J. Intelligent Learning Systems & Applications, 2010, 2: 97-109
doi:10.4236/jilsa.2010.22013 Published Online May 2010 (http://www.SciRP.org/journal/jilsa)

Scientific Research

Design of Hybrid Fuzzy Neural Network for Function Approximation

Amit Mishra[1]*, Zaheeruddin[1]

[1]Jamia Millia Islamia (A Central University), Department of Electrical Engineering, New Delhi, India; *Jaypee Institute of Engineering and Technology, Madhya Pradesh, India.
Email: amitutk@ gmail.com

Received November 21st, 2009; revised April 25th, 2010; accepted April 30th, 2010.

ABSTRACT

In this paper, a hybrid Fuzzy Neural Network (*FNN*) *system for function approximation is presented. 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 rule based knowledge is translated directly into network architecture. 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 method of activation spread in the network is based on a fuzzy mutual subsethood measure. Rule* (*hidden*) *node activations are computed as a fuzzy inner product. For a given numeric o fuzzy input, numeric outputs are computed using volume based defuzzification. A supervised learning procedure based on gradient descent is employed to train the network. The model has been tested on two different approximation problems: sine-cosine function approximation and Narazaki-Ralescu function and shows its natural capability of inference, function approximation, and classification.*

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.* differential 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.

In recent past, artificial neural network has also played an important role in solving many engineering problems. Neural network has advantages such as learning, adaption, fault tolerance, parallelism, and generalization. Fuzzy systems utilizing the learning capability of neural networks can successfully construct the input output mapping for many applications. The benefits of combining fuzzy logic and neural network have been explored extensively in the literature [1-3].

The term neuro-fuzzy system (also neuro-fuzzy methods or models) refers to combinations of techniques from neural networks and fuzzy system [4-8]. This never means 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 approximations or special case of it, like, classification or control [9-12] and the otherwise manual design process should be supported and replaced by an automatic learning process.

Here, 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 framework 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

Copyright © 2010 SciRes.

model. Sections 3 and 4 demonstrate the gradient descent learning procedure and the applications of the model to the task of function approximation respectively. Finally, the Section 5 concludes the paper.

2. Architectural and Operational Detail

To develop a fuzzy neural model, following issues are important to be discussed:

1) Signal transmission at input node.
2) Signal transmission method (similarity measures) from input nodes to rule nodes.
3) Method for activity aggregation at rule nodes.
4) Signal computation at output layer.
5) Learning technique and its mathematical formulation used in model.

The proposed architecture of subsethood based Fuzzy neural network is shown in **Figure 1**. Here x_1 to x_m and 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 antecedents. 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 and spread. 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 $w_{ij} = (c_{ij}, \sigma_{ij})$. As this model can handle simultaneous admission of numeric as well as fuzzy inputs, Numeric inputs are first fuzzified so that all outputs transmitted from the input layer of the network are fuzzy. Now, since the antecedent weights are also fuzzy, this requires the design of a method to transmit a fuzzy signal along a fuzzy weight. In this model signal transmission along the fuzzy weight is handled by calculating the mutual subsethood. A product aggregation operator computes the strength of firing at rule node. At output layer the signal computation is done with volume defuzzification to generate numeric outputs (y_1 to y_p). A gradient descent learning technique allows the model to fine tune rules with the help of numeric data.

2.1 Signal Transmission at Input Nodes

Since input features $x_1, \ldots, x_n$ can be either numeric and linguistic, there are two kinds of nodes in the input layer. Linguistic nodes accept a linguistic input represented by 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 **Figure 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 (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.

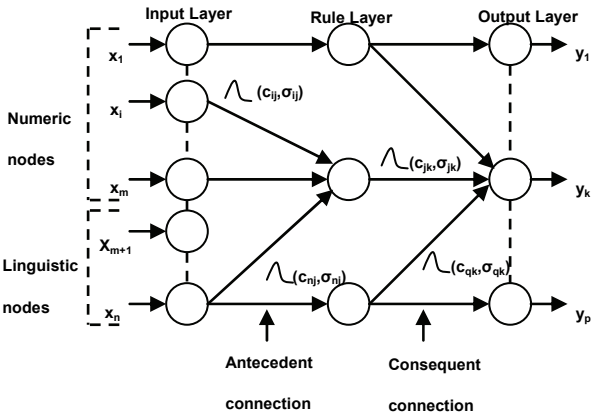


Figure 1. Architecture of subsethood based FNN model

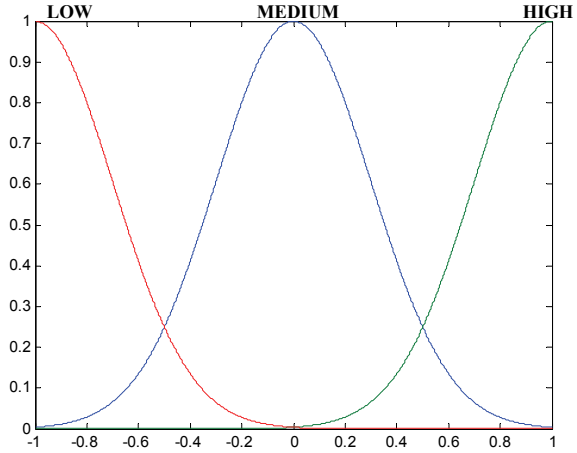


Figure 2. Fuzzy sets for fuzzy inputs

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 **Figure 3**, where a numeric feature value of 0.3 has been fuzzified 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 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

Copyright © 2010 SciRes.

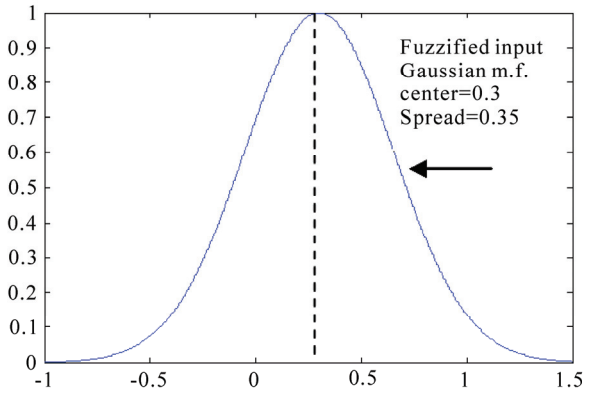


Figure 3. Fuzzification of numeric input

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* [13].

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:

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

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

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

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

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

$$\varepsilon(A,B) = \frac{C(A \cap B)}{C(A) + C(B) - C(A \cap B)} \tag{4}$$

Further detail on the mutual subsethood measure can be found in [13]. Depending upon the relative values of centers and spreads of fuzzy sets A and B, the following four different cases of nature of overlap arise:

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 **Figure 4**.

To calculate the crossover points, by setting $a(x) = b(x)$, the two cross over points h_1 and h_2 yield as,

$$h_1 = \frac{c_1 + \frac{\sigma_1}{\sigma_2} c_2}{1 + \frac{\sigma_1}{\sigma_2}} \tag{5}$$

$$h_2 = \frac{c_1 - \frac{\sigma_1}{\sigma_2} c_2}{1 - \frac{\sigma_1}{\sigma_2}} \tag{6}$$

These values of h_1 and h_2 are used to calculate the mutual subsethood $\varepsilon(A,B)$ 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

$$\varepsilon_{ij} = \varepsilon(s_i, w_{ij}) = \frac{C(s_i \cap w_{ij})}{C(s_i) + C(w_{ij}) - C(s_i \cap w_{ij})} \tag{7}$$

As shown in **Figure 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 ε_{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).

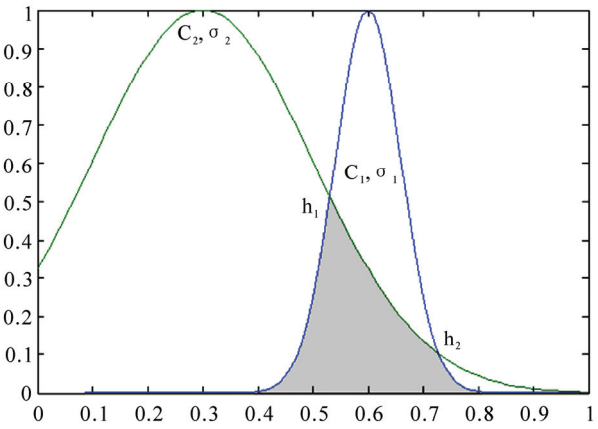


Figure 4. Example of overlapping: $c_1 > c_2$ and $\sigma_1 < \sigma_2$

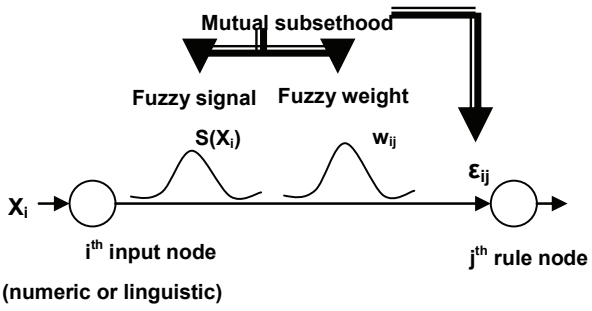


Figure 5. Fuzzy signal transmission

Copyright © 2010 SciRes.

The expression for cardinality can be evaluated for each of the four cases in terms of standard error function *erf*(*x*) represented as (8).

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

The case wise expressions for $C(s_i \cap w_{ij})$ are given in Appendix (1).

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

$$z_j = \prod_{i=1}^{n} \varepsilon_{ij} = \prod_{i=1}^{n} \varepsilon(S(x_i), w_{ij}) \tag{9}$$

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

$$S(z_j) = z_j \tag{10}$$

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 [13]. The activation of the output node is y_k, and V_{jk}'s denote consequent set volumes, then the general expression of defuzzification is

$$y_k = \frac{\sum_{j=1}^{q} z_j c_{jk} V_{jk}}{\sum_{j=1}^{q} z_j V_{jk}} \tag{11}$$

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 $S(y_k) = y_k$.

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. 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:

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

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

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

3.2 Partial Derivative 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

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

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,

Copyright © 2010 SciRes.

$$\frac{\partial e}{\partial c_{ij}} = \sum_{k=1}^{p} \frac{\partial e}{\partial y_k}\frac{\partial y_k}{\partial z_j}\frac{\partial z_j}{\partial \varepsilon_{ij}}\frac{\partial \varepsilon_{ij}}{\partial c_{ij}} = \sum_{k=1}^{p} -(d_k - y_k)\frac{\partial y_k}{\partial z_j}\frac{\partial z_j}{\partial \varepsilon_{ij}}\frac{\partial \varepsilon_{ij}}{\partial c_{ij}} \quad (18)$$

$$\frac{\partial e}{\partial \sigma_{ij}} = \sum_{k=1}^{p} \frac{\partial e}{\partial y_k}\frac{\partial y_k}{\partial z_j}\frac{\partial z_j}{\partial \varepsilon_{ij}}\frac{\partial \varepsilon_{ij}}{\partial \sigma_{ij}} = \sum_{k=1}^{p} -(d_k - y_k)\frac{\partial y_k}{\partial z_j}\frac{\partial z_j}{\partial \varepsilon_{ij}}\frac{\partial \varepsilon_{ij}}{\partial \sigma_{ij}} \quad (19)$$

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

$$\frac{\partial e}{\partial \sigma_j} = \sum_{j=1}^{q}\sum_{k=1}^{p} \frac{\partial e}{\partial y_k}\frac{\partial y_k}{\partial z_j}\frac{\partial z_j}{\partial \varepsilon_{ij}}\frac{\partial \varepsilon_{ij}}{\partial \sigma_i} = \sum_{j=1}^{q}\sum_{k=1}^{p} -(d_k - y_k)\frac{\partial y_k}{\partial z_j}\frac{\partial z_j}{\partial \varepsilon_{ij}}\frac{\partial \varepsilon_{ij}}{\partial \sigma_i} \quad (20)$$

where

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

and

$$\frac{\partial z_j}{\partial \varepsilon_{ij}} = \prod_{i=1,\, i\neq j}^{n} \varepsilon_{ij} \quad (22)$$

The expressions for antecedent connection, mutual subsethood partial derivatives $\frac{\partial \varepsilon_{ij}}{\partial c_{ij}}$ and $\frac{\partial \varepsilon_{ij}}{\partial \sigma_{ij}}$ are obtained by differentiating (7) with respect to c_{ij}, σ_{ij} and σ_i as in (23), (24) and (25).

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

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

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

In these equations, the calculation of $\partial C(s_i \cap w_{ij})/\partial c_{ij}$, $\partial C(s_i \cap w_{ij})/\partial \sigma_{ij}$ and $\partial C(s_i \cap w_{ij})/\partial \sigma_i$ are required which depends on the nature of overlap of the input feature fuzzy set and weight fuzzy set. The case wise expressions are demonstrated in Appendix (2).

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 with the drawback of their black box nature and heuristic decisions regarding network structure and tunable parameters. Interpretability of learnt knowledge in conventional neural networks is a severe problem. On the other hand, function approximation by fuzzy systems 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 how finally the sub regions are generated. The practical imitation 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. It is now well known that both fuzzy system and

Copyright © 2010 SciRes.

neural network are universal function approximators and can approximate functions to any arbitrary degree of accuracy [13,14]. Fuzzy neural system also has capability of approximating any continuous function or modeling a system [15-17].

There are two broad applications of function approximation-prediction and interpretation. In this paper, the work has been done on applications of function approximation related to prediction. In prediction, it is expected that, in future, new observations will be encountered for which only the input values are known, and the goal is to predict a likely output value for each case. The function estimate obtained from the training data through the learning algorithm is used for this purpose.

The subsethood based linguistic network proposed in the present paper has been tested on two different approximation problem: *sine-cosine* function approximation and *Narazaki-Ralescu* function [18].

4.1 Sine-Cosine Function

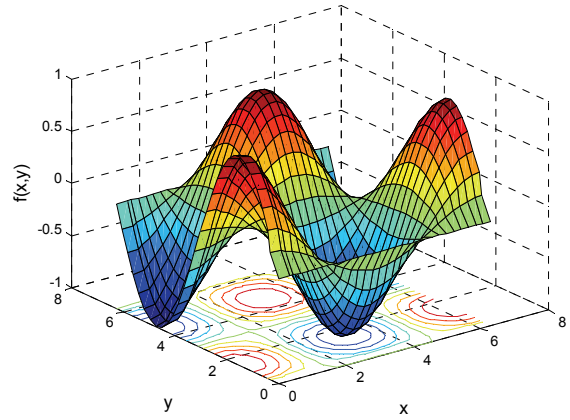
The learning capabilities of the proposed model can be demonstrated by approximating the *sine-cosine* function given by

$$f(x, y) = \sin(x)\cos(y) \tag{26}$$

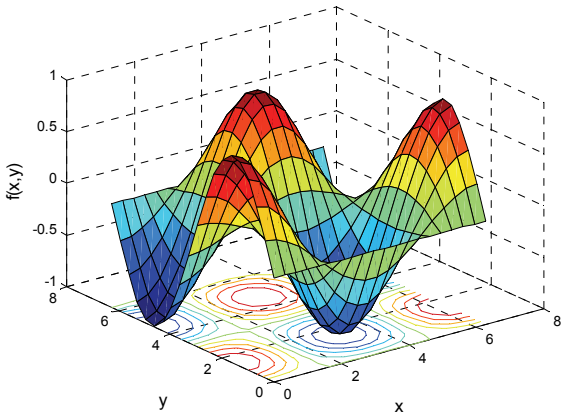
for the purpose of training the network the above function was described by 900 sample points, evenly distributed in a 30 × 30 grid in the input cross-space [0, 2π] × [0, 2π]. The model is tested by another set of 400 points evenly distributed in a 20 × 20 grid in the input cross-space [0, 2π] × [0, 2π]. The mesh plots of training and testing patterns are shown in **Figure 6**. For training of the model, the centers of fuzzy weights between the input layer and rule layer are initially randomized in the range [0, 2π] while the centers of fuzzy weights between rule layer and output layer are initially randomized in the range [–1, 1]. The spreads of all the fuzzy weights and the spreads of input feature fuzzifiers are 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 6*r* + 2 free parameters.

Model was trained for different number of rules— 5, 10, 15, 20, 30 and 50. Simulations were performed with different learning schedules given in **Table 1** to study the effect of learning parameters on the performance of model.



(a)



(b)

Figure 6. (a) Mesh plot and counters of 900 training patterns; (b) mesh plot and counters of 400 testing

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 0.2
LS = 0.01	η and α are fixed to 0.2
LS = 0.001	η and α are fixed to 0.2

(η -learning rate and α -momentum)

The root mean square error, evaluated for both training and testing patterns, is given as

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

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

Copyright © 2010 SciRes.

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 is generated. **Figure 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 can be observed that a model of mere 5 rules seems to be coarsely approximating 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 as can be observed as shown in **Figure 7**. As per the observation shown in **Table 2**, we can conclude that for learning schedule LS = 0.2 or higher and with small rule count the subsethood model is unable to train, resulting in oscillations in error trajectories shown as **Figure 8**. This may occur due to the improper selection of learning parameters (learning rate (η) and momentum (α)) and number of rules. But with 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 the 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) In general, Model works fairly well even for simple learning schemes 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,

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

and the plot of the function is shown in **Figure 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 is calculated to be as, one spread for one 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 is 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:

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

The evaluation was done using 101 test data taken at intervals of 0.01. The training and test sets generated are

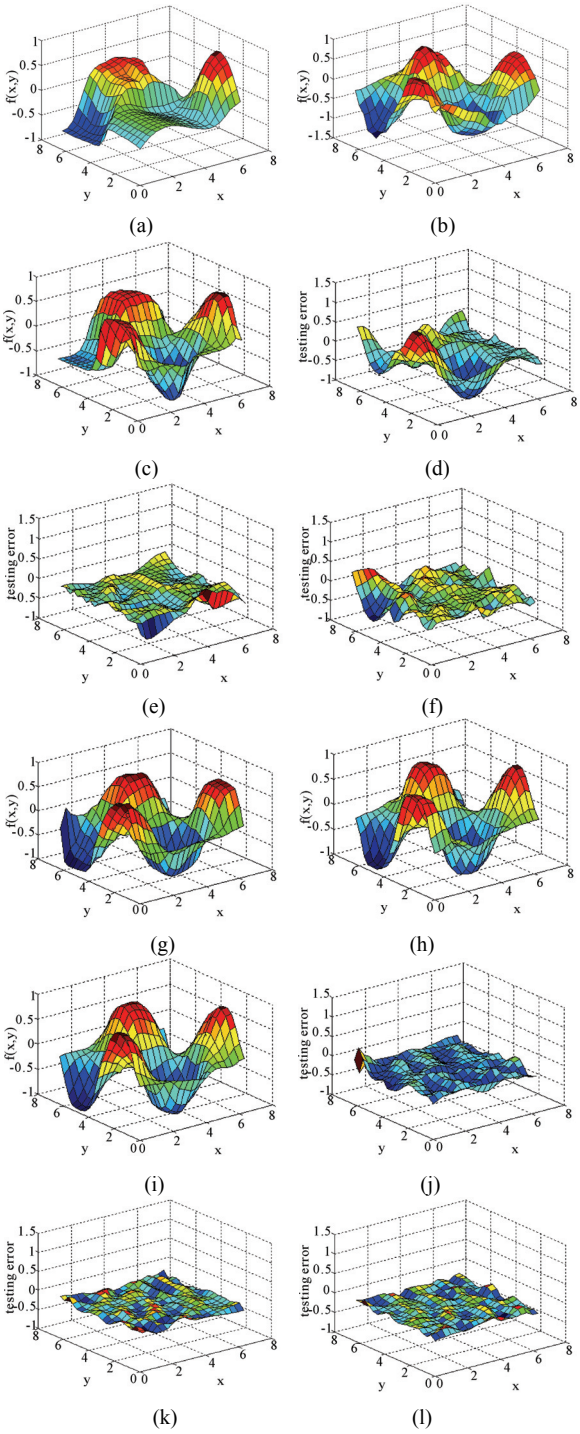


Figure 7. *f*(*x*, *y*) surface plot and their corresponding testing error surface after 250 epochs for different rule counts with learning schedule as LS = 0.01, (a) f(x, y) surface for 5 rules; (b) f(x, y) surface for 10 rules; (c) f(x, y) surface for 15 rules; (d) error surface for 5 rules; (e) error surface for 10 rules; (f) error surface 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 surface for 20 rules; (k) error surface for 30 rules; (l) error surface for 50 rules

Copyright © 2010 SciRes.

Table 2. Root mean square errors for different rule count and learning schedules 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

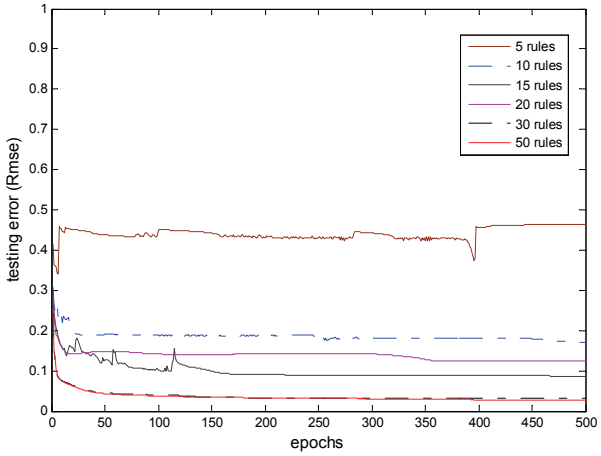
mutually exclusive. Two performance indices $J1$ and $J2$ as defined in [18], used for evaluation are given below:

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

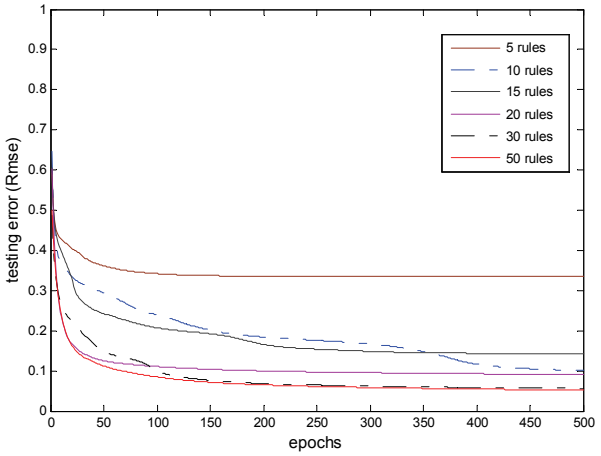
$$J2 = 100 \times \frac{1}{101} \sum_{test\ data} \frac{|actual\ output - desired\ output|}{desired\ output} \tag{32}$$

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 our model obtained $J1 = 0.9467$ and $J2 = 0.7403$ as better than all 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 models.



(a)



(b)

Figure 8. Error trajectories for different rules and learning schedule (a) LS = 0.2, (b) LS = 0.01

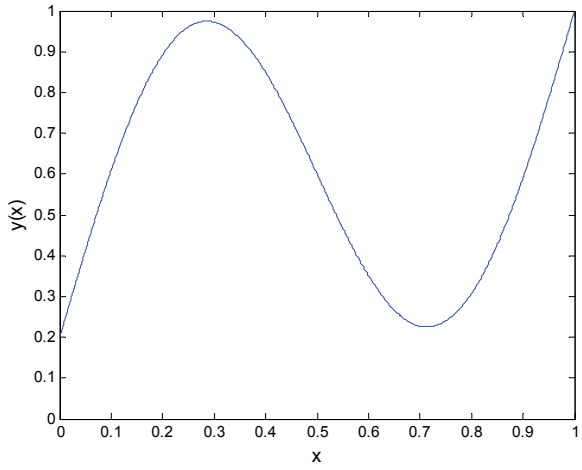


Figure 9. Narazaki-Ralescu function

Copyright © 2010 SciRes.

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	0.9468	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 [19]	5	15	0.856
Lin and Cunningham III [20]	4	16	0.987
Narazaki and Ralescu [18]	Na	12	3.19
Subsethood based FNN	3	13	1.7015
Subsethood based FNN	5	21	0.7403

5. Conclusions

In this paper the proposed subsethood based Fuzzy Neural Network model has now proved to be a universal function approximator. The model is tested on two different applications and found suitable for any function approximation problem empirically. The applications of function approximation are diverse and include the domain of physics, economics, control, planning, forecasting, machine learning and image compression. In future work authors shall incorporate concepts of fuzzy neural function approximator in image compression.

REFERENCES

[1] C. T. Lin and C. S. G. Lee, "Neural-Network-Based Fuzzy Logic Control and Decision System," *IEEE Transactions on Computers*, Vol. 40, No. 12, 1991, pp. 1320-1336.

[2] J. M. Keller, R. R. Yager and H. Tahani, "Neural Network Implementation of Fuzzy Logic," *Fuzzy Sets and Systems*, Vol. 45, No. 5, 1992, pp. 1-12.

[3] S. Horikawa, T. Furuhashi and Y. Uchikawa, "On Fuzzy Modeling Using Fuzzy Neural Networks with the Back Propagation Algorithm," *IEEE transactions on Neural Networks*, Vol. 3, No. 5, 1992, pp. 801-806.

[4] D. Nauck and R. Kruse, "A Neuro-Fuzzy Method to Learn Fuzzy Classification Rules from Data," *Fuzzy Sets and Systems*, Vol. 89, No. 3, 1997, pp. 277-288.

[5] J. S. R. Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference System," *IEEE Transactions on Systerms*, Vol. 23, 1993, pp. 665-685.

[6] S. Mitra and S. K. Pal, "Fuzzy Multilayer Perceptron, Inferencing and Rule Generation," *IEEE Transactions on Neural Networks*, Vol. 6, No. 1, 1995, pp. 51-63.

[7] W. L. Tung and C. Quek, "A Mamdani-Takagi-Sugeno Based Linguistic Neural-Fuzzy Inference System for Improved Interpretability-Accuracy Representation," *IEEE International Conference on Fuzzy Systems*, Jeju Island, August 2009, pp. 367-372.

[8] W. L. Tung and C. Quek, "eFSM-A Novel Online Neural-Fuzzy Semantic Memory model," *IEEE Transactions on Neural Networks*, Vol. 21, No. 1, 2010, pp. 136-157.

[9] P. K. Simpson, "Fuzzy Min-Max Neural Networks-Part 1: Classification," *IEEE Transactions on Neural Networks*, Vol. 3, No. 5, 1992, pp. 776-786.

[10] P. K. Simpson, "Fuzzy Min-Max Neural Networks-Part 2: Clustering," *IEEE Transaction on Fuzzy Systems*, Vol. 1, No. 1, 1992, pp. 32-45.

[11] R.-J. Wai and Z.-W. Yang, "Adaptive Fuzzy Neural Network Control Design via a T-S Fuzzy Model for a Robot Manipulator Including Actuator Dynamics," *IEEE Transactions on System*, *Man and Cybernetics-Part B*, Vol. 38, No. 5, 2008, pp. 1326-1346.

[12] P. Sandeep and S. 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, 2003, pp. 248-258.

[13] B. Kosko, "Fuzzy Engineering," Prentice-Hall, Englewood Cliffs, New Jersey, 1997.

[14] K. Hornik, "Approximation Capabilities of Multilayer Feed forward Networks are Universal Approximators," *IEEE Transaction on Neural Networks*, Vol. 2, No. 5, 1989, pp. 359-366.

[15] B. Kosko, "Fuzzy Systems as Universal Approximators," *IEEE Transactions on computers*, Vol. 43, No. 11, 1994, pp. 1329-1333.

[16] C. T. Lin and Y. C. Lu, "A Neural Fuzzy System with Linguistic Teaching Signals," *IEEE Transactions Fuzzy Systems*, Vol. 3, No. 2, 1995, pp. 169-189.

[17] L. X. Wang and J. M. Mendel, "Generating Fuzzy Rules from Numerical Data, with Application," Technical Report 169, USC SIPI, University of Southern California, Los Angeles, January 1991.

[18] H. Narazaki and A. L. Ralescu, "An Improved Synthesis Method for Multilayered Neural Networks Using Qualitative Knowledge's," *IEEE Transactions Fuzzy Systems*, Vol. 1, No. 2, 1993, pp. 125-137.

[19] M. Russo, "FuGeNeSys-a Fuzzy Genetic Neural System for Fuzzy Modeling," *IEEE Transactions Fuzzy Systems*, Vol. 6, No. 3, 1993, pp. 373-388.

[20] Y. Lin and G. A. Cunningham, "A New Approach to Fuzzy-Neural System Modeling," *IEEE Transactions Fuzzy Systems*, Vol. 3, No. 2, 1995, pp. 190-198.

Copyright © 2010 SciRes.

Appendix

1. 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 $C(s_i \cap w_{ij}) = C(s_i)$

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

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 (34).

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

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

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

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

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

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 cross over 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

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

Copyright © 2010 SciRes.

$$+\sigma_i \frac{\sqrt{\pi}}{2}\left[erf\left(\frac{(h_2 - c_i)}{\sigma_i}\right) - erf\left(\frac{(h_1 - c_i)}{\sigma_i}\right)\right] \quad (38)$$

if $c_{ij} < c_i$, the expression for $C(s_i \cap w_{ij})$ is identical to (38).

Corresponding expressions for $\varepsilon(s_i \cap w_{ij})$ are obtained by substituting the values of $C(s_i \cap w_{ij})$ from (34)-(38) to (7).

2. 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 from c_{ij}, and therefore,

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

Similarly the first derivative of (34) with respect to σ_{ij} and σ_i is shown as (40) and (41).

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

$$\frac{\partial C(s_i \cap w_{ij})}{\partial \sigma_i} = \begin{cases} 0, & if\ c_{ij} = c_i\ and\ \sigma_{ij} \le \sigma_i \\ \sqrt{\pi}, & if\ c_{ij} = c_i\ and\ \sigma_{ij} > \sigma_i \end{cases} \quad (41)$$

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

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

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

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

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

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

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

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

Copyright © 2010 SciRes.

$$= -\frac{h_1 - c_i}{\sigma_i} e^{-((h_1 - c_i)/\sigma_i)^2} + \frac{\sqrt{\pi}}{2}\left[erf\left(\frac{h_1 - c_i}{\sigma_i} \right) + 1 \right] \quad (47)$$

Case 3— $C_i \neq C_{ij}$, $\sigma_i < \sigma_{ij}$: once again, two sub cases arise similar to those of Case 2. When $c_{ij} > c_i$, the values of $\partial C(s_i \cap w_{ij}) / \partial c_{ij}$, $\partial C(s_i \cap w_{ij}) / \partial \sigma_{ij}$ and $\partial C(s_i \cap w_{ij}) / \partial \sigma_i$ are derived by differentiating (36) as follows:

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

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

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

Similarly, if $c_{ij} < c_i$

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

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 same as (49) and (50) respectively in both the conditions.

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

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

Copyright © 2010 SciRes.

$$
\begin{aligned}
&+\int_{h_1}^{h_2}\frac{\partial}{\partial c_{ij}}e^{-((x-c_j)/\sigma_i)^2}dx \\
&=-e^{-((h_1-c_{ij})/\sigma_{ij})^2} \\
&+e^{-((h_2-c_{ij})/\sigma_{ij})^2}
\end{aligned} \tag{52}
$$

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

$$
\begin{aligned}
\frac{\partial C(s_i\cap w_{ij})}{\partial \sigma_i} &= \int_{-\infty}^{h_1}\frac{\partial}{\partial \sigma_i}e^{-((x-c_{ij})/\sigma_{ij})^2}dx \\
&+\int_{h_1}^{h_2}\frac{\partial}{\partial \sigma_i}e^{-((x-c_i)/\sigma_i)^2}dx \\
&+\int_{h_2}^{\infty}\frac{\partial}{\partial \sigma_i}e^{-((x-c_{ij})/\sigma_{ij})^2}dx \\
&=\frac{h_1-c_i}{\sigma_i}e^{-((h_1-c_i)/\sigma_i)^2} \\
&-\frac{h_2-c_i}{\sigma_i}e^{-((h_2-c_i)/\sigma_i)^2} \\
&+\frac{\sqrt{\pi}}{2}\left[\left\{erf\left(\frac{\left(h_2-c_i\right)}{\sigma_i}\right)\right\}\right. \\
&\left.-\left\{erf\left(\frac{\left(h_2-c_{ij}\right)}{\sigma_{ij}}\right)\right\}\right]
\end{aligned} \tag{54}
$$

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.

Copyright © 2010 SciRes.