vol. 3, no. 2, pp. 169-189, 1995.

[25] L. X. Wang, and J. M.Mendel, "Generating fuzzy rules from numerical data, with application", *Tech. Rep. 169, USC SIPI*, Univ. Southern California, Los Angeles, January 1991.

[26] H. Narazaki, and A. L. Ralescu, "An improved synthesis method for multilayered neural networks using qualitative knowledges", *IEEE Transactions Fuzzy Systems*, vol. 1, no. 2, pp. 125-137, 1993.

[27] R. A. Fisher, "The use of multiple measurements in taxonomic problems.", *Ann. Eugenics*, 7, 2, 179-188, 1986.

[28] L. I. Kuncheva, and J. C. Bezdek, "Nearest prototype classification: Clustering, genetic algorithms or random search?", *IEEE Trans. Syst., Man, Cybern.*, vol. C 28, pp.160-164, February 1998.

[29] S. Halgamuge and M.Glesner, "Neural networks in designing fuzzy systems for real world applications", *Fuzzy Sets Syst.*, vol. 65, pp. 1-12, 1994.

[30] N. Kasabov, and B. Woodford, "Rule insertion and rule extraction from evolving fuzzy neural networks: Algorithms and applications for building adaptive, intelligent expert systems", *In Proc. IEEE Intl. Conf. Fuzzy Syst. FUZZ-IEEE 99*, Seoul, Korea, vol. 3, pp. 1406-1411, August 1999.

[31] M. Russo, "FuGeNeSys-a fuzzy genetic neural system for fuzzy modeling", *IEEE Transactions Fuzzy Systems*, vol. 6, no. 3, pp. 373-388, 1993.

[32] Y. Lin, and G. A. Cunningham III, "A new approach to fuzzy-neural system modeling", *IEEE Transactions Fuzzy Systems*, vol. 3, no. 2, pp. 190-198, 1995.