COMPARISON OF ROUGH MULTI LAYER PERCEPTRON AND ROUGH RADIAL BASIS FUNCTION NETWORKS USING FUZZY ATTRIBUTES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

HÜLYA VURAL

IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2004

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Canan Özgen
Director

I certified that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Ayşe Kiper
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Ferda Nur Alpaslan
Supervisor

Examining Committee Members

Assoc. Prof. Dr. Ferda N. Alpaslan	(METU, CENG)	
Prof. Dr. Mehmet Tolun	(Çankaya Uni.)	
Prof. Dr. Adnan Yazıcı	(METU, CENG)	
Assoc. Prof. Dr. İ. Hakkı Toroslu	(METU, CENG)	
Dr. Ayşenur Birtürk	(METU, CENG)	

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name:

Signature :

ABSTRACT

COMPARISON of ROUGH MLP and ROUGH RBF USING FUZZY ATTRIBUTES

Vural, Hülya
MS, Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. Ferda Nur Alpaslan

September 2004, 64 pages

The hybridization of soft computing methods of Radial Basis Function (RBF) neural networks, Multi Layer Perceptron (MLP) neural networks with back-propagation learning, fuzzy sets and rough sets are studied in the scope of this thesis. Conventional MLP, conventional RBF, fuzzy MLP, fuzzy RBF, rough fuzzy MLP, and rough fuzzy RBF networks are compared. In the fuzzy neural networks implemented in this thesis, the input data and the desired outputs are given fuzzy membership values as the fuzzy properties “low”, “medium” and “high”. In the rough fuzzy MLP, initial weights and near optimal number of hidden nodes are estimated using rough dependency rules. A rough fuzzy RBF structure similar to the rough fuzzy MLP is proposed. The rough fuzzy RBF was inspected whether dependencies like the ones in rough fuzzy MLP can be concluded.

Keywords: Radial Basis Function network, Multi Layer Perceptron, fuzzy sets, rough sets, hybrid soft computing.

ÖZ

HAM ÇKP ve HAM MÇDTF'NİN BELİRSİZ NİTELİKLERİN KULLANILARAK KARŞILAŞTIRILMASI

Vural, Hülya

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Doç. Dr. Ferda Nur Alpaslan

Eylül 2004, 64 sayfa

Bu tez çalışmasının erimi Merkez Çevre Doğrultulu Taban Fonksiyonları'nın (MÇDTF), geri yayılım yöntemi ile öğrenen Çok Katmanlı Pörseptron (ÇKP) konseptine sahip ağların, belirsiz setlerin, ve ham setlerin melezleştirilmesini kapsamaktadır. Geleneksel ÇKP, geleneksel MÇDTF, belirsiz ÇKP, belirsiz MÇDTF, ham belirsiz ÇKP ve ham belirsiz MÇDTF ağlar karşılaştırılmıştır. Bu tezde gerçekleştirilen belirsiz nöron konseptine sahip ağlardaki girdi bilgilerine ve istenilen çıktı bilgilerine "düşük", "orta" ve "yüksek" gibi belirsiz üyelik değerleri atanmıştır. Ham Belirsiz ÇKP için atanan başlangıç ağırlıkları ve yaklaşık optimal gizli düğüm sayısı ham bağımlılık kuralları kullanılarak tahmin edilmiştir. Ham belirsiz ÇKP'ye benzer bir ham belirsiz MÇDTF yapısı önerilmiştir. Bu ham belirsiz MÇDTF için, ham belirsiz ÇKP'de olanlara benzer bağımlılıkların varlığı araştırılmıştır.

Anahtar Kelimeler: Merkez Çevre Doğrultulu Taban Fonksiyonlu ağlar, Çok Katmanlı Perseptron, belirsiz setler, ham setler, melez elektronik hesaplamalar

To my dear family and to my love

ACKNOWLEDGEMENTS

I would like to express sincere appreciation to Assoc. Prof. Dr. Ferda Nur Alpaslan for her precious guidance and invaluable insight throughout the research.

I also would like to thank to my family, - especially to my sister, Hüsniye - for their ever-lasting support and encouragement.

Finally, my admiration goes to my love, Murat, for cheering me up all the time with love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

According to Lotfi A. Zadeh “Soft computing differs from conventional (hard) computing in that, unlike hard computing, it is tolerant of imprecision, uncertainty, partial truth, and approximation...”(1994). The basic soft computing methods are fuzzy logic, neural computing, and genetic programming. There are also many other soft computing methods like; rough sets, probabilistic reasoning, belief networks, chaos theory, and so on. This chapter is a short introduction to fuzzy sets, rough sets, and two feedforward neural networks which are radial basis function network and multi layer perceptron.

I.1. Radial Basis Function Networks

A Radial Basis Function (RBF) network is a two layered network, whose hidden layer activation function is a basis function, having linear output (see Figure 1). The basis function which produces a localized response can be Gaussian, Mexican hat, etc. Gaussian function of the form $e^{-(x^b/a^2)}$ is the mostly used basis function. The activation level of O_i of hidden node i is:

$$O_i = e^{[-(X-W_i).(X-W_i)/2\sigma_i^2]} \qquad (0.1)$$

where O_i is the activation level of hidden node i, X is the input vector, and W_i is the centre (weight vector) and σ_i^2 is the normalization factor of hidden node i respectively.

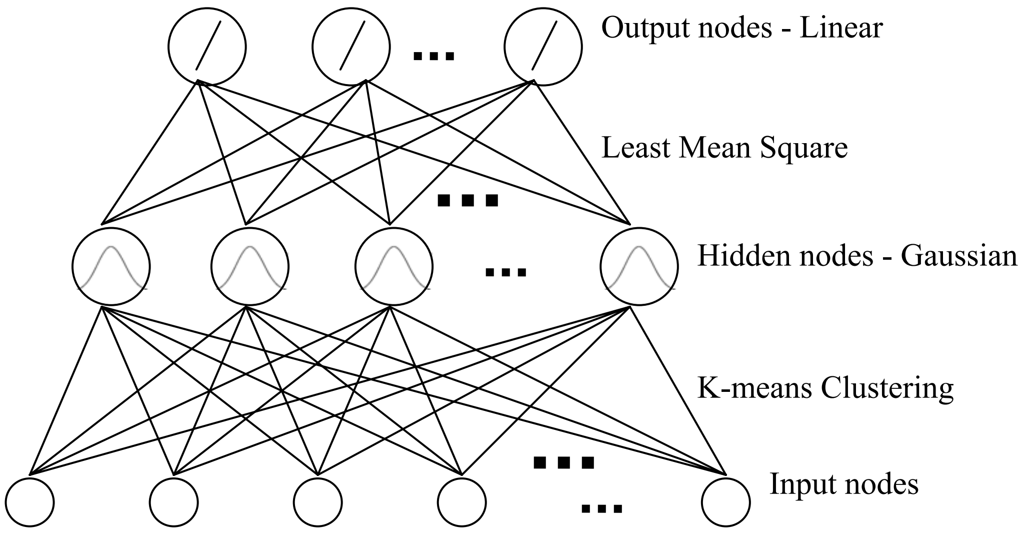


Figure 1 Radial Basis Function Network

Let O_i be the activation level of hidden node i and O_j be the activation level of output node j, the activation level of output node j is:

$$O_j = \sum W_{ji} O_i \tag{0.2}$$

where W_{ji} is the weight value between hidden node i and output node j

Normalization factor σ_i^2 is a measure of spread of data and calculated as follows:

$$\sigma_i^2 = \tfrac{1}{M} \sum (X - W_i).(X - W_i) \tag{0.3}$$

where X is the input vector clustered in hidden node i , W_i is the weight vector of hidden node i and M represents the number of inputs clustered in hidden unit i.

The learning of the hidden layer is carried out by unsupervised algorithms such as k-means clustering or self organized feature maps. Learning of the output layer is carried out by supervised algorithms like least mean square algorithm. [24]

I.1.1. K-means Clustering Algorithm

The k-means clustering is used for portioning N data points to K disjoint clusters. The algorithm tries to minimize J which is the sum of squares of the distance between the data points and the cluster centres that they are belonging to.

$$J = \sum_{j=1}^{K} \sum_{n \in C_j} \left| x_n - c_j \right|^2 \tag{0.4}$$

where K represents the number of clusters, C_j cluster j, x data point and c_j geometric centre of cluster j respectively.

Algorithm processes as follows:

- The dataset is partitioned into K clusters and the data points are randomly assigned to the clusters.
- For each data point:
    - Calculate the distance from the data point to each cluster.
    - Find the cluster that has the minimum distance and move the data point to that cluster.
- Repeat the above step until there is no change in the members of the clusters. [24]

I.1.2. Least Mean Square Algorithm

Least Mean Square (LMS) algorithm tries to minimize the mean square error between the desired output and the actual output. Let η be the learning rate and O_i be the actual activation level value for node i. Weight update rule for linear activation function is follows:

$$\begin{aligned} W_{ji}(t+1) &= W_{ji}(t) + \Delta W_{ji} \\ \Delta W_{ji} &= \eta \delta_j O_i \\ \delta_j &= T_j - O_j \end{aligned} \tag{0.5}$$

where T_i is the desired output value for output node i and O_j is the actual output value for node j. [24]

I.2. Feedforward Multi-Layer Perceptron

A Multi-Layer Perceptron is a two or more layered network, whose activation functions of all layers can be sigmoid, linear, ramp, etc. (see Figure 2). Sigmoid function is the most popular activation function with the formula

$$f(a) = \frac{1}{1+e^{-Ka}} \tag{0.6}$$

where a is the activation output of the neuron and K is the exponential constant.

The backpropagation algorithm is the most widely used training algorithm for multi-layered feedforward networks. [24]

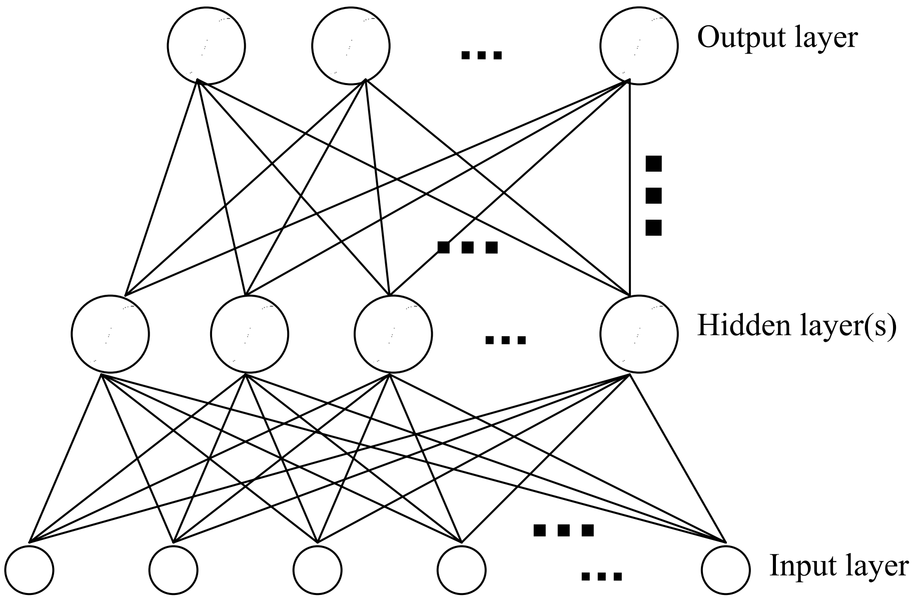


Figure 2 Multi-Layer Perceptron

I.3. Backpropagation Algorithm

The main aim of the backpropagation algorithm is to minimize the error between the desired output and the actual output. During convergence, the algorithm propagates the error backwards and updates the weights in accordance. [24]

Let O_i be the output value of node i which is one level below the node j and Θ_j be the bias parameter for node j. Output value of a hidden node or an output node O_j is as follows:

$$O_j = \frac{1}{1+e^{-(\sum W_{ji}O_i - \Theta_j)}} \tag{0.7}$$

where W_{ji}: Weight value between the node i and node j.

$$\begin{aligned} W_{ji}(t+1) &= W_{ji}(t) + \Delta W_{ji} + \alpha[W_{ji}(t) - W_{ji}(t-1)] \\ \Delta W_{ji} &= \eta \delta_j O_i \end{aligned} \tag{0.8}$$

where T_i is the desired output value for output node i, O_j is the actual output value for node i and δ_j is the error gradient of node j. The terms η and α represent the learning rate and the momentum term respectively.

Error gradient of output node j is:

$$\delta_j = O_j(1-O_j)(T_j - O_j) \tag{0.9}$$

Error gradient of a hidden node j is calculated as follows

$$\delta_j = O_j(1-O_j)\sum_k \delta_k W_{kj} \tag{0.10}$$

I.4. Fuzzy Sets

Fuzzy sets were introduced by Zadeh in 1965 [1]. Fuzzy sets can be seen as a generalized version of crisp (ordinary) sets. Fuzzy set theory basically extends the crisp set theory that has membership of its elements described by the classical characteristic function to allow for partial membership described by a membership function.

I.4.1. Fuzzy Set Membership

Let A be a set in universe X and $x \in X$. According to crisp set theory definition of set A is:

$$A(x) = \begin{cases} 1, & if\ x \in A \\ 0, & if\ x \notin A \end{cases} \tag{0.11}$$

Fuzzy set definition enables giving a degree of membership of an object to a set.

Definition: Let X be a space of points (objects), with a generic element of X denoted by x. A fuzzy set A in X is characterized by a membership function $\mu_A(x)$ which associates with each point x a real number in the interval [0,1] representing the grade of membership function of x in A. [3]

$$A = \{(x, \mu_A(x)); x \in X\} \tag{0.12}$$

where

$$\mu_A(x) : X \to [0,1] \tag{0.13}$$

Zadeh proposed another notion for fuzzy sets [4]:

$$A = \sum_{x \in X} \mu_A(x) / x \tag{0.14}$$

when universe of discourse is a finite set

$$A = \int_X \mu_A(x) / x \tag{0.15}$$

when universe of discourse is an infinite set.

As an example; in crisp sets a man with height 1.85 meters is accepted as "Tall", but a man with height 1.75 meters is accepted as short (see Figure 3). In fuzzy sets, a man with height 1.85 is a member of fuzzy set *Tall* to degree 1.0 and a man with height 1.75 is a member of fuzzy set *Tall* to degree 0.5 (see Figure 4).

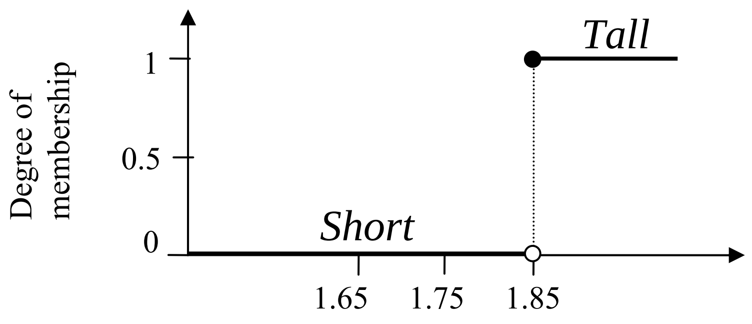


Figure 3 Crisp set representation of sets *Short* and *Tall*.

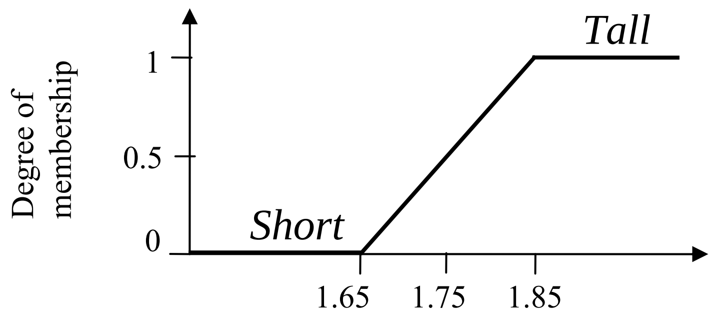


Figure 4 Fuzzy set representation of sets *Short* and *Tall*.

I.4.2. Fuzzy Membership Functions

Fuzzy set membership is defined by fuzzy membership functions those can be triangular, trapezoidal, $\prod_1$, gaussian, and so on. The formulas of the triangular, trapezoidal, $\prod_1$ membership functions are shown below [5]:

$$triangle(x:a,b,c)=\begin{cases} 0 & x<a \\ (x-a)/(b-a) & a\le x\le b \\ (c-x)/(c-b) & b\le x\le c \\ 0 & x>c \end{cases} \quad (0.16)$$

$$trapezoid(x:a,b,c,d) = \begin{cases} 0 & x < a \\ (x-a)/(b-a) & a \le x < b \\ 1 & b \le x < c \\ (d-x)/(d-c) & c \le x < d \\ 0 & x \ge d \end{cases} \tag{0.17}$$

$$\prod_1(x:a,b) = \frac{1}{1+\left(\frac{x-a}{b}\right)^2} \tag{0.18}$$

$\prod_1$ membership function is a symmetric function having value of $0.0.5$ at points $a-b$ and $a+b$. [5]

I.5. Rough Sets

Rough set theory was developed by Zdislaw Pawlak in 1982 [2]. The main goal of rough set analysis is to derive concepts using the data set which is called data table.

I.5.1. Information and Decision System

Data set is represented as a table whose rows are the objects and columns are the attributes. Information system $\mathcal{A}$ is represented as $\mathcal{A} = (U, A)$, where U (universe) is a non-empty set of objects and A is non-empty finite set of attributes such that $a : U \to Va$ for every $a \in A$, Va is the value set of a.

When there is a decision attribute in the data set then the relevant information system is called decision system. Decision system $\mathcal{A}$ is represented as follows: $\mathcal{A} = (U, A \cup \{d\})$ where d is the decision attribute such that $d \notin A$. The elements of A are called conditional attributes or conditions.

I.5.2. Indiscernibility

A decision system can become excessively large because of the repetition of data. This repetition can be seen in two ways; repetition of an object or repetition of a column. If two rows or two columns are indiscernible (same) then there won't be data loss if one of them is omitted.

To define the indiscernibility equivalence relation will be used. A binary relation $R \subseteq X \times X$ which is reflexive, symmetric, and transitive is called an "equivalence relation". The equivalence class of an element $x \in X$ consists of all objects $y \in X$ such that xRy.

Let $\mathcal{A} = (U, A)$ be an information system, then with any $B \subseteq A$ there is associated an equivalence relation $IND_{\mathrm{A}}(B)$ which is called B-indiscernibility relation:

$$IND_{\mathcal{A}}(B) = \{(x, x') \in U^2 \mid \forall a \in B, a(x) = a(x')\} \qquad (0.19)$$

When $(x, x') \in IND_{\mathcal{A}}(B)$, objects x and x' are indiscernible from each other by the attributes from B. The equivalence classes of the B-indiscernible relation are denoted as $[x]_B$.

Let $\mathcal{A} = (U, A \cup \{d\})$ be a decision system, if $(x, x') \in IND_{\mathcal{A}}(A)$ then omitting one of x or x' will not cause loss of data. This is the one way of reducing the data table; removing an object. [6]

I.5.3. Rough Set Approximation

Let $\mathcal{A} = (U, A \cup \{d\})$ be a decision system where d is the decision attribute, A is the attribute set $A = \{a_1, a_2\}$, U is the universe $U = \{F_1, F_2, F_3, F_4, F_5, F_6\}$ as seen in Table 1. Let set X be a rough set such that $X \subset U$. In crisp sets, an object can be an element of a set or not, but in rough sets, an object can be certainly an element, certainly not an element, or roughly an element of set as seen in the Figure 5. In Table 1, F_1 and F_2 are indiscernible according to attributes a_1 and a_2. The objects

F_3 and F_4 are also indiscernible according to attributes a_1 and a_2. In Figure 5, the objects in the white shaded area are exactly the element of set X and the objects in the black shaded area are exactly not the element of set X. The objects in the grey part can not be decisively classified into set X on the basis of knowledge in A.

Table 1 Decision table of $\mathcal{A}$

	a_1	a_2	d
F_1	1	1	1
F_2	1	1	0
F_3	1	0	1
F_4	1	0	1
F_5	0	1	0
F_6	0	2	0

In Figure 5, each of $\{F_3, F_4\}$, $\{F_1, F_2\}$, $\{F_5\}$ and $\{F_6\}$ is an equivalence class of set X according to attribute set A.

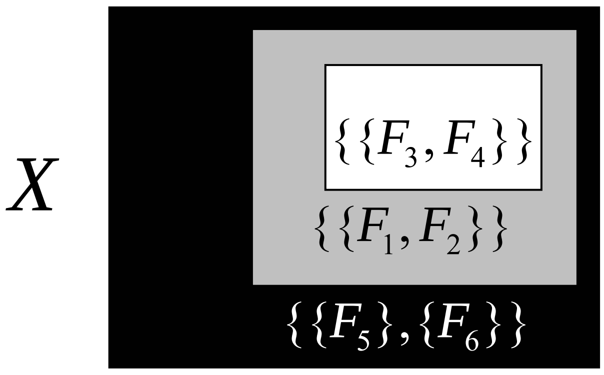


Figure 5 Rough set approximation of decision system $\mathcal{A}$

Set X can be approximated according to attributes contained in A. As seen in Figure 5 white shaded area is the A-lower approximation of X and denoted by $\underline{A}X$ such that $\underline{A}X = \{x \mid [x]_A \subseteq X\}$. White union grey shaded area is the A-upper approximation of X, denoted by $\overline{A}X$ such that $\overline{A}X = \{x \mid [x]_A \cap X \neq 0\}$.

The A boundary region of X is $BN_A(X) = \overline{A}X \Leftrightarrow \underline{A}X$, containing the objects that are not certainly the element of X when approximated by the attributes of A. If the boundary region of a set is empty then the set is a crisp (conventional set). If a set has a non-empty boundary region then that set is a rough set. Accuracy of set approximation of a rough set is:

$$\alpha_A(X) = {|\underline{A}(X)|} \Big/ {|\overline{A}(X)|} \tag{0.20}$$

Quality of approximation of X by A which expresses the percentage of objects correctly classified, is:

$$\gamma_A(X) = {|\underline{A}(X)|} \Big/ {|U|} \tag{0.21}$$

As a concrete example assume an indecisive set *Tall*. Let the Table 2 be the decision matrix of set *Tall* which depends on attributes *gender* and *height*.

Table 2 Decision table of set *Tall* with two conditional attributes

	gender	*height*	*Tall*
F_1	Woman	1.70	YES
F_2	Woman	1.70	NO
F_3	Woman	1.50	NO
F_4	Man	1.90	YES
F_5	Man	1.65	NO

In Figure 6 the object F_4 is the element of rough set $Tall$. The objects F_3 and F_5 are exactly not the elements of set $Tall$. The objects F_1 and F_2 are at the boundary of rough set $Tall$. This means that only the attributes $gender$ and $height$ are not enough to represent the set $Tall$.

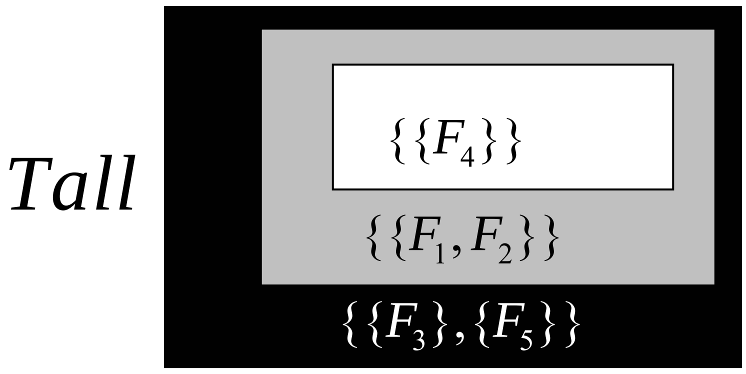


Figure 6 Rough set approximation of set $Tall$ with two attributes

If one more attribute is added to set $Tall$, it becomes a decisive set. Let the Table 3 be the decision matrix of set $Tall$ which depends on attributes $gender$, $height$ and $nationality$.

Table 3 Decision table of set $Tall$ with three conditional attributes

	$gender$	$height$	$nationality$	$Tall$
F_1	Woman	1.70	China	YES
F_2	Woman	1.70	Russia	NO
F_3	Woman	1.50	Russia	NO
F_4	Man	1.90	China	YES
F_5	Man	1.65	Russia	NO

In Figure 7 the objects F_1 and F_4 are the element of rough set $Tall$. The objects F_2, F_3 and F_5 are exactly not the element of set $Tall$.

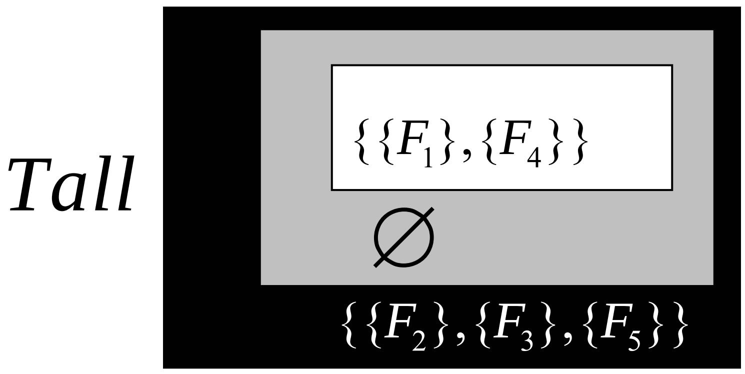


Figure 7 Rough set approximation of set $Tall$ with three attributes

I.5.4. Reducts

A reduct of information system $\mathcal{A}$ is a minimal set of attributes $B \subseteq A$ such that $IND_{\mathcal{A}}(B) = IND_{\mathcal{A}}(A)$. One way of reducing the data table is removing superfluous attributes. The superfluous attributes are $A - B$ for a reduct consisting of elements of B. In order to find a set of superfluous attributes having the maximum cardinality, minimal reduct has to be found. Finding minimal reduct is a NP-hard problem.

Discernibility function, which is derived by using discernibility matrix, is used for finding the set of all reducts of $\mathcal{A}$. Discernibility matrix for the information system $\mathcal{A}$ with n objects and m attributes has the elements c_{ij}. Each c_{ij} element is the list of attributes upon which the objects x_i and x_j differ; $c_{ij} = \{a \in A \mid a(x_i) \neq a(x_j)\}$ for $i, j = 1, \ldots, n$. Discernibility function

$f_{\mathrm{A}}(a_1^*,\ldots,a_m^*) = \wedge\{\vee c_{ij}^* \mid 1 \le j \le i \le n, c_{ij} \ne \varnothing\}$, where $c_{ij}^* = \{a^* \mid a \in c_{ij}\}$ and $(a_1^*,\ldots,a_m^*)$ are the boolean variables corresponding to attributes $(a_1,\ldots,a_m)$. [6]

I.5.5. Attribute Dependency

Given a decision system $\mathcal{A} = (U, A \cup \{d\})$, the cardinality of the image $d(U) = \{k \mid d(x) = k, x \in U\}$ is called the rank of d and denoted by $r(d)$. Let the set V_d of values of decision d is equal to $\{v_d^1,\ldots,v_d^{r(d)}\}$.

The decision attribute d determines the partition $CLASS_{\mathcal{A}}(d) = \{X_{\mathcal{A}}^1,\ldots,X_{\mathcal{A}}^{r(d)}\}$ of the universe U, where $X_{\mathcal{A}}^k = \{x \in U \mid d(x) = v_d^k\}$ for $1 \le k \le r(d)$. $CLASS_{\mathcal{A}}(d)$ is called the classification of objects in $\mathcal{A}$ determined by the decision d. The set $X_{\mathcal{A}}^i$ is called the i th decision class of $\mathcal{A}$. By $X_{\mathcal{A}}(u)$ we denote the decision class $\{x \in U \mid d(x) = d(u)\}$, for any $u \in U$. If $X_{\mathcal{A}}^1,\ldots,X_{\mathcal{A}}^{r(d)}$ are the decision classes of $\mathcal{A}$, then the set $\underline{B}X_1 \cup \ldots \cup \underline{B}X_{r(d)}$ is called the B-positive region of $\mathcal{A}$ and denoted by $POS_B(d)$ [6].

Let $C \subseteq A, D \subseteq A$, D totally depends on C, denoted by $C \Rightarrow D$, if all values of attributes from D are uniquely determined by values of attributes from C. Degree of dependency of D on C is denoted by and $C \Rightarrow_k D$ [6],

$$k = \gamma(C,D) = \frac{|POS_C(D)|}{|U|} \qquad (0.22)$$

I.5.6. Rough Membership

The rough membership function quantifies the degree of relative overlap between the set X and the equivalence $[x]$ class to which x belongs. It is defined as follows: [6].

$$\mu_X^B : U \to [0,1] \text{ and } \mu_X^B(x) = \frac{\left|[x]_B \cap X\right|}{\left|[x]_B\right|} \qquad (0.23)$$

The rough membership function is the probability that object x is an element of set X regarding the frequencies according to attribute set B.

A concrete example of rough set can be seen in Table 4. The rough set membership of the object F_{101} with attribute values (0,1) is $\mu_X^A(F_{101}) = 1$, because it is exactly the element of set X. The rough set membership of the object F_{301} with attribute values (0,0) is $\mu_X^A(F_{301}) = 0$, because it is exactly not the element of set X. The rough set membership value of an object with value 1 for attribute a_1 and value 1 for attribute a_2 is 0.99.

Table 4 Decision table of set X

	a_1	a_2	X	Number of Data Samples
$F_1 - F_{99}$	1	1	YES	99
F_{100}	1	1	NO	1
$F_{101} - F_{300}$	0	1	YES	200
$F_{301} - F_{310}$	0	0	NO	10

In Figure 8, the equivalence classes of rough set X are $\{\{F_{101}, F_{102}, \ldots, F_{300}\}\}$, $\{\{F_1, F_2, ..., F_{99}\}, F_{100}\}$ and $\{\{F_{301}, F_{302}, \ldots, F_{310}\}\}$. The objects $F_1, F_2, ..., F_{99}$ are closer to set X then the object F_{100} in some manner that is not obvious by the current attributes. When the number of points closer to lower boundary of rough set X increases, the rough set memberships of the points in the boundary also increase.

X

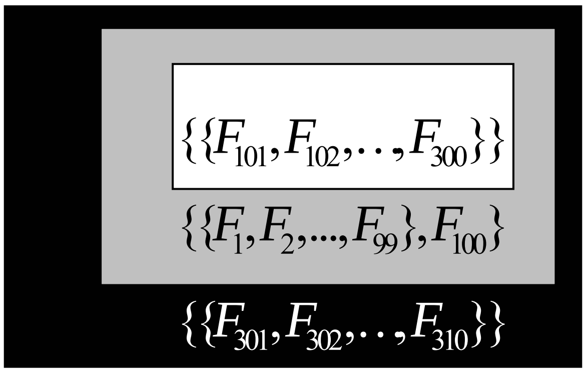

$\{\{F_{101}, F_{102}, \ldots, F_{300}\}\}$
$\{\{F_1, F_2, \ldots, F_{99}\}, F_{100}\}$
$\{\{F_{301}, F_{302}, \ldots, F_{310}\}\}$

Figure 8 Rough set approximation of set X

I.6. Relationship between Rough Sets and Fuzzy Sets

"Rough set theory and fuzzy set theory are complementary. It is natural to combine the two models of uncertainty (vagueness for fuzzy sets and coarseness for rough sets) in order to get a more accurate account of imperfect information" [25].

In rough set theory, approximations of sets are defined according to a background knowledge given by a data table. The rough membership function μ_X^B, where $X \subseteq U$ and $B \subseteq A$, can be used to define approximations and boundary region of a set, as shown below [6]:

$$\underline{B}(X) = \{x \in U : \mu_X^B(x) = 1\} \tag{0.24}$$

$$\overline{B}(X) = \{x \in U : \mu_X^B(x) > 0\} \tag{0.25}$$

$$BN_B(X) = \{x \in U : 0 < \mu_X^B(x) < 1\} \tag{0.26}$$

The rough membership function has the following properties [26]:

1. $\mu_X^B(x) = 1$ iff $x \in \underline{B}(X)$
2. $\mu_X^B(x) = 0$ iff $x \in -\overline{B}(X)$
3. $0 < \mu_X^B(x) < 1$ iff $x \in BN_B(X)$

4. If $IND(B) = \{(x,x) : x \in U\}$, then $\mu_X^B(x)$ is the characteristic function of X
5. If $xIND(B)y$, then $\mu_X^B(x) = \mu_X^B(y)$
6. $\mu_{U-X}^B(x) = 1 - \mu_X^B(x)$ for any $x \in U$
7. $\mu_{X \cup Y}(x) \geq \max(\mu_X^B(x), \mu_Y^B(x))$ for any $x \in U$
8. $\mu_{X \cap Y}^B(x) \leq \min(\mu_X^B(x), \mu_Y^B(x))$ for any $x \in U$
9. If X is a family of pairwise disjoint sets of U, then for any $x \in U$ $\mu_{\bigcup \mathrm{X}}^B(x) = \sum_{X \in \mathrm{X}} \mu_X^B(x)$

From the above properties, the 7th and the 8th one show that rough sets can be seen as a generalized version of fuzzy sets. Also it has been shown in [26] that if the 7th and the 8th property above become $\mu_{X \cup Y}^B(x) = \max(\mu_X^B(x), \mu_Y^B(x))$ for any $x \in U$ and $\mu_{X \cap Y}^B(x) = \min(\mu_X^B(x), \mu_Y^B(x))$ for any $x \in U$ respectively, they are not true in general. Besides, the rough membership function, in contrast to fuzzy membership function, has a probabilistic flavor.

CHAPTER III

COMPARISON of ROUGH MLP and ROUGH RBF USING FUZZY ATTRIBUTES

II.1. Background

There have been lots of hybridizations of soft computing methods. In these hybridizations, fuzzy Neural Networks have the first rank in number of work performed until now. The network structure in [7] was selected as a model in fuzzy MLP with Back Propagation (fuzzy BPNN). In [7], the input vector was fuzzified to represent fuzzy linguistic properties low, medium and high. The desired output vector was also fuzzified, so it can take values in the range [0,1]. The resultant fuzzy MLP was superior to conventional MLP and Bayes classifier on highly overlapping data sets. Unlike the fuzzy BPNN used in this thesis, a decay factor [16] was used in calculation of change in weights.

In [14], a fuzzy RBF was proposed. In the hidden layer of the RBF network, fuzzy basis functions were used and the parameters of these functions were tuned by genetic algorithms. The proposed controller did better than a PID controller.

Another fuzzy RBF was defined in [15]. This fuzzy RBF constructed the base used in this thesis, but the fuzzification functions of these two approaches were different. The fuzzification function used in [7] is included in the fuzzy RBF structure used in this thesis.

Rough sets are the newest paradigm of soft computing and there is a tremendous amount of research ongoing on this topic recently [10, 11, 18].

R. Yasdi proposed a method in [17] to construct weight values of MLP according to dependency rules generated by rough sets. The method first finds reducts of the system. Secondly, it simplifies the indiscernibility matrix according to this reduct. The reduced indiscernibility matrix is used for finding the dependency rules and the dependency factors of these rules are used for generating weight values of the neural network.

In this thesis, fuzzy-rough membership concept was used in RBF [13]. The hidden node outputs are the fuzzy membership values of the input. The weights between the hidden nodes and the output layer correspond to rough fuzzy membership functions. The output of the network is rough-fuzzy membership value of the input data.

There are many combinations of rough sets and fuzzy sets. Some of these are on rough fuzzy set or membership [9] and some are on fuzzy-rough set or membership [19].

In [22], using rough sets, a knowledge base was determined. By the help of this knowledge base, number of hidden nodes in MLP is calculated. Additionally, initial weight values of the neural network are also found according to the dependency factors of the dependency rules of this knowledge base. This network structure was taken as the model to the rough-fuzzy BPNN used in this thesis.

In [12], quantitative input data was fuzzified to get fuzzy set of linguistic terms. Rough lover and upper bounds are calculated for the fuzzy set of linguistic terms. Fuzzy rules were generated while the missing attributes were being estimated.

The c-means clustering is dependent on the initial cluster centres and the sequence of the data introduced to the c-means. In [20] subtractive learning was used to overcome the dependency of c-means onto initial cluster centres. Also the number of clusters does not need to be given to the algorithm, it concludes to an optimum solution itself.

All the soft computing methods, which were explained in ‘Chapter I’, have their own strengths and weaknesses. The feedforward neural networks are good at handling error prone data, but their black-box structure prevents obtaining human interpretative rules. Fuzzy logic enables human type reasoning but it needs expert knowledge to construct rules. It is possible to construct generalized rules by means of rough sets by reducing the size of data table. Combining fuzzy and rough sets leads to more accurate account of imperfect information.

Mostly combination of these methods do better than when they are alone. The absence of a property in one method can be found in the other method. In this thesis MLP, RBF, fuzzy MLP, fuzzy RBF, rough fuzzy MLP and rough fuzzy RBF will be compared.

II.2. Fuzzy Neural Networks

Each input feature is represented with membership values of overlapping linguistic properties low, medium and high. This representation enables easier human interpretation and also more appropriate way of showing overlapping data. In conventional neural networks, an output vector is expected to have the value 1 at one node and value 0 at the other nodes. In a fuzzy neural network output membership function is in the interval [0, 1], which allows a better classification when the feature class has overlapping pattern classes.

II.2.1. Input Fuzzification

Each input feature $\vec{F}_i = [F_{i1,}F_{i2},\ldots,F_{in}]$ is represented in 3n-dimensional vector, where F_{in} represents the n^{th} attribute of input feature F_i. After the fuzzification, the fuzzy form of the input vector becomes

$$\vec{F}_i = [\mu_{low(F_{i1})}(\vec{F_i}), \mu_{medium(F_{i1})}(\vec{F_i}), \mu_{high(F_{i1})}(\vec{F_i}), \ldots, \mu_{high(F_{in})}(\vec{F_i})] \qquad (0.27)$$

where $\mu_{low(F_{i1})}(\vec{F_i})$ represents the fuzzy membership value of F_{i1} to fuzzy property "low", $\mu_{medium(F_{i1})}(\vec{F_i})$ represents the fuzzy membership value of F_{i1} to fuzzy property "$medium$", and $\mu_{high(F_{i1})}(\vec{F_i})$ represents the fuzzy membership value of F_{i1} to fuzzy property "$high$". These membership values are calculated by using π membership function

$$\pi(r;c,\lambda)=\begin{cases}2\left(1-\dfrac{\|r-c\|}{\lambda}\right)^2 & for \quad \dfrac{\lambda}{2}\le\|r-c\|\le\lambda\\ 1-2\left(\dfrac{\|r-c\|}{\lambda}\right)^2 & for \quad 0\le\|r-c\|\le\dfrac{\lambda}{2}\\ 0 & otherwise\end{cases} \tag{0.28}$$

such that $\lambda > 0$. The parameters c and λ represent the centre and the radius of the π function respectively.

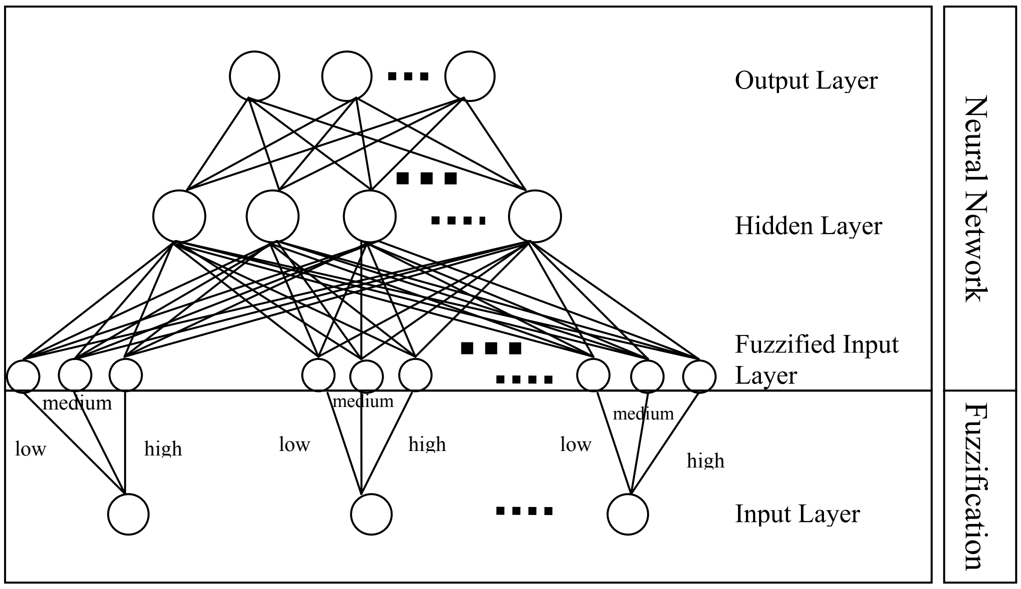


Figure 9 Structure of the process with fuzzification and neural network parts

As seen in Figure 9, the data coming from the data table are first fuzzified and the result obtained in 3n-dimensional vector is used as the input layer of the neural network.

In this thesis three π functions, $\pi_{low}(r;c_{low},\lambda_{low})$, $\pi_{medium}(r;c_{medium},\lambda_{medium})$, $\pi_{high}(r;c_{high},\lambda_{high})$ has been chosen in fuzzification to be consistent with the work done in [7] to construct an objective comparison.

Let the data table has L input patterns and, $\overrightarrow{F_i} = [a_1, a_2, \ldots, a_i, \ldots, a_n]$ where a_i is the i^{th} attribute. In the scope of L patterns, $a_{i\min}$ and $a_{i\max}$ denote minimum and maximum value of the attribute a_i respectively. Let $\overrightarrow{F_i} = [F_{i1}, F_{i2}, \ldots, F_{ij}, \ldots, F_{in}]$, where F_{ij} is the j^{th} attribute of input feature $\overrightarrow{F_i}$ and $\overrightarrow{F_j}$ represents the j^{th} attribute of all L pattern points. Then $\overrightarrow{F_j} = [F_{1j}, F_{2j}, \ldots, F_{ij}, \ldots F_{Lj}]$, where F_{ij} is the j^{th} attribute of input feature $\overrightarrow{F_i}$.

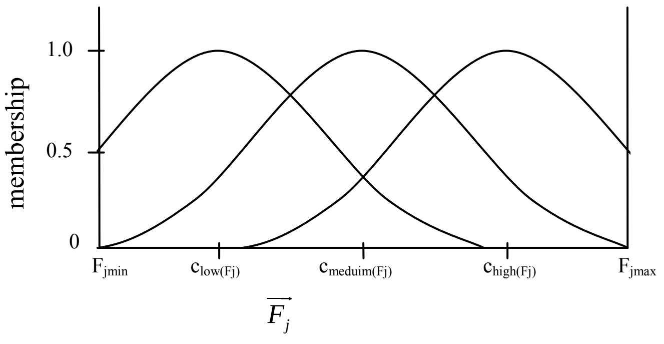


Figure 10 The π functions for linguistic properties low, medium and high.

As seen in Figure 10, $F_{j\min}$ and $F_{j\max}$ represent the minimum and the maximum values in F_j respectively. Centre (c) and radius (λ) of the π functions are computed as follows:

$$\begin{aligned} \lambda_{medium(F_j)} &= \frac{1}{2}(F_{j\max} - F_{j\min}) \\ c_{medium(Fj)} &= F_{j\min} + \lambda_{medium(F_j)} \end{aligned} \quad (0.29)$$

$$\begin{aligned} \lambda_{low(F_j)} &= \frac{1}{fdenom}(c_{medium(F_j)} - F_{j\min}) \\ c_{low(Fj)} &= c_{medium(F_j)} - 0.5\lambda_{low(F_j)} \end{aligned} \quad (0.30)$$

$$
\begin{aligned}
\lambda_{high(F_j)} &= \frac{1}{fdenom}(F_{j\max} - c_{medium(F_j)}) \\
c_{high(Fj)} &= c_{medium(F_j)} + 0.5\lambda_{high(F_j)}
\end{aligned}
\qquad (0.31)
$$

The parameter $fdenom$ is used for controlling the overlapping of the fuzzy membership functions. The above c and λ computations ensure that a feature vector F_{ij} has a value greater than 0.5 in one of the $\mu_{low(F_{ij})}(\vec{F_i})$, $\mu_{medium(F_{ij})}(\vec{F_i})$, $\mu_{high(F_{ij})}(\vec{F_i})$. So each feature value has a strong membership in one or more of the properties low, medium, and high.

II.2.2. Output Fuzzification

In conventional neural networks, desired output has to be a distinct value. Although there can be points on the boundaries or in the overlapping sets, winner takes all mechanism is applied. In these fuzzy neural networks, desired output is calculated by means of fuzzy data and takes values in the range $[0,1]$. As a result desired output can take a value greater than 0 in more than one output node. This representation is highly suitable for learning data with so much overlapping.

Output fuzzification is the process of finding the membership of an input data to each output class. While finding this membership, weighted distance of the input to the corresponding output class is used.

Table 5 An ℓ-output-class sample train data table

	$\vec{F}_j$			
	$F_{1,1}$	$\dots$	$F_{1,n-1}$	$F_{1,n}$
	$F_{2,1}$	$\dots$		
$\vec{F}_i$	$\vdots$		$\vdots$	$\vdots$
	$F_{L-1,1}$	$\dots$	$F_{L-1,n-1}$	$F_{L-1,n}$
	$F_{L,1}$	$\dots$	$F_{L,n-1}$	$F_{L,n}$

Assume training data in Table 5 is given, where $\overrightarrow{F_i}$ represents i^{th} object, and $\overrightarrow{F_j}$ represents attribute vector j. Let for each output class k, O_k be the mean and V_k be the standard deviation vectors with n dimensions. Let Z_{ik} be the weighted distance of object F_i from the k^{th} class. Then

$$Z_{ik} = \begin{cases} \sqrt{\sum_{j=1}^{n} \left[\frac{F_{ij} - O_{kj}}{V_{kj}} \right]^2} & , V_{kj} > 0 \text{ , for } k = 1, \ldots, l \\ \sqrt{\sum_{j=1}^{n} \left[F_{ij} - O_{kj} \right]^2} & , V_{kj} = 0 \text{ , for } k = 1, \ldots, l \end{cases} \tag{0.32}$$

Let $\mu_k\left(\overrightarrow{F_i}\right)$ represent the membership of the input object i to output class k. Then

$$\mu_k\left(\overrightarrow{F_i}\right) = \frac{1}{1 + \left(\frac{z_{ik}}{f_d}\right)^{f_e}} \quad , \quad \text{where } f_e > 0, f_d > 0 \tag{0.33}$$

The variables f_d (denominational fuzzy generator) and f_e (exponential fuzzy generator) are used to control the fuzziness of the membership.

In the training data set, there can be objects with the same attributes except the decision attribute. This case is mostly seen in points which reside through the overlapping boundaries.

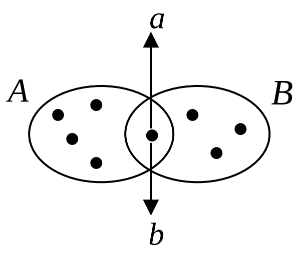


Figure 11 Train data with two output classes, A and B

In the example in Figure 11, point a and point b are the same points. They are at the intersection set of set A and set B. According to input data set, the decision attribute of point a is A and the decision attribute of point b is B. In the test data set, if any point x takes place at the intersection of set A and B, the decision attribute of point x should better be element of both A and B. In conventional neural networks, point x can be either element of A or B, but not both.

In the fuzzy neural network approach, point a (or b) is used for calculating mean and standard deviation of both the set A and the set B. So, in the test set, if a point with the same characteristics of point a (or b) is seen, it is said to be both the element of the set A and the set B.

The fuzziest case is defined as the case that a point belongs to all output classes of the train set. Fuzziest cases need extra computation to enhance difference. As a result, the desired output of the j^{th} output node is [7]

$$d_j = \begin{cases} \mu_{INT(k)}\left(\overrightarrow{F_i}\right) = \begin{cases} 2\left[\mu_k\left(\overrightarrow{F_i}\right)\right]^2 & \text{, for } 0 < \mu_k \\ 1-2\left[\mu_k\left(\overrightarrow{F_i}\right)\right]^2 & \text{,otherwise} \end{cases} & \text{, } \textit{for the fuzziest case} \\ \mu_k\left(\overrightarrow{F_i}\right) & \text{,} \textit{otherwise} \end{cases} \tag{0.34}$$

where the fuzzy membership of the input object i to output class k ($\mu_k\left(\overrightarrow{F_i}\right)$) is given in equation (0.29). In equation (0.30), the fuzzy membership of the input object i to output class k for the fuzziest case is represented with $\mu_{INT(k)}\left(\overrightarrow{F_i}\right)$.

II.3. Rough Fuzzy Neural Network Structure

Embedding rough sets into fuzzy neural networks enables deciding the number of nodes in the hidden layer. It also gives rough weight values, which can be used as initial weight in the BPNN. The steps of finding dependency rules are given in Table 6. [17]

Each group found in the 2nd step of the Table 6 is divided into subgroups having same decision attributes. Unlike [22], Tr of the group α in step 2 is the mean value of cardinalities of the sub groups that take place in group α. The variable Tr in step 3 of Table 6 is calculated for each output class.

Table 6 Dependency rule generation

```
For each output class:
1. Fuzzify the input data.
2. Group data having the same conditional attributes.
3. Apply threshold $Tr$ to grouped fuzzy data and take the groups having
   cardinality more than or equal to $Tr$.
4. Construct a decision matrix using a representative element from each
   selected group.
5. Find the decision function of this decision matrix.
6. Turn this decision function into Disjunctive Normal Form (DNF) and
   simplify it; e.g. $(L_1 \wedge M_1) \vee (M_2 \wedge H_2)$.
7. Select a minimum reduct. Each element of "$\vee$" operator in the DNF form
   is a reduct; e.g. $(L_1 \wedge M_1)$.
8. Reduce the discernibility matrix using the selected reduct [22].
9. Construct a function from the reduced discernibility matrix for each object
   [17].
10. Apply "$\vee$" operator to combine the results in step 9.
11. Simplify the result in 10 to get the dependency rule for the class selected.
```

The number of hidden nodes in neural network can be decided by the dependency rules generated for each class. With the help of this method, tremendous search for finding optimum number of hidden nodes is prevented. Assume the dependency rule found for output class k in Table 6 is

$M_2 \vee (M_1 \wedge H_1)$. Then there will be two hidden nodes for output class k in rough fuzzy neural network. One of the hidden nodes is for M_2 and the other is for $(M_1 \wedge H_1)$.

The dependency factors of the dependency rules found in Table 6 are used as the initial weight values of the fuzzy BPNN. The structure of the fuzzy BPNN for the output class k is seen in Figure 12, where α and β are very small numbers. They are included to the network to add randomization in weight values. In Figure 12 only the output node k is shown. Although the rough fuzzy BPNN has a fully connected structure, only the calculated weights for the output node k are shown in the figure. The other weights are small random values. Signs of the all weight values are given randomly.

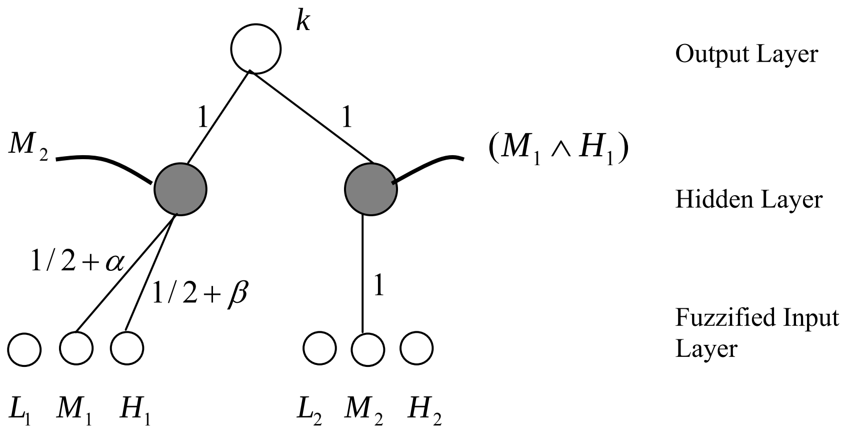


Figure 12 Sample rough fuzzy BPNN

II.4. Implementation and Results

The BPNN and RBF included in this thesis are in their conventional structures as explained in Chapter I. In this thesis the fuzzy BPNN is same as the one in [7]. The rough fuzzy BPNN is same as the one in [22] except the calculation of threshold Tr used in rough set dependency rule generation. Fuzzy RBF is a mixture of [15] and [22]. It takes the fuzzification function from [22] and the structure from [15]. In literature, the rough fuzzy RBF proposed in section II.3 is the attempt on hybridization of rough, fuzzy, and RBF soft computing methods. The comparison of BPNN, fuzzy BPNN, and rough fuzzy BPNN and RBF, fuzzy RBF, and rough fuzzy RBF is done in the scope of this thesis.

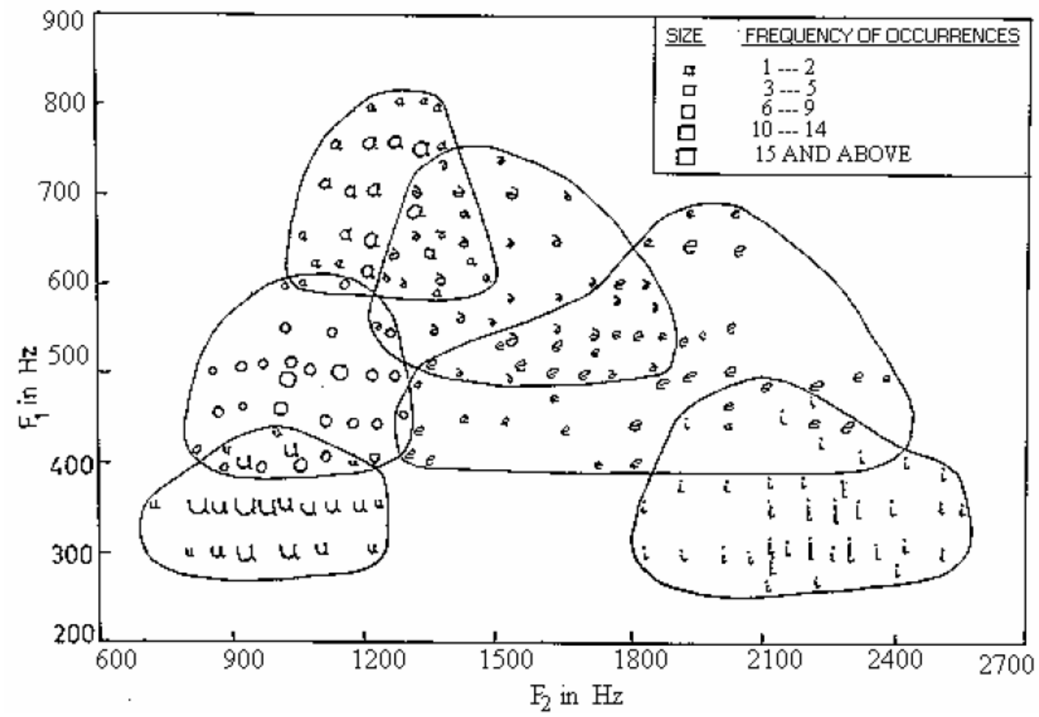


Figure 13 Projection of vowel data set onto F_1 and F_2.

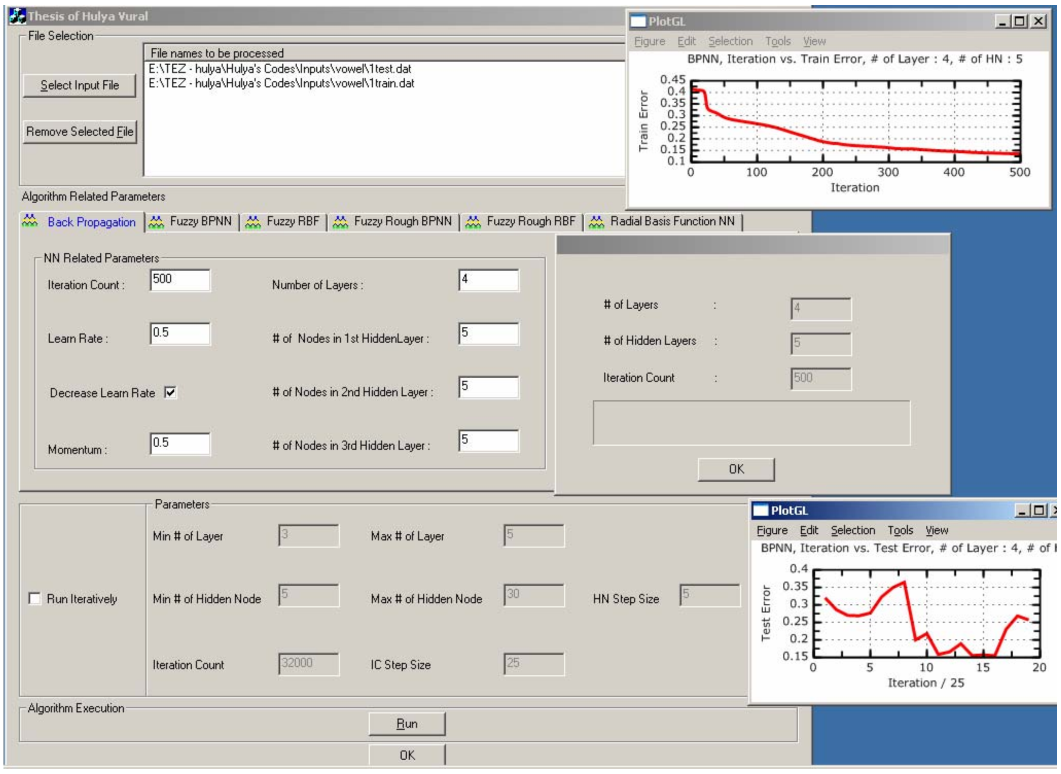
During the comparison, three different data sets are used. The first one is the "Vowel Data" taken from http://isical.ac.in/~sushmita, which includes Indian Telugu vowel sounds. The sounds are represented in three different frequencies and the results are six different vowel sounds. There are 871 objects in the data set, 67% of these sounds are used for training and the rest are left for the testing process. This data exhibits a highly overlapping structure. The vowels which are ∂, a, i, u, e, o, are shown in Figure 13 according to their frequency of occurrences [8, 21]. Actually, there are three frequencies in the data set, but in the figure only the 2 dimensional projection on frequency 1 (F_1) - frequency 2 (F_2) space is shown.

The second data set "Pat1" is also taken from http://isical.ac.in/~sushmita. It is a synthetic data with 2 attributes and 3 different output classes. The data is linearly inseparable and has 880 objects. In the training phase 67% of the data set is used, and in the testing phase the remaining 33% of the data is used.

The third data set used is the "Iris Plants Database" prepared by R.A. Fisher and Michael Marshall [23]. It has 4 attributes and 3 output classes. One of these output classes is linearly separable, whereas the others are not. This is the simplest data set used in the content of this thesis.

The fourth data set is "EnglishVow" which is also known as "vowel.data". The sources of "EnglishVow" are D. Deterding, M. Niranjan, and T. Robinson [23]. The data set is a speaker independent recognition of the eleven steady state vowels of British English using a specified training set of lpc derived log area ratios. There are total of 990 instances in the data set with 10 attributes. The data set is divided into 11 output classes.

In order to find the best results of the mentioned algorithms an automated testing was generated. A range for the number of layers and the number of hidden nodes can be defined in the user interface. The lower boundary and the upper boundary for the range of layers are taken from the user interface. The application is executed for each number between the boundaries and at the boundaries. The lower bound and the upper bound and also the hidden node step-size are defined for the range of hidden number of nodes in each layer. The application is executed for the lower bound and the previous number of hidden node plus the hidden node step-size until the number of hidden nodes exceeds the upper boundary. The number of iterations and the step size for the iterations ("IC Step Size") can also be defined during the automated execution. The iteration count step-size decides on the frequency of the test performed.



File Selection

File names to be processed
E:\TEZ - hulya\Hulya's Codes\Inputs\vowel\1test.dat
E:\TEZ - hulya\Hulya's Codes\Inputs\vowel\1train.dat

BPNN, Iteration vs. Train Error, # of Layer : 4, # of HN : 5

Algorithm Related Parameters

NN Related Parameters

Iteration Count : 500
Number of Layers : 4
Learn Rate : 0.5
# of Nodes in 1st HiddenLayer : 5
Decrease Learn Rate ✓
# of Nodes in 2nd Hidden Layer : 5
Momentum : 0.5
# of Nodes in 3rd Hidden Layer : 5

# of Layers : 4
# of Hidden Layers : 5
Iteration Count : 500

Parameters

Run Iteratively ☐
Min # of Layer 3
Max # of Layer 5
Min # of Hidden Node 5
Max # of Hidden Node 30
HN Step Size 5
Iteration Count 32000
IC Step Size 25

BPNN, Iteration vs. Test Error, # of Layer : 4, # of I

Algorithm Execution

Figure 14 Snapshot of the program.

If the “Run Iteratively” checkbox in Figure 14 is checked, then the program executes for each number of layers for each number of hidden nodes for iterations time. The number of tests performed is equal to $\lfloor \text{Iteration Count} \,/\, \text{IC Step Size} \rfloor$. A test graphic and a train graphic are drawn after number of iterations is defined. As this process can be very long especially for the BPNN, a progress bar is added to the application to see the progress.

Mean square error is used as the criteria to the evaluation of the results. In the test graphs included below, the behavior of the mean square error is analyzed according to the number of iterations carried out during the training. For every “IC Step Size” iterations, a test error is calculated. As the number of train iterations increases the train error decreases. Most of the times, this is the case for the test error after a settlement in learning, unless there is an oscillation.

The “Vowel” and the “Pat1” data are executed for each of the six algorithms mentioned. The “Iris Plant Database” and “EnglishVow” data are used for rough fuzzy RBF. When the distributions of test and training data sets change, the results also differ. So the test and training sets of “Vowel” and “Pat1” data were randomly selected for three times. The resultant data are called “vowel1”, “vowel2”, “vowel3” for the versions of “Vowel” data and “pat1_1”, “pat1_2”, “pat1_3” for the versions of “Pat1” data. For this randomization and split operations two different implementations were generated. The files are first randomized then split into train and test files with the percentage of 67 for train and 33 for the test.

As an analysis and inspection tool for the results of the testing and training, Plot Graph Library (PGL) is integrated into the program. PGL is a graphic library especially specialized in scientific chart drawings on VC 6.0 and VC 7.0 environments by inputting the data from the code. It is designed to be able to easily plot data generated in a project without the need of any external software. PGL was originally based on the OpenGL to raster graphics, but revised versions uses GDI+ - revised version of Graphics Device Interface- for that purpose, like the one used in this thesis.

II.4.1. Parameters Used in the Compared Algorithms

There are three fuzzification parameters, which are fd, fe and $fdenom$, used in fuzzy BPNN and fuzzy RBF. As seen in equation (0.29), the output membership function is directly proportional with denominational fuzzification parameter (fd), and inversely proportional with the exponential fuzzification parameter (fe). When the fd is decreased, the output layer values of fuzzy neural network decreases, so does the mean square error. However, this decrease does not mean an improvement in the algorithm. In addition, when the fe is decreased, the output values of fuzzy neural network increases, so does the mean square error. Nonetheless, this increase does not show poor performance of the algorithm. The results are the same as they were but only a kind of scaling is done. For this reason, fd and fe parameters are kept constant throughout the entire implementation. The value for *fd* was chosen as 5 and the value of *fe* was chosen as 1. The other fuzzification parameter *fdenom* gives the extent of fuzzification. As *fdenom* increases, the overlapping structure of the membership functions increases. The *fdenom* was chosen as 0.8.

The threshold *Th* is used for selecting fuzzy inputs for rough set indiscernibility matrix formation. When the *Th* decreases, the number of objects included into the indiscernibility matrix increases. So the number of hidden nodes estimated changes. The *Th* is chosen as 0.8 to be consistent with [22].

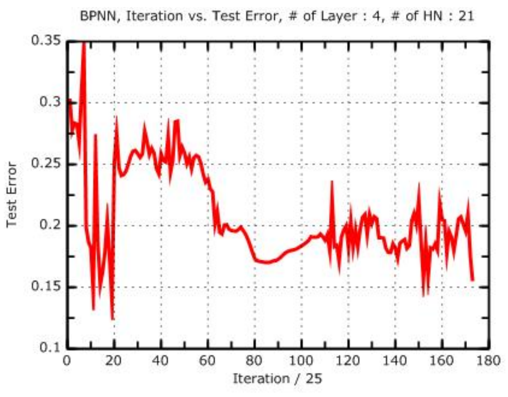
The neural network parameter "learning rate" was given as 0.5 at the beginning. Afterwards, it was continually decreased at each iteration. The momentum value was given 0.5 to keep the direction of search.

II.4.2. Vowel Input Set

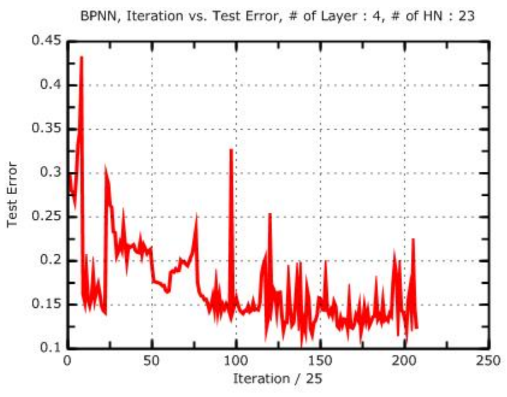
All the algorithms considered are executed with "vowel1" data. The other two "vowel" data were used for fuzzy BPNN, fuzzy rough BPNN, fuzzy RBF and fuzzy rough RBF. The test error graphics of the results were shown near the optimum runs.

II.4.2.1. Back-Propagation Neural Network (BPNN) Algorithm

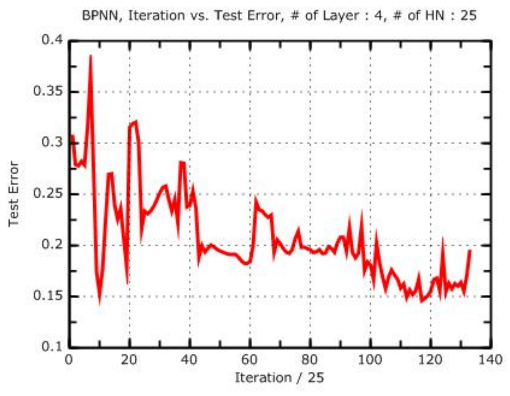
The "vowel1" data was run with 3, 4 and 5 number of layers. Theoretically, 5-layers are enough for modeling any type of data. The networks with 3 and 4 layers were run to find out whether less number of layers are enough to model the data – "vowel1". All these results were tried for wider ranges of number of hidden nodes (2 to 40) and iterations (25 to 32000), but only the results near the optimum ones are included.



(a)



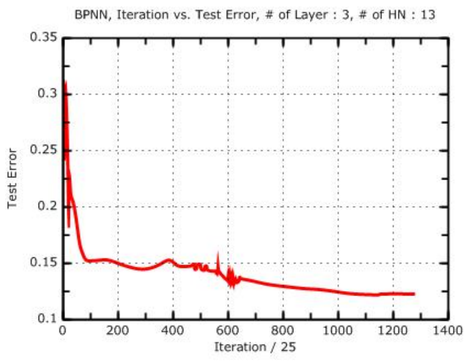
(b)



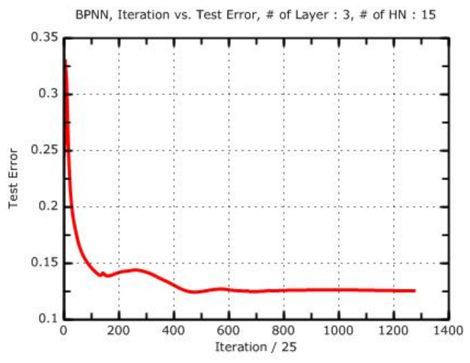
(c)

Figure 15 Mean square test error graphics of "vowel1" data for 4-layered BPNN with different numbers of hidden nodes **(a)** 21 Hidden Nodes **(b)** 23 Hidden Nodes **(c)** 25 Hidden Nodes

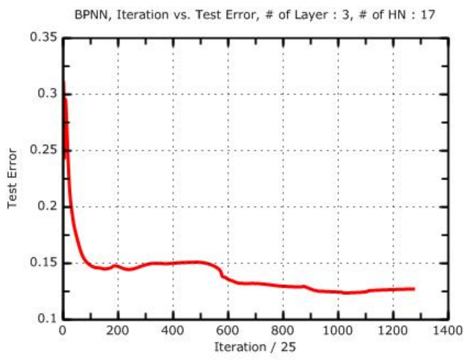
As seen in Figure 16, the best result for 3-layered BPNN was reached with 17 hidden nodes at the iteration 32000. For the 4-layered BPNN, Figure 15 shows that the minimum error is produced near the 5500 iterations for 23 hidden nodes in each layer. In Figure 17, the test results that belong to 5-layered BPNN can be seen. The best result for 5-layered BPNN is between 9 and 11 hidden nodes. The number of iterations needed is between 2500 and 3500.



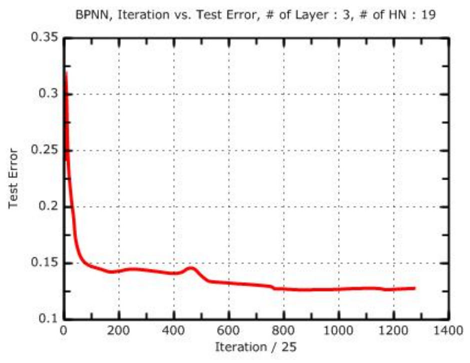
(a)



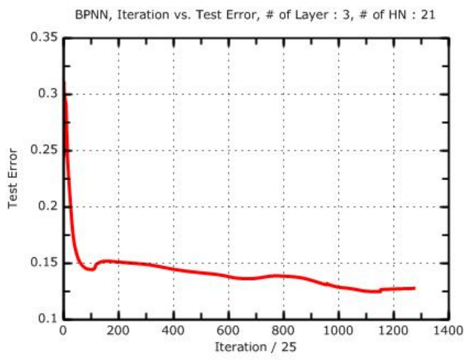
(b)



(c)

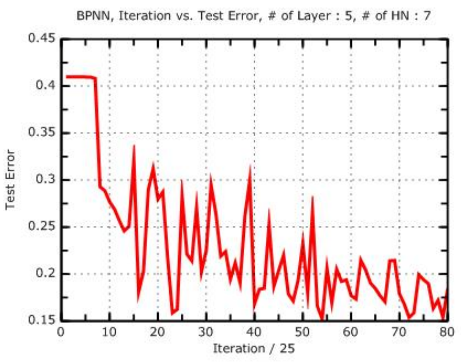


(d)

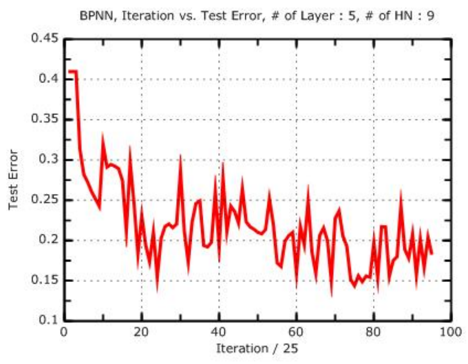


(e)

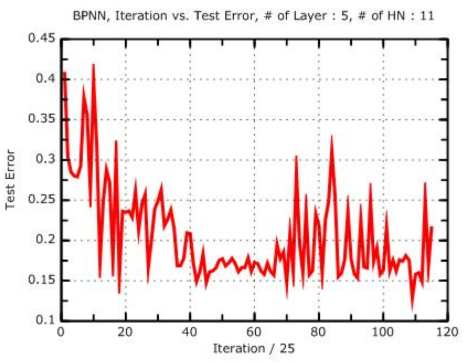
Figure 16 Mean square test error graphics of "vowel1" data for 3-layered BPNN with different numbers of hidden nodes **(a)** 13 Hidden Nodes **(b)** 15 Hidden Nodes **(c)** 17 Hidden Nodes **(d)** 19 Hidden Nodes **(e)** 21 Hidden Nodes



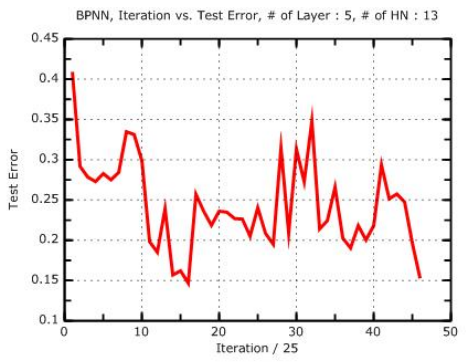
(a)



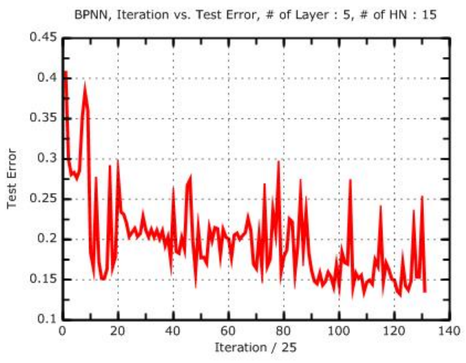
(b)



(c)



(d)



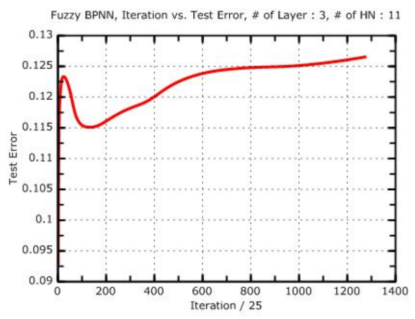
(e)

Figure 17 (a) Mean square test error graphics of "vowel1" data for 5-layered BPNN with different numbers of hidden nodes **(a)** 7 Hidden Nodes **(b)** 9 Hidden Nodes **(c)** 11 Hidden Nodes **(d)** 13 Hidden Nodes **(e)** 15 Hidden Nodes

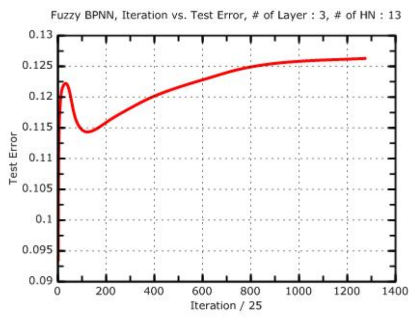
The optimum result for this algorithm was found as 5-layered network with 10 hidden nodes at about 3000 iterations. This result is consistent with the one stated in [7].

II.4.2.2. Fuzzy BPNN Algorithm

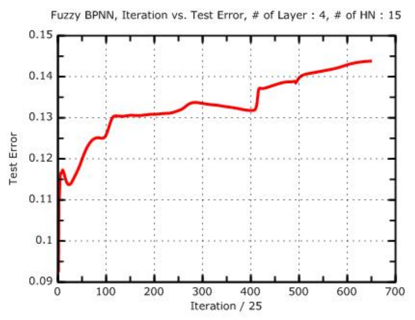
The fuzzy BPNN algorithm was run with all three versions of "vowel" data, which are "vowel1", "vowel2", and "vowel3". Although all the results were taken for wider ranges of number of hidden nodes, only the optimal number of hidden nodes for each run and two graphics below optimal and two graphics above optimal were chosen.



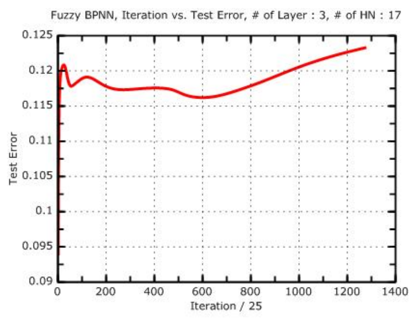
(a)



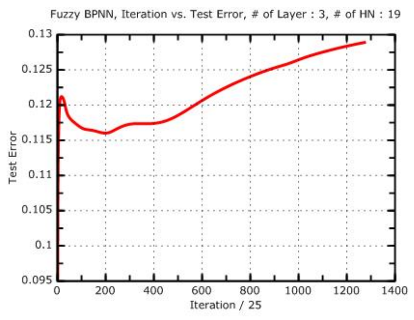
(b)



(c)

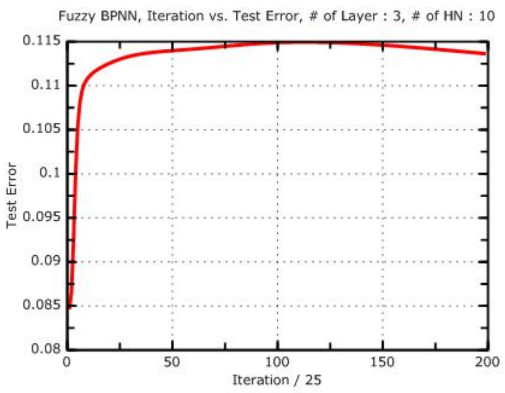


(d)

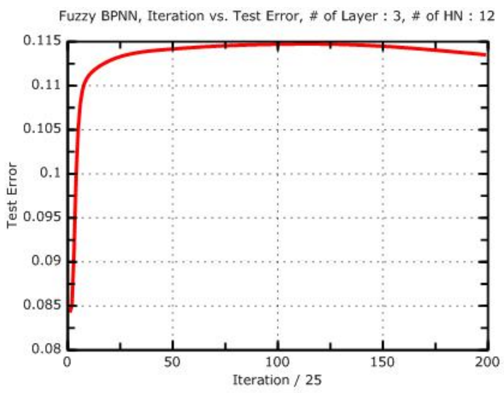


(e)

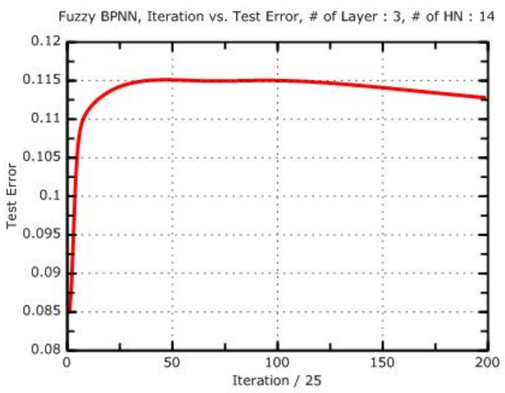
Figure 18 Mean square test error graphics of "vowel1" data for fuzzy BPNN with different numbers of hidden nodes **(a)** 11 Hidden Nodes **(b)** 13 Hidden Nodes **(c)** 15 Hidden Nodes **(d)** 17 Hidden Nodes **(e)** 19 Hidden Nodes



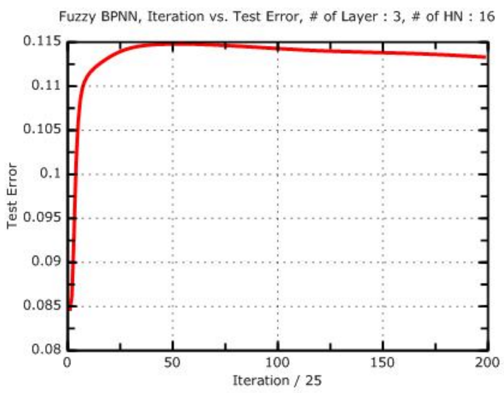
(a)



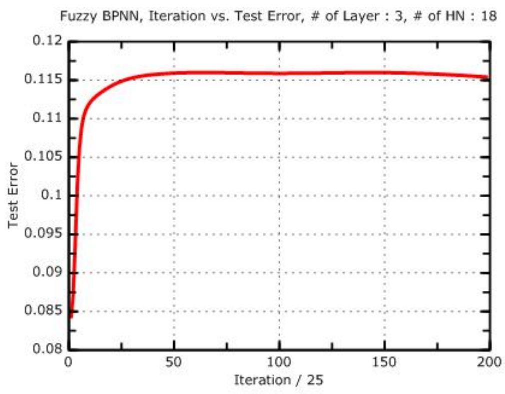
(b)



(c)

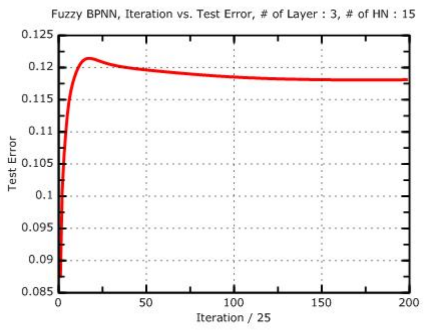


(d)

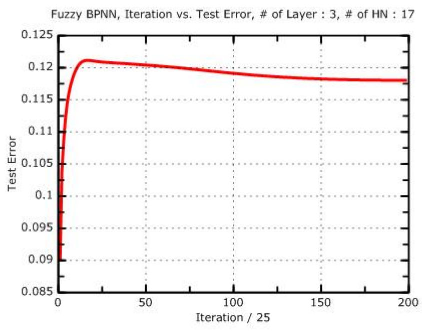


(e)

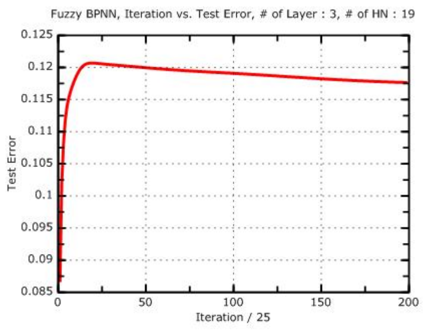
Figure 19 Mean square test error graphics of "vowel2" data for fuzzy BPNN with different numbers of hidden nodes **(a)** 10 Hidden Nodes **(b)** 12 Hidden Nodes **(c)** 14 Hidden Nodes **(d)** 16 Hidden Nodes **(e)** 18 Hidden Nodes



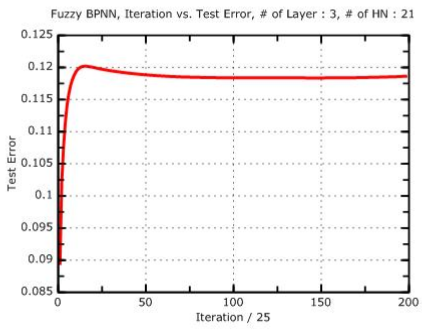
(a)



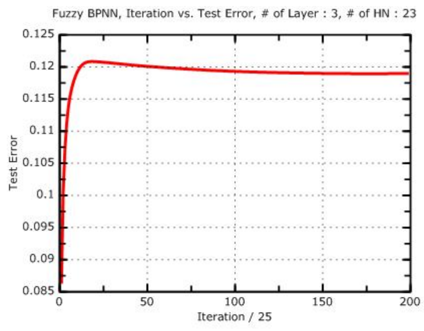
(b)



(c)



(d)



(d)

Figure 20 Mean square test error graphics of “vowel3” data for fuzzy BPNN with different numbers of hidden nodes **(a)** 15 Hidden Nodes **(b)** 17 Hidden Nodes **(c)** 19 Hidden Nodes **(d)** 21 Hidden Nodes **(e)** 23 Hidden Nodes

As seen in Figure 18, 15 hidden nodes are optimal for fuzzy BPNN with “vowel1” data. When “vowel2” data was given as input to fuzzy BPNN, the NN with 14 hidden nodes (Figure 19-(c)) did the best. In Figure 20, 19 hidden nodes gave the best result for fuzzy BPNN run with “vowel3” data. These results show that the optimal number of hidden nodes changes according to random distribution of data chosen as input. The results of fuzzy BPNN are better than the conventional MLP for “vowel” data.

II.4.2.3. Rough Fuzzy BPNN Algorithm

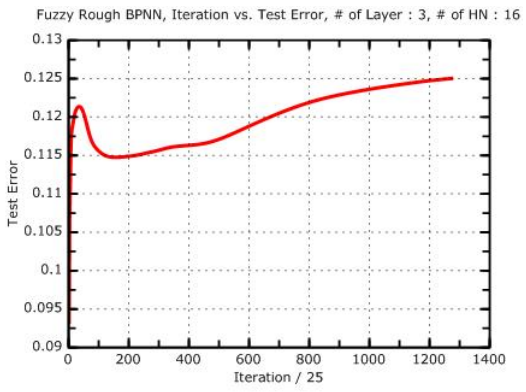


Figure 21 Mean square test error graphic of “vowel1” data for rough fuzzy BPNN with 16 hidden nodes

When the rough set dependency rules were generated for “vowel1” data, the estimated hidden node numbers for fuzzy BPNN was found as 16 as seen in Figure 21. The optimum hidden node number found was 15 for fuzzy BPNN with “vowel1” data (Figure 18). Although the estimation proposed by rough set dependency rules is not exactly the same as the optimum result, it is not exceptionally far away. The number of 14 hidden nodes is estimated for near optimum solution of “vowel2” data for fuzzy BPNN. This result coincides with the result found in Figure 19. The estimation of 19 hidden nodes was done for fuzzy BPNN with “vowel3” data. This result is same with the one found for fuzzy BPNN in Figure 20.

II.4.2.4. Radial Basis Function (RBF) Algorithm

The conventional RBF was run with wider range of number of hidden nodes for “vowel1” data. The results included here are the ones that are enough to show the tendency of the algorithm for different number of hidden nodes. In Figure 22, the best result was gained with 31 hidden nodes for RBF in “vowel1” data.

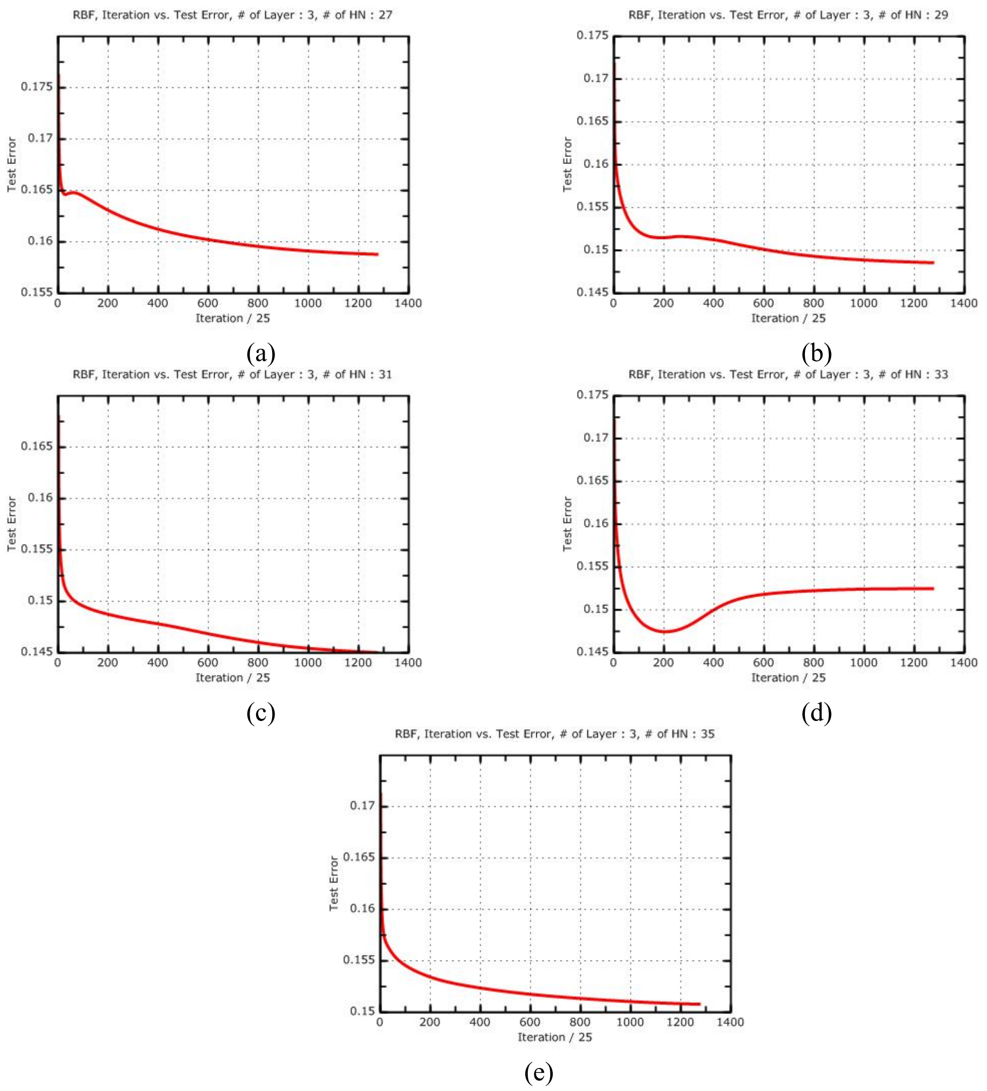
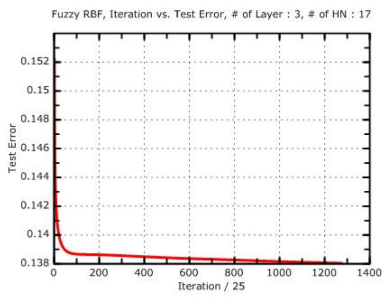


Figure 22 Mean square test error graphics of “vowel1” data for RBF with different numbers of hidden nodes **(a)** 27 Hidden Nodes **(b)** 29 Hidden Nodes **(c)** 31 Hidden Nodes **(d)** 33 Hidden Nodes **(e)** 35 Hidden Nodes

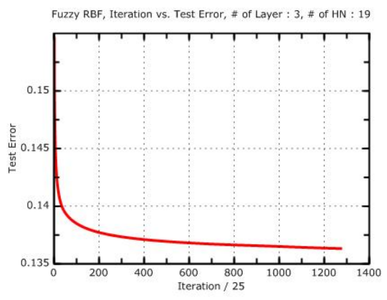
II.4.2.5. Fuzzy RBF

The fuzzy RBF was run with all versions of “vowel” data which are “vowel1”, “vowel2” and “vowel3”. The graphics enough to show the tendency of the algorithm and the optimum number of hidden nodes - if there exists - were included.

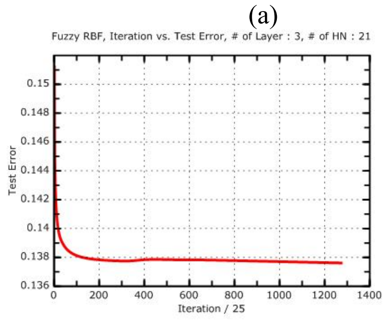
In Figure 23, the minimum error is reached with 23 hidden nodes for “vowel1” data executed for fuzzy RBF.



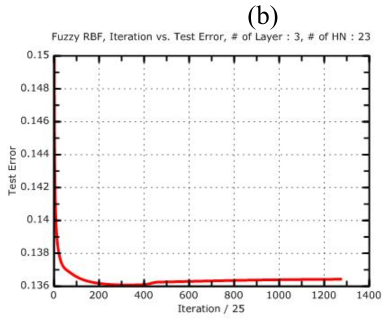
(a)



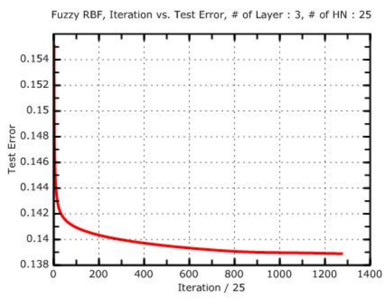
(b)



(c)

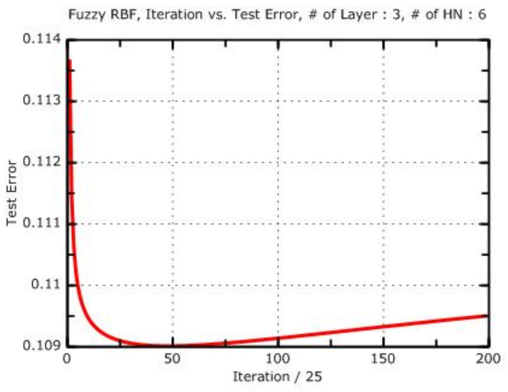


(d)

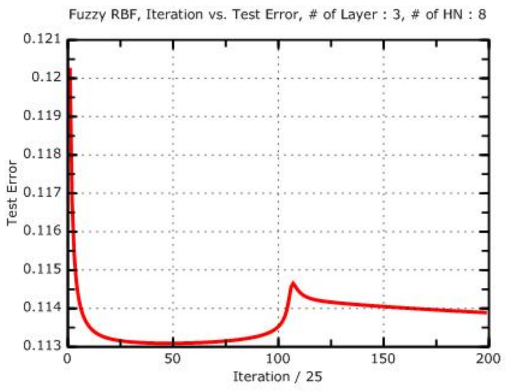


(e)

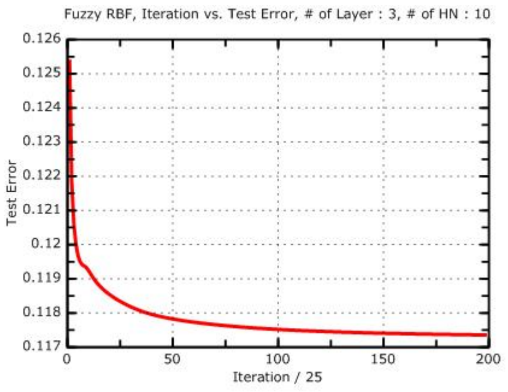
Figure 23 Mean square test error graphics of “vowel1” data for fuzzy RBF with different numbers of hidden nodes **(a)** 17 Hidden Nodes **(b)** 19 Hidden Nodes **(c)** 21 Hidden Nodes **(d)** 23 Hidden Nodes **(e)** 25 Hidden Nodes



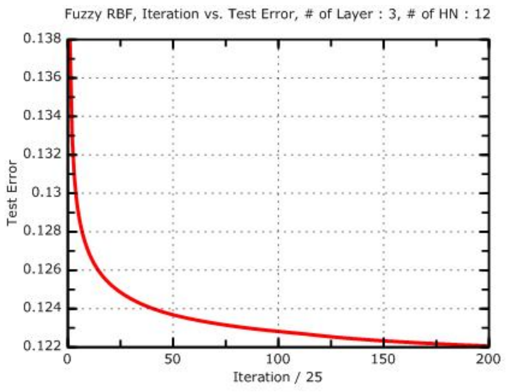
(a)



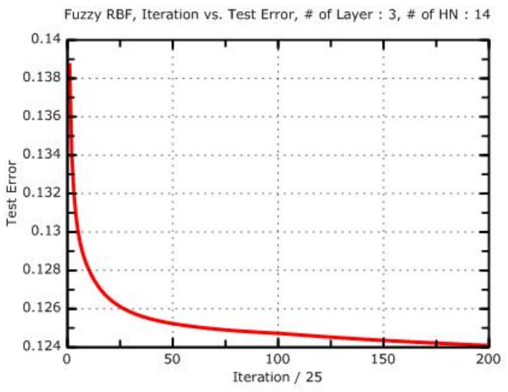
(b)



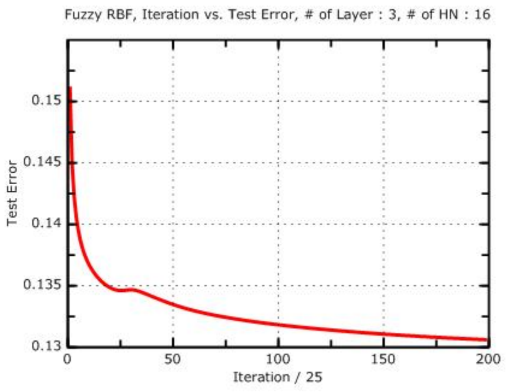
(c)



(d)

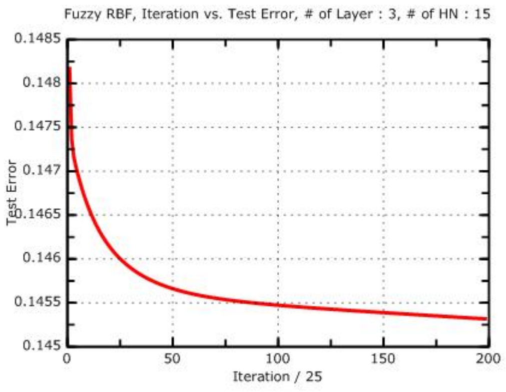


(e)

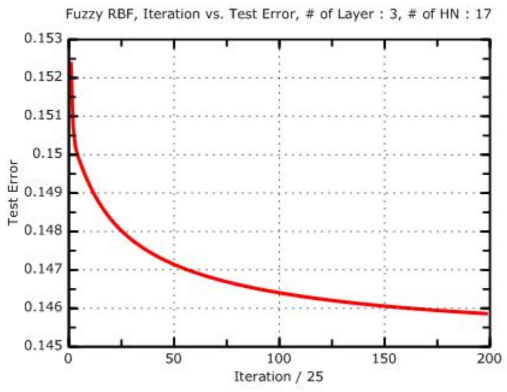


(f)

Figure 24 Mean square test error graphics of “vowel2” data for fuzzy RBF with different numbers of hidden nodes **(a)** 6 Hidden Nodes **(b)** 8 Hidden Nodes **(c)** 10 Hidden Nodes **(d)** 12 Hidden Nodes **(e)** 14 Hidden Nodes **(f)** 16 Hidden Nodes

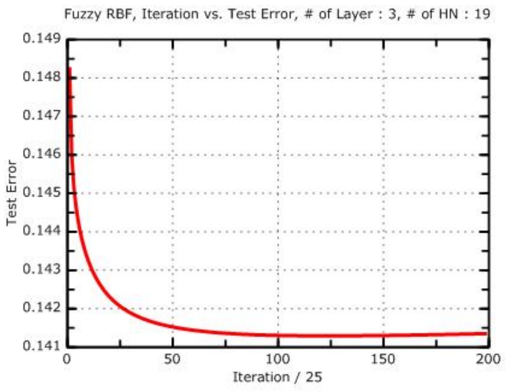


(a)

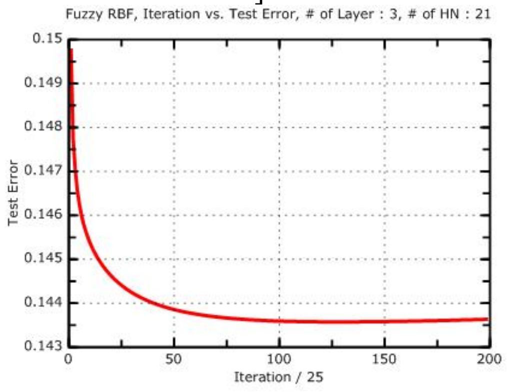


(b)

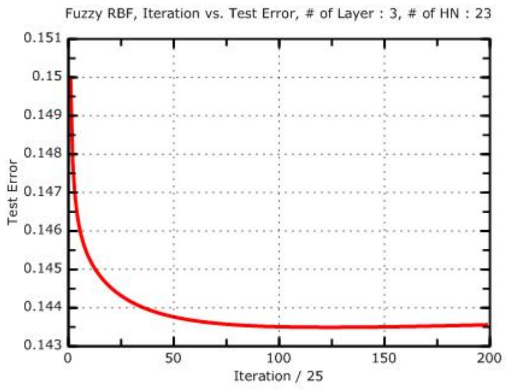
]



(c)



(d)



(e)

Figure 25 Mean square test error graphics of “vowel3” data for fuzzy RBF with different numbers of hidden nodes **(a)** 15 Hidden Nodes **(b)** 17 Hidden Nodes **(c)** 19 Hidden Nodes **(d)** 21 Hidden Nodes **(e)** 23 Hidden Nodes

In Figure 24, the test error of fuzzy RBF continually decreases as the number of hidden node decreases. Therefore, a conclusion on the optimum number of hidden nodes cannot be stated for fuzzy RBF which was executed with “vowel2” data.

In Figure 25, the optimum number of nodes is 19 for fuzzy RBF that has “vowel3” as input data. The results of the fuzzy RBF are better than that of the conventional RBF.

II.4.2.6. Rough Fuzzy RBF Algorithm

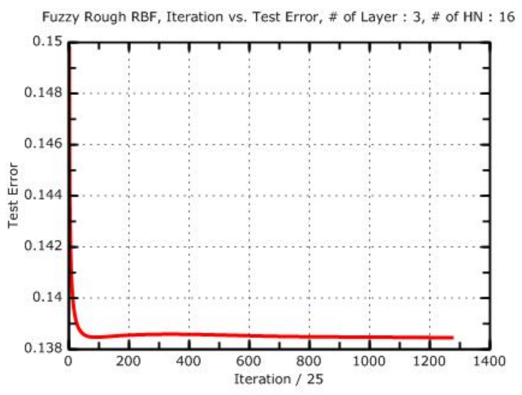


Figure 26 Mean square test error graphic of “vowel1” data for rough fuzzy RBF with 16 hidden nodes

According to rough set dependency rule generation, near optimum number of hidden nodes is estimated as 19 for “vowel3” data. This coincides with the result found in Figure 25. However, the optimal hidden node number is found as 16 for “vowel1” data (Figure 26), and this contradicts with the result found in Figure 23. The estimated and the actual number for hidden nodes also differ for “vowel2” data. The estimated number of hidden nodes is 14 whereas the same result cannot be observed in the actual results. (see Figure 24)

II.4.3. PAT1 Input Set

In conventional BPNN and RBF, only the “pat1_1” data is used. In fuzzy BPNN, fuzzy RBF, rough fuzzy BPNN and rough fuzzy RBF, all “pat1” data versions are used. (“pat1_1”, “pat1_2”, and “pat1_3”).

II.4.3.1. BPNN Algorithm

The conventional MLP algorithm was run for 3, 4, and 5 layers for “pat1_1” data. The aim was to decide on the optimum number of hidden nodes for the optimum number of layers.

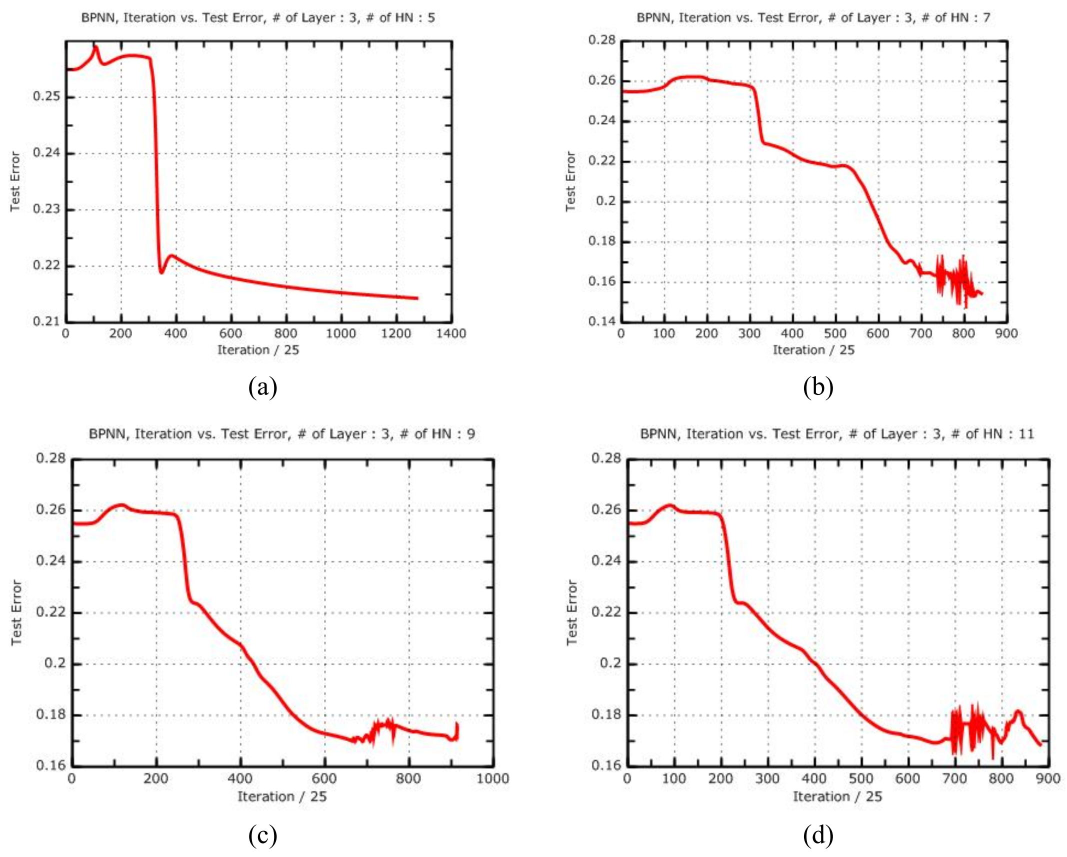
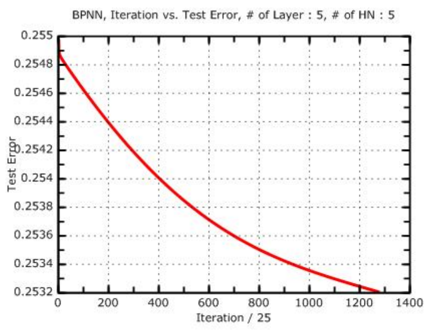
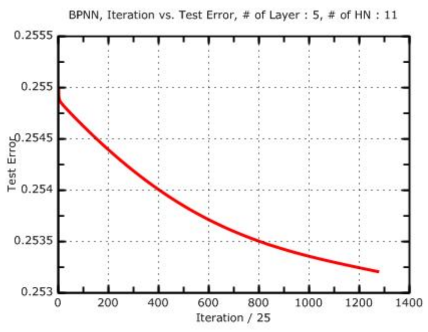


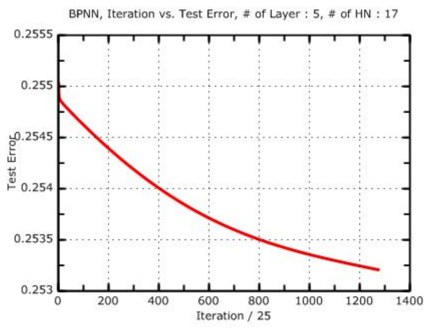
Figure 27 Mean square test error graphics of “pat1_1” data for 3-layered BPNN with different number of hidden nodes **(a)** 5 Hidden Nodes **(b)** 7 Hidden Nodes **(c)** 9 Hidden Nodes **(d)** 11 Hidden Nodes



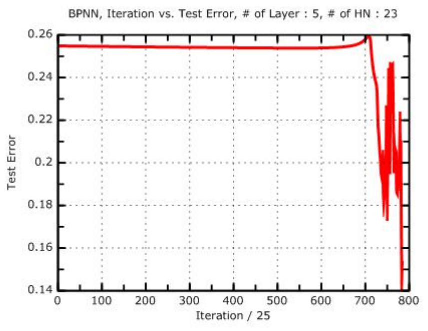
(a)



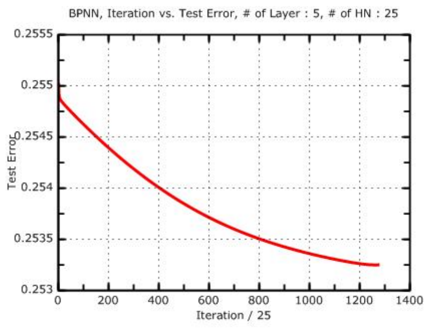
(b)



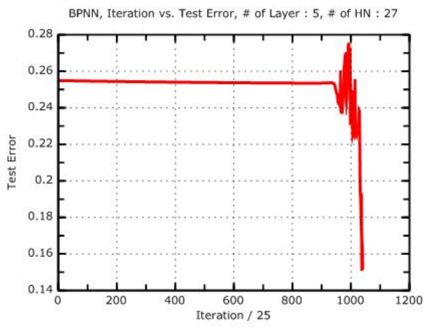
(c)



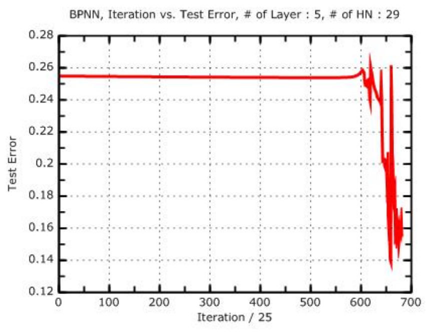
(d)



(e)



(f)



(g)

Figure 28 Mean square test error graphics of "pat1_1" data for 5-layered BPNN with different number of hidden nodes **(a)** 5 Hidden Nodes **(b)** 11 Hidden Nodes **(c)** 17 Hidden Nodes **(d)** 23 Hidden Nodes **(e)** 25 Hidden Nodes **(f)** 27 Hidden Nodes **(g)** 29 Hidden Nodes

The best result was achieved with 7 hidden nodes for 3-layered BPNN with “pat1_1” data as seen in Figure 27. Optimum number of hidden nodes is 5 at each layer for 4-layered BPNN as seen in Figure 29. The results of 4-layered BPNN were better than those of the 3-layered BPNN.

5-layered BPNN for “pat1_1” data has an unstable structure as seen in Figure 28. In addition, its generalization capability is not as good as that of a 4-layered BPNN. The cause for these results is that the structure of “pat1_1” data is much simpler than the “vowel1” data.

The optimum number of layers is 4 and the optimum number of hidden nodes is 5 for the conventional MLP with “pat1_1” data.

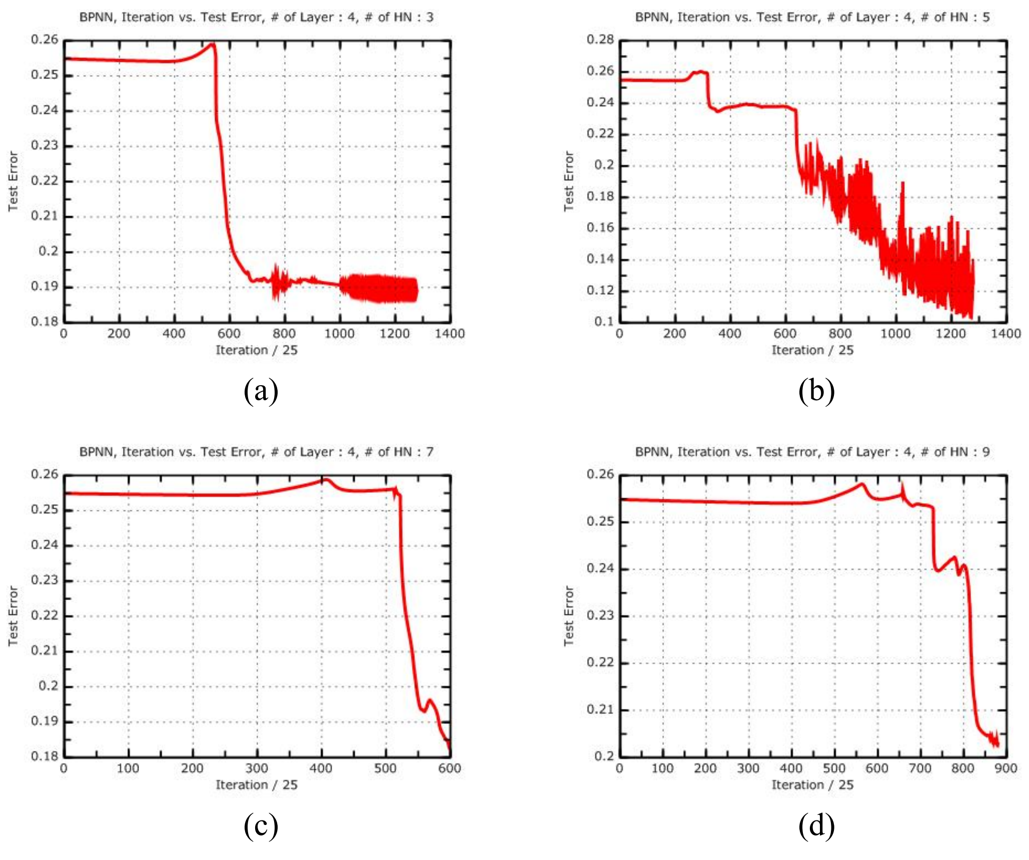
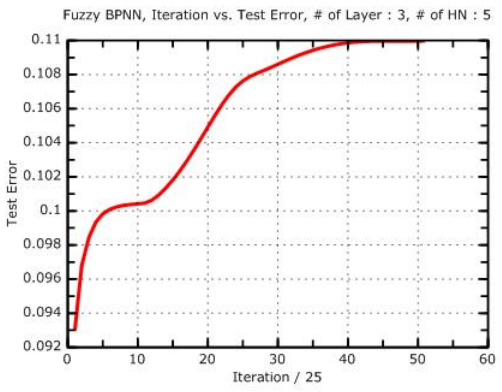


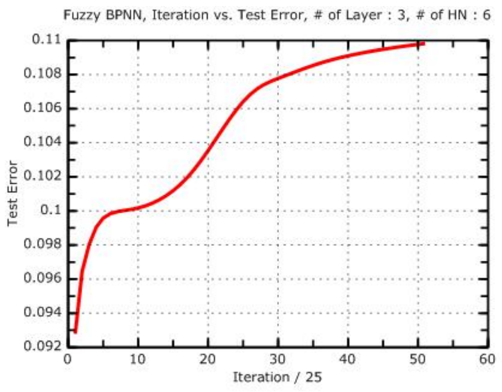
Figure 29 Mean square test error graphics of “pat1_1” data for 4-layered BPNN with different number of hidden nodes **(a)** 3 Hidden Nodes **(b)** 5 Hidden Nodes **(c)** 7 Hidden Nodes **(d)** 9 Hidden Nodes

II.4.3.2. Fuzzy BPNN Algorithm

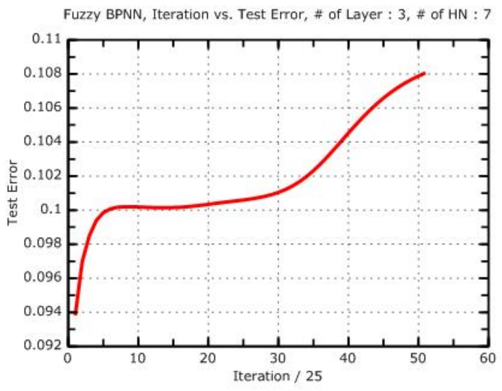
The fuzzy BPNN algorithm was run for “pat1_1”, “pat1_2”, and “pat1_3” data. The optimum number of hidden nodes was explored for each run. The graphics below were included to show the tendency of the algorithm near optimum number of hidden nodes.



(a)

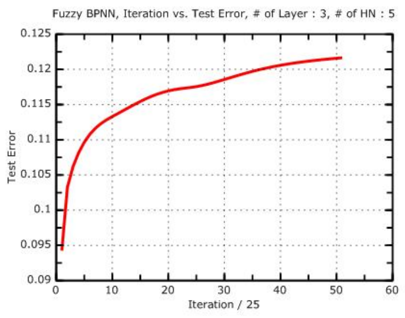


(b)

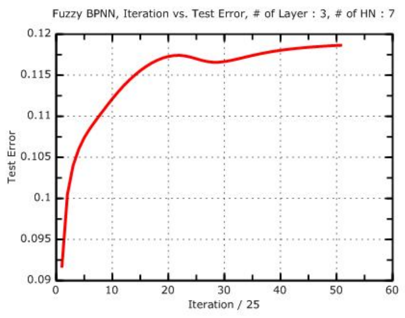


(c)

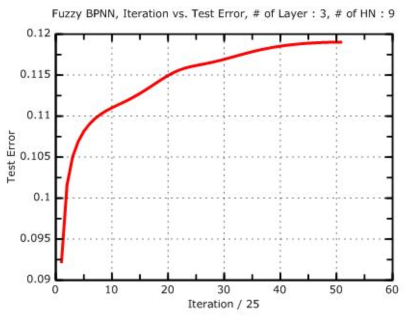
Figure 30 Mean square test error graphics of “pat1_1” data for fuzzy BPNN with different number of hidden nodes **(a)** 5 Hidden Nodes **(b)** 6 Hidden Nodes **(c)** 7 Hidden Nodes



(a)

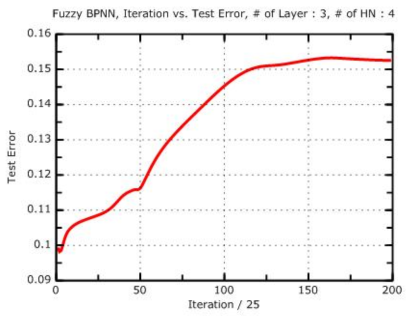


(b)

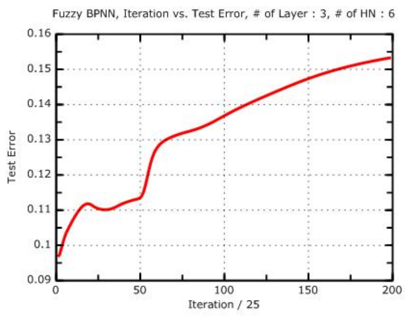


(c)

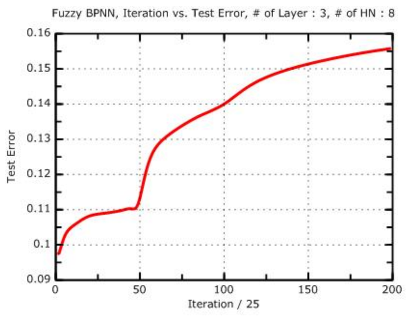
Figure 31 Mean square test error graphics of "pat1_2" data for fuzzy BPNN with different number of hidden nodes **(a)** 5 Hidden Nodes **(b)** 7 Hidden Nodes **(c)** 9 Hidden Nodes



(a)



(b)



(c)

Figure 32 Mean square test error graphics of "pat1_3" data for fuzzy BPNN with different number of hidden nodes **(a)** 4 Hidden Nodes **(b)** 6 Hidden Nodes **(c)** 8 Hidden Nodes

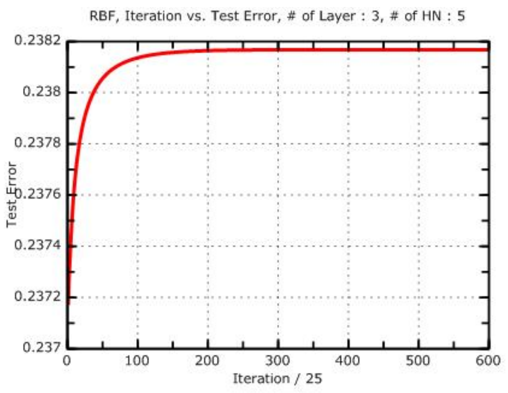
In Figure 30, the minimum error is reached with 6 hidden nodes in fuzzy BPNN which was executed with “pat1_1” data. When the fuzzy BPNN has “pat1_1” as input data, 7 as the number of hidden nodes was the best result as can be seen in Figure 31. The best result is reached at 6 hidden nodes in fuzzy BPNN with “pat1_3” data, as seen in Figure 32. Though the “pat1_1”, “pat1_2”, and “pat1_3” train data are the random selection of the same data, the optimum number of hidden nodes differs. These results show that the optimum number of hidden nodes is dependent on the random distribution of data. When the overall performance is taken into consideration, the results of the fuzzy BPNN are better than the conventional MLP for “pat1_1”data.

II.4.3.3. Rough Fuzzy BPNN Algorithm

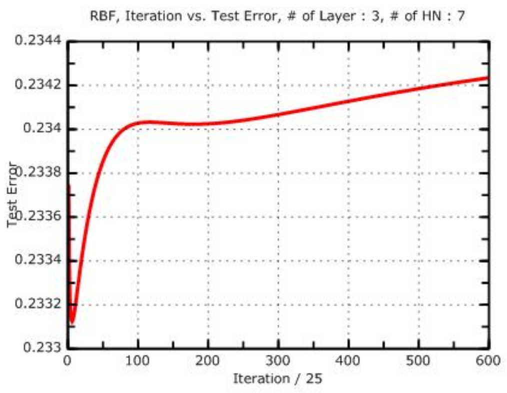
In rough fuzzy BPNN, the estimated hidden nodes numbers is 6 according to rough set dependency rules generated for “pat1_1” data. The actual optimum number of hidden nodes shown in Figure 30 was also 6 for “pat1_1” data. The rough set dependency rules generated for “pat1_2” data estimated the optimum number of hidden nodes as 7. This result is exactly the same as the one shown in Figure 31. The optimum number of hidden nodes for “pat1_3” was found as 6 and this coincides with the result in Figure 32.

II.4.3.4. RBF Algorithm

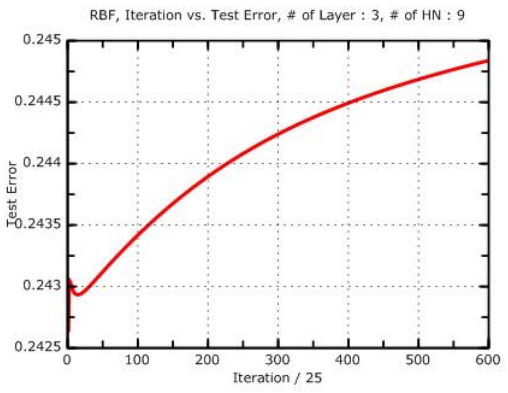
The conventional RBF was run with “pat1_1” to find the optimum number of hidden nodes and to see the performance of the algorithm. In conventional RBF, when the “pat1_1” data was given as input, best result is reached at 7 hidden nodes, as seen in Figure 33. The optimum results were reached less in less than 750 iterations. This indicates that the “pat1_1” data is simpler then the “vowel” data.



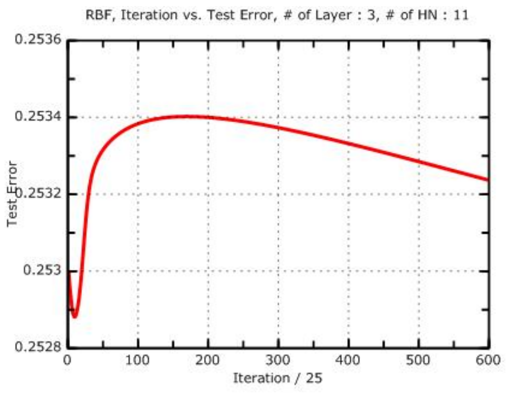
(a)



(b)



(c)

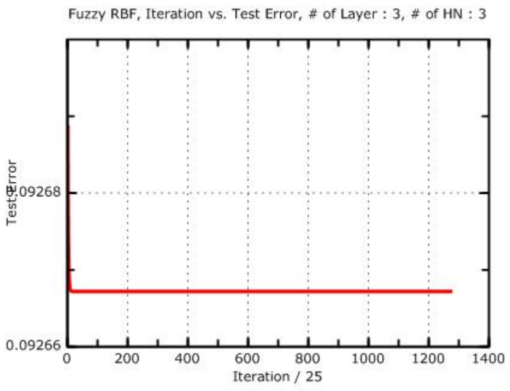


(d)

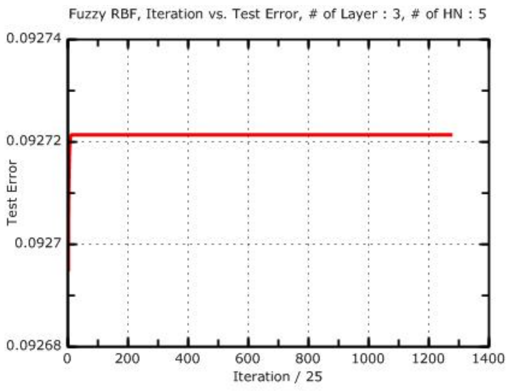
Figure 33 Mean square test error graphics of "pat1_1" data for RBF with different number of hidden nodes **(a)** 5 Hidden Nodes **(b)** 7 Hidden Nodes **(c)** 9 Hidden Nodes **(d)** 11 Hidden Nodes

II.4.3.5. Fuzzy RBF Algorithm

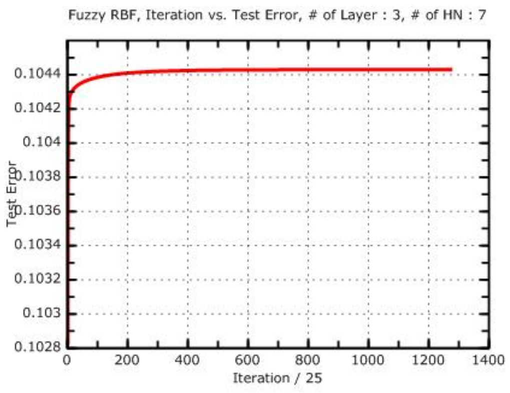
The fuzzy RBF was run with all versions of "pat1" data which are "pat1_1", "pat1_2", and "pat1_3". The tendency of the algorithm in each run is shown in Figure 34, Figure 35, and Figure 36.



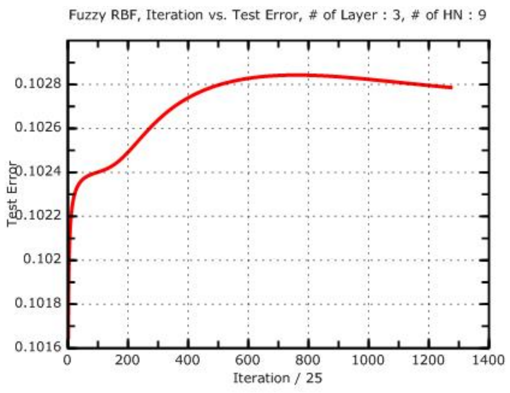
(a)



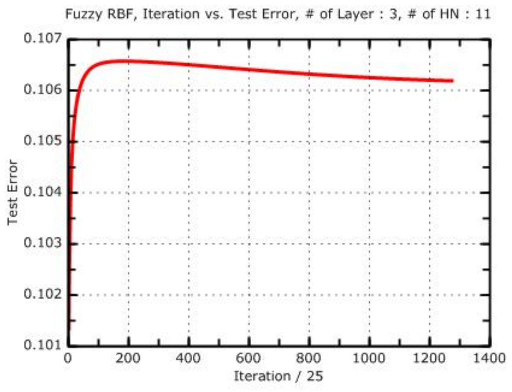
(b)



(c)

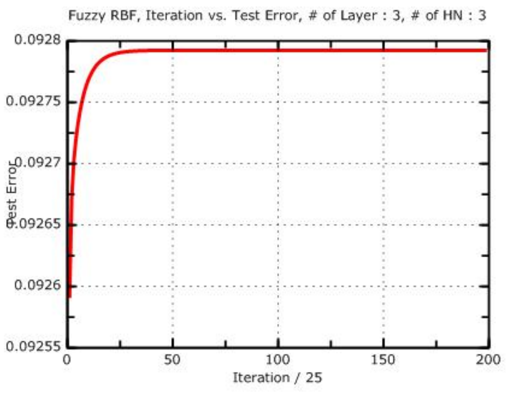


(d)

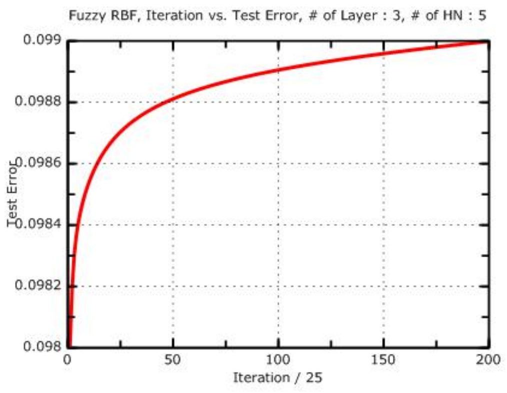


(e)

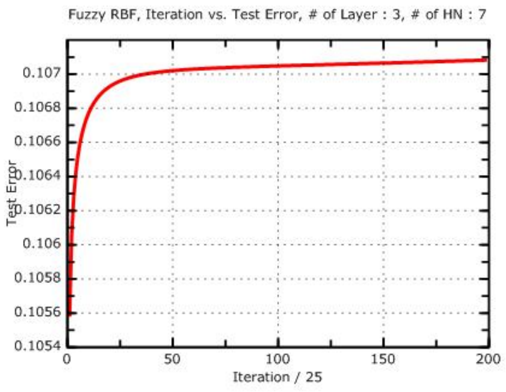
Figure 34 Mean square test error graphics of “pat1_1” data for fuzzy RBF with different number of hidden nodes **(a)** 3 Hidden Nodes **(b)** 5 Hidden Nodes **(c)** 7 Hidden Nodes **(d)** 9 Hidden Nodes **(e)** 11 Hidden Nodes



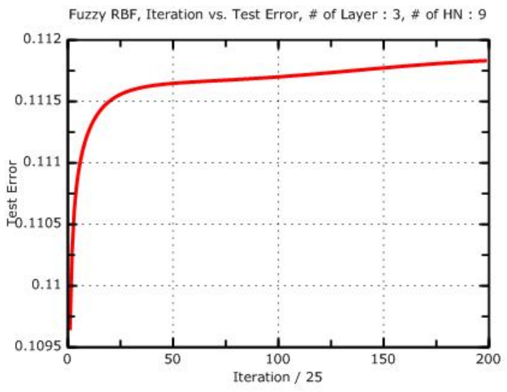
(a)



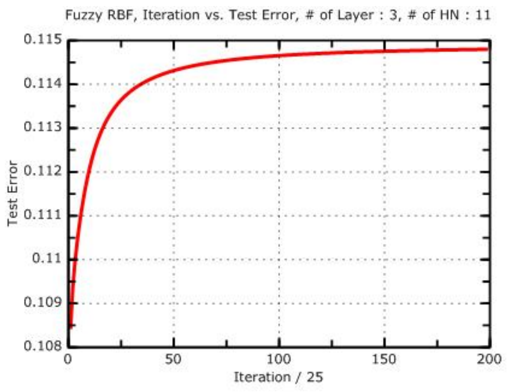
(b)



(c)

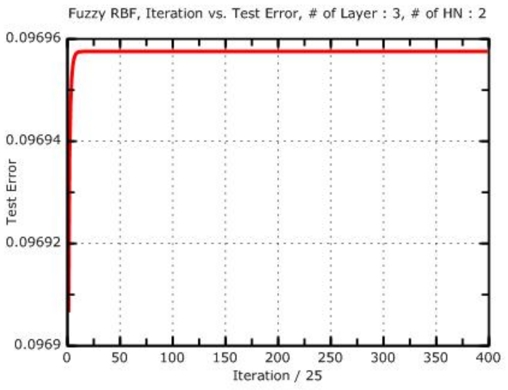


(d)

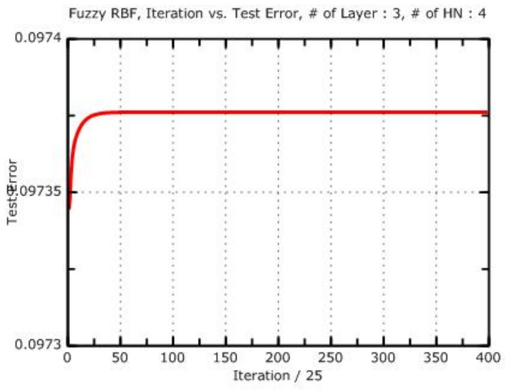


(e)

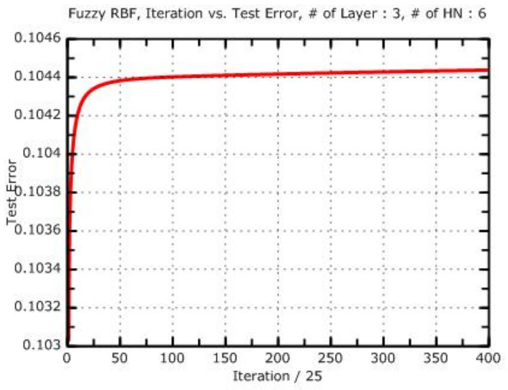
Figure 35 Mean square test error graphics of “pat1_2” data for fuzzy RBF with different number of hidden nodes **(a)** 3 Hidden Nodes **(b)** 5 Hidden Nodes **(c)** 7 Hidden Nodes **(d)** 9 Hidden Nodes **(e)** 11 Hidden Nodes



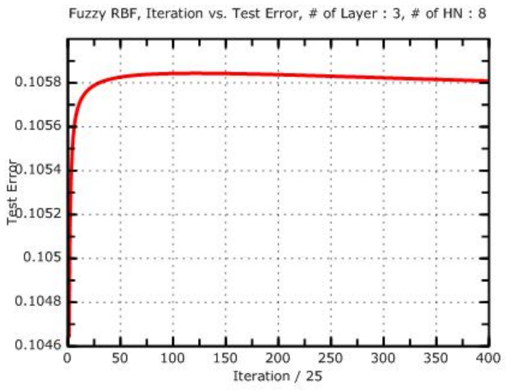
(a)



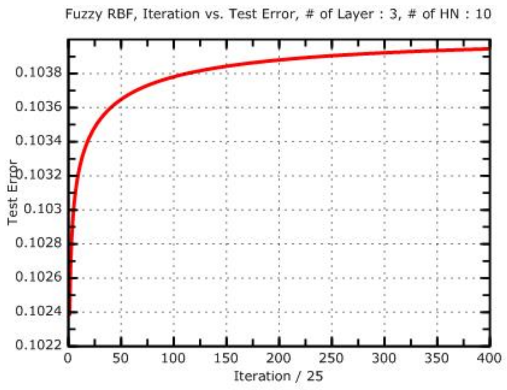
(b)



(c)



(d)



(e)

Figure 36 Mean square test error graphics of "pat1_3" data for fuzzy RBF with different number of hidden nodes **(a)** 2 Hidden Nodes **(b)** 4 Hidden Nodes **(c)** 6 Hidden Nodes **(d)** 8 Hidden Nodes **(e)** 10 Hidden Nodes

When the fuzzy RBF is executed for every version of "pat1" data, error decreases as the number of hidden nodes decreases, as seen in Figure 34, Figure 35, and Figure 36. Consequently, it is impossible to decide on the minimum number of nodes. Additionally, the optimum error value is reached with a very few iterations – less than 600. This implies the conclusion that "pat1" data is too simple for fuzzy RBF to learn.

II.4.3.6. Rough Fuzzy RBF Algorithm

Rough set dependency rules generated for "pat1_1" data estimates 6 for number of hidden nodes, as it is seen in Figure 37. This estimation is not true because there is no minimum number of hidden nodes for "pat1_1" data for fuzzy RBF. The estimated results for "pat1_2" and "pat1_3" are 7 and 6; however these are also wrong because the "pat1_2" and "pat1_3" does not have minimum number of hidden nodes for fuzzy RBF.

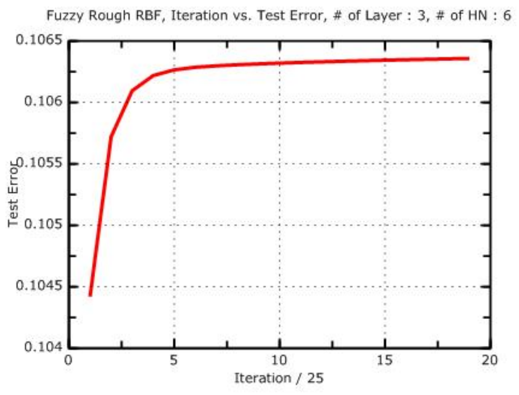


Figure 37 Mean square test error graphic of "pat1_1" data for rough fuzzy RBF with 6 hidden nodes

II.4.4. IRIS Input Set

The “iris” input set is run with fuzzy RBF to explore why the algorithm does not have a minimum number of hidden nodes for “pat1” data. The reason for choosing the “iris” data is that the data set is simpler than the “pat1” data. If the “pat1” data is too simple for fuzzy RBF to learn, then also the “iris” data has to be too simple for fuzzy RBF to learn.

II.4.4.1. Fuzzy RBF Algorithm

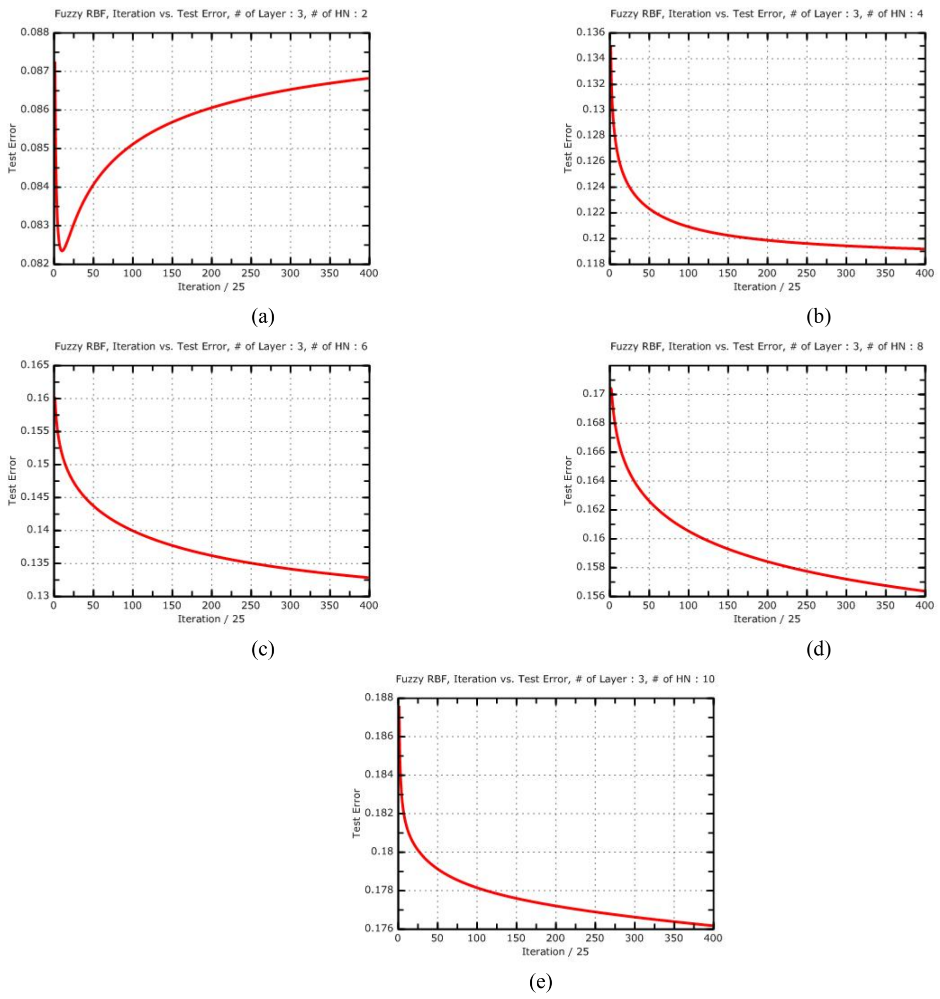


Figure 38 Mean square test error graphics of “iris” data for fuzzy RBF with different number of hidden nodes **(a)** 2 Hidden Nodes **(b)** 4 Hidden Nodes **(c)** 6 Hidden Nodes **(d)** 8 Hidden Nodes **(e)** 10 Hidden Nodes

The “iris” data was given as input to the fuzzy RBF. The aim is to decide whether the data given is too simple for fuzzy RBF to learn.

In Figure 38, as the number of hidden nodes decreases, the error continually decreases, as it does in “pat1” input data set. The minimum number of hidden nodes could not be calculated in “iris” data for fuzzy RBF. This results shows that the “iris” data is too simple for the fuzzy RBF to give acceptable results.

II.4.4.2. Rough Fuzzy RBF Algorithm

According to rough set dependency rules, the estimated number of hidden nodes is 6 for “iris” data (see Figure 39). It is concluded that the “iris” data is also too simple for fuzzy RBF. This result is not surprising as the “iris” data is the simplest data set of all data sets considered in this thesis.

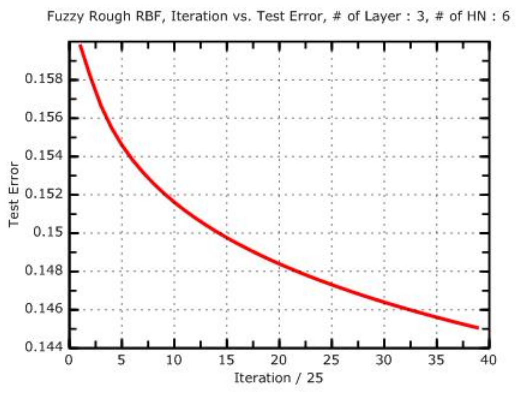


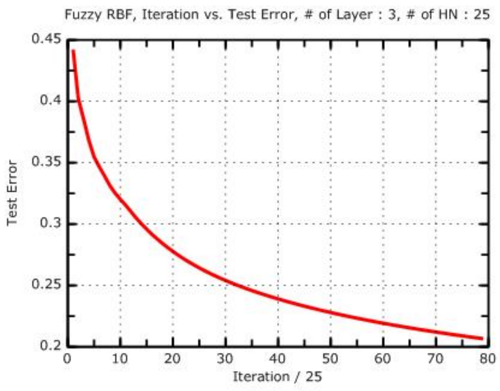
Figure 39 Mean square test error graphic of “iris” data for rough fuzzy RBF with 6 hidden nodes

II.4.5. EnglishVow Input Set

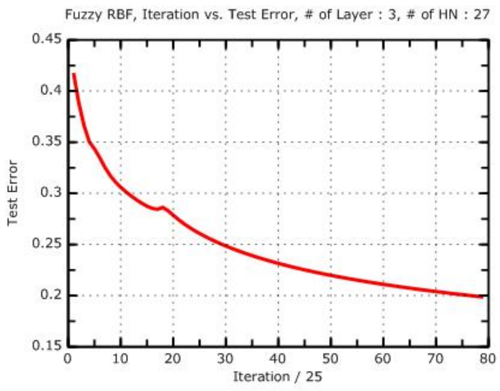
The “EnglishVow” data set is more complex than the “pat1” and the “iris” data. This data set is used to understand the characteristic of the rough fuzzy RBF.

II.4.5.1. Fuzzy RBF Algorithm

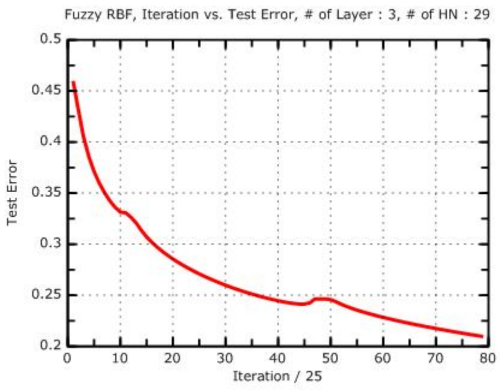
The “EnglishVow” data is given as input to fuzzy RBF with different number of hidden nodes. Only the results which are enough to show the tendency of the algorithm and the optimum number of hidden nodes are included. As seen in Figure 40, the optimum number of hidden nodes for fuzzy RBF is 27. The fuzzy RBF has meaningful results for “EnglishVow” data, because this data set is more complex than “pat1” and “iris” data sets.



(a)



(b)



(c)

Figure 40 Mean square test error graphics of “EnglishVow” data for fuzzy RBF with different number of hidden nodes **(a)** 25 Hidden Nodes **(b)** 27 Hidden Nodes **(c)** 29 Hidden Nodes

II.4.5.2. Rough Fuzzy RBF Algorithm

The number of hidden nodes estimated according to the rough set dependency rules generated for “EnglishVow” data is 27 (see Figure 41). This result is exactly the same as the actual optimum number of hidden nodes. It can be concluded that whenever a data is complex enough to be learned by fuzzy RBF, there is a probability that estimated number of hidden nodes by rough fuzzy RBF coincides with the optimum number of hidden nodes.

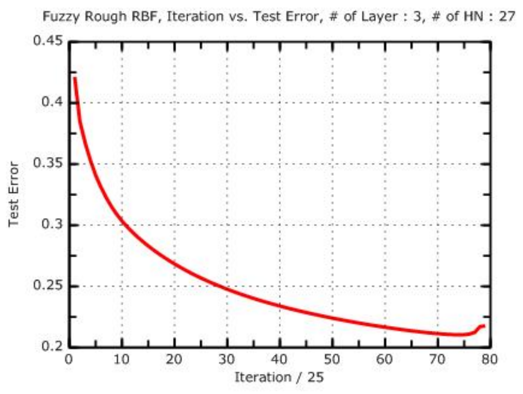


Figure 41 Mean square test error graphic of “EnglishVow” data for rough fuzzy RBF with 27 hidden nodes

CHAPTER IV

CONCLUSION

In this thesis conventional MLP, fuzzy BPNN, rough fuzzy BPNN and RBF fuzzy RBF, rough fuzzy RBF are compared. During these comparisons, four different types of data were used. The characteristics of rough fuzzy RBF were explored.

The test error of fuzzy BPNN is less than that of conventional MLP for both the “vowel” and the “pat1” data. This result is same as the ones in [7, 21]. The test error in fuzzy RBF is less than that of conventional RBF for the “vowel” data. This result is parallel with [15]. The “vowel” data has highly overlapping output regions. It has three input attributes and six output classes. On the other hand “pat1” data has also overlapping structure, but not as much as “vowel” data. The “pat1” data has two input attributes and three output classes. The “pat1” data is simpler than “vowel” data. As a result conventional methods did better for “pat1” than fuzzy RBF. The “iris” data is the simplest data set used in comparisons. The “iris” data has three output classes, one of which is linearly separable from other two. The overlapping structure of the data is so less for fuzzy RBF to be successful.

Rough fuzzy BPNN found the near optimal number of hidden nodes in each run of “vowel” and “pat1” data. This result was also shown in [22]. The initial weight values of fuzzy BPNN were generated by rough set dependency factors of dependency rules constructed.

A known study of rough fuzzy RBF was carried out for the first time in this thesis. It was explored whether the there can be a relation between the number of hidden nodes and dependency rules in fuzzy RBF. Because the results showed that “pat1” and “iris” data are not appropriate for the fuzzy RBF, number of hidden nodes estimated using rough sets can not be taken as the only base for the evaluation of rough fuzzy RBF. When the rough fuzzy RBF was run with three versions of the “vowel” data, estimated number of hidden nodes found by rough sets coincide the actual results for only one version of “vowel” data. In “vowel1” data the estimated number is 16 whereas the actual number is 23. The error in “vowel2” decreases, as the number of nodes decreases.

When *fdenom* and *Th* values were changed in rough fuzzy RBF, in some cases estimated number of hidden nodes coincide with the optimum number of hidden nodes. However, as these cases do not show common properties, a generalization could not be made. As a result, in rough fuzzy RBF proposed, a direct relationship between dependency rules and optimum number of hidden nodes could not be made. Because the result of RBF is highly dependent on the initial clusters given and the sequence of data introduced to the network, defining optimum number of hidden nodes for RBF has to be dependent on what RBF is dependent.

REFERENCES

[1] L. A. Zadeh, "Fuzzy sets." *Information and Control*, Vol. 8, no. 3, pp. 338-353, June 1965.

[2] Z. Pawlak, "Rough sets." *International Journal of Computer and Information Science*, Vol. 11, pp. 341-356, 1982.

[3] D. Rutkowska, "Neuro-Fuzzy Architectures and Hybrid Learning", *Physica Verlag Heidelberg New York*, 2002.

[4] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes", *IEEE Transactions on System, Man and Cybernetics,* Vol. 3, no. 1, pp. 28-44, 1973.

[5] J. Yen, R. Langari, "Fuzzy logic: intelligence, control and information", *Prentice-Hall*, 1999.

[6] J. Komorowski, L. Polkowski, A. Skowron, "Rough Sets: A Tutorial", *Lecture Notes of the 11th European Summer School in Logic, Language and Information (ESSLLI)*, 1999.

[7] S. K. Pal, S. Mitra, "Multilayer Perceptron, Fuzzy Sets, and Classification", *IEEE Transactions on Neural Networks,* Vol. 3, no. 5, pp. 683-697, 1992.

[8] S. K. Pal, S. Mitra, P. Mitra, "Rough Fuzzy MLP: Modular Evolution, Rule Generation and Evaluation", *IEEE Transactions on Knowledge and Data Engineering,* Vol. 15, pp. 14-25, 2003.

[9] M. Sarkar, B. Yegnanarayana, "Rough-Fuzzy Membership Functions", *Proc. IEEE International Conference on Fuzzy Systems,* Alaska, USA, pp. 796-801, May 4-9, 1998.

[10] M. Sarkar and B. Yegnanarayana, "A review on merging some recent techniques with artificial neural networks**",** *Proc. IEEE International Conference on Systems, Man and Cybernetics,* California, USA, October 11-14, 1998.

[11] S. Mitra and Y. Hayashi, "Neuro-fuzzy rule generation: Survey in soft computing framework", *IEEE Transactions on Neural Networks*, Vol. 11, pp. 748-768, 2000.

[12] T. P. Hong, L. H. Tseng, B. C. Chien, “Learning fuzzy rules from incomplete Quantitative Data by Rough Sets”, *IEEE World Congress on Computational Intelligence,* Hawaii, USA, pp. 1438-1443, May 12-17, 2002.

[13] M. Sarkar, B. Yegnanarayana, “Fuzzy-Rough Neural Networks for Vowel Classification”, *IEEE Transactions on System, Man, and Cybernetics,* Vol. 5, pp. 4160-4165, 1998.

[14] T. L. Seng, M. Khalid, R. Yusof, “Tuning of a Neuro-Fuzzy Controller by Genetic Algorithms with an application to a Coupled-Tank Liquid-Level Control System”, *International Journal of Engineering Applications on Artificial Intelligence, Pergamon Press,* Vol. 11, pp. 517-529, 1998.

[15] S. Mitra and J. Basak, “FRBF: A fuzzy radial basis function network”, *Neural Computing and Applications*, Vol. 10, pp. 244-252, 2001.

[16] L. M. Fu, “Knowledge-Based Connectionism for Revising Domain Theories”, *IEEE Transactions on System, Man, and Cybernetics,* Vol. 3, pp. 173-182, 1993.

[17] R. Yasdi, “Combining Rough Sets Learning- and Neural Network Learning-method to Deal with Uncertain and Imprecise Information”, *Neurocomputing, Elsevier Science,* Vol. 7, pp. 61-84, 1995.

[18] S. Mitra, S. K. Pal and P. Mitra, “Data mining in soft computing framework: A survey”, *IEEE Transactions on Neural Networks*, Vol. 13, pp. 3-14, 2002.

[19] M. Sarkar, “Fuzzy-rough nearest neighbors algorithm”, *Proc. IEEE Internat. Conf. on Systems, Man and Cybernetics,* Tennessee, USA, pp. 3556-3601, October 8-11, 2000.

[20] W. Y. Liu, C. J. Xiao, B. W. Wang, Y. Shi, S. F. Fang, “Study on Combining Subtractive Clustering with Fuzzy C-means Clustering”, *Proc. of the Second International Conference on Machine Learning and Cybernetics,* 2-5 November 2003.

[21] S. Mitra and S. K. Pal, “Fuzzy multi-layer perceptron, inferencing and rule generation”, *IEEE Transactions on Neural Networks*, Vol. 6, pp. 51-63, 1995.

[22] M. Banerjee, S. Mitra and S. K. Pal, “Rough fuzzy MLP: Knowledge encoding and classification”, *IEEE Transactions on Neural Networks*, Vol. 9, pp. 1203-1216, 1998.

[23] C.L. Blake, C.J. Merz,. “UCI Repository of machine learning databases”, *[http://www.ics.uci.edu/~mlearn/MLRepository.html],* Irvine, CA: University of California, Department of Information and Computer Science, 1998. Last reached on 18 August 2004.

[24] L. Fu, “Neural Networks in Computer Intelligence”, *The McGraw-Hill Companies, New York,* January 1994.

[25] D. Dubois, H. Prade, “Putting rough sets and fuzzy sets together”, *Intelligent Decision Support, Handbook of Applications and advances of the Rough Set Theory, R. Slowinski (Ed.), Kluwer Academic Publishers, Boston,* 204-232, 1992.

[26] Z. Pawlak, “Conflict analysis”, *Proceedings of the Fifth European Congress on Intelligent Techniques and Soft Computing (EUFIT'97)*, pp.1589-1591, Verlag Mainz, Aachen, 1997.