RESEARCH AND DEVELOPMENT TECHNICAL REPORT
CECOM-TR-98-4

A Fuzzy Hypercube Artificial Neural Network Classifier

Joseph A. Karakowski and Hai H. Phu

October 1998

CECOM
U.S. ARMY COMMUNICATIONS-ELECTRONICS COMMAND
RESEARCH, DEVELOPMENT AND ENGINEERING CENTER
FORT MONMOUTH, NEW JERSEY 07703-5000

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The citation of trade names and names of manufacturers in this report is not to be construed as official Government endorsement or approval of commercial products or services referenced herein.

REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1998	Technical Report

4. TITLE AND SUBTITLE
A FUZZY HYPERCUBE ARTIFICIAL NEURAL NETWORK CLASSIFIER

5. FUNDING NUMBERS

6. AUTHOR(S)
Joseph A. Karakowski and Hai H. Phu

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES)
US Army Communications-Electronics Command (CECOM)
Research, Development and Engineering Center (RDEC)
Intelligence and Information Warfare Directorate (I2WD)
ATTN: AMSEL-RD-IW-TP
Fort Monmouth, NJ 07703-5211

8. PERFORMING ORGANIZATION REPORT NUMBER
CECOM-TR-98-4

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12 b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

An Artificial Neural Network classifier, based on fuzzy Min-Max, Adaptive Resonant Theory (ART), and fuzzy ART is described. The outputs of the classifier are fuzzy hypercubes representing functional categories of its input functions. A hypothesis and test paradigm compares input data and existing hypercube categories, and results in either network "resonance or dissonance," depending on the test outcome. A hypothesis is formed by two match functions: Degree of Inclusion, and Degree of Perfect Match. An overall hypothesis is chosen with the best Degree of Match. Tests are then performed to verify the hypothesis. The vigilance test measures the top down match between the hypothesized category and the input. The overall hypervolume test ensures that any category adjustments keep the total category hypercube volume within bounds. The fuzzy hypercube classifier was tested using two standard sets: Iris Flower and Wisconsin Diagnostic Breast Cancer. The network produced 88% and 76% correct classification, respectively. A speaker recognition system using a fuzzy hypercube classifier was also tested using the Switchboard and Greenflag databases. Test results are discussed.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Speaker Recognition; Fuzzy Logic; Signal Processing; Neural Nets; Neural Networks; Speech Recognition	47
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

LIST OF ACRONYMS

ANN	Artificial Neural Network
ART	Adaptive Resonance Theory
DOI	Degree of Inclusion
DPM	Degree of Perfect Match
FHANN	Fuzzy Hypercube Artificial Neural Network
FV	Feature Vector
HV	Hypercube Volume
LTM	Long Term Memory
NN	Neural Network
SFAM	Simplified Fuzzy ART Map
STM	Short Term Memory

SUMMARY

An Artificial Neural Network (ANN) classifier, based on fuzzy Min-Max [21], Adaptive Resonant Theory (ART) [2,5], and fuzzy ART [7,17] is described. The outputs of the classifier are fuzzy hypercubes representing functional categories of its input functions. A hypothesis and test paradigm compares input data and existing hypercube categories, and results in either network "resonance or dissonance," depending on the test outcome. A hypothesis is formed by two functions which measure the degree of match between input and each hypercube category. The hypothesis match functions are Degree of Inclusion, and Degree of Perfect Match. An overall hypothesis is chosen based on the best hypothesis, i.e., the hypothesis which has the greatest match. A test is then performed to verify the hypothesis. The test consists of both a vigilance, and overall hypervolume limit test. The vigilance test measures the top down match between the hypothesized category and the input. The overall hypervolume test ensures that any category adjustments keep the total category hypercube volume within bounds.

Category representation is extended beyond a unit hypercube as in [21] reflecting the interpretation as a degree of typicality, rather than relative [18] to allow for more "noisy" feature hypercubes. The network has 7 layers; input, transform, process, hypothesize, test, functional, and category. The network is described in a hybrid neuronal-functional approach.

The network was tested with standard classification test data for recogniton of speakers in an open set, text-independent environment. The standard sets were Iris and Wisconsin Diagnostic Breast Cancer (WDBC) [26]. The network produced 88% correct classification for WDBC and 76% for Iris.

A speaker recognition system [12,13,14] using a fuzzy hypercube classifier was tested using the Switchboard [22] and Greenflag [23] data bases. The fuzzy hypercube ANN, characterizing one speaker per category, produced an average of 6.29 correct and 0.29 incorrect categories out of a possible 8 total, with no prior training. The overall average correct classification produced by the network using a fixed set of signal features for 8, 12, and 16 speaker groups was 67%. See [15] for details.

1. INTRODUCTION

Classification is the process of labeling data as groups which are related by one or more common concepts. There are numerous approaches and definitions of classification in the literature based on statistical methods. See [8] for an introduction. Fuzzy methods in classification are developed in [1], along with categorization of classifier designs according to model, data, output, decision regions, algorithm, and architecture. In this report, we are concerned with designs which are based on the general hypothesize and test paradigm with data driven hypothesis formation [24]. Here the knowledge is decomposed hierarchically into "levels of analysis" [24].

The use of an Artifical Neural Network as a classifier, particularly for unsupervised learning, is based on the architectures of ART [2,3,4], fuzzy ART [6,17], and fuzzy min-max [21]. These all employ the general hypothesis and test paradigm, but are more neuronal "model-based" methods. In each the general processes of hypothesis generation are roughly maintained as a neural structure, and a testing mechanism is described as a somewhat complex feedforward/feedback functional control mechanism.

A brief introduction to the basic and fuzzy ART functions, architectures, standard terminology, and equations will be presented next as background. Familiarity with basic fuzzy sets, fuzzy set theory and clustering is assumed. See Chapter 2 in [1] for introductory papers in fuzzy set theory and clustering, and Dubois and Prade [9] for further details on general fuzzy set theory.

1.1 ART2

A typical ART2 [14] network is composed of two layers (or fields) of neurons which form the *Attentional Subsystem*. The first field is the *Feature Representation Field*, or F_1, which contains processing elements (PE) that form three intra-PE sublayers which are responsible for processing one element in the input pattern. The main function of the feature representation field is to enhance the current input pattern's salient features while suppressing noise.

The second layer in the attentional subsystem is called the *Category Representation Field*, or F_2, which represents one category (or class) that has been learned by the network. The connections from a particular F_2 neuron store a category *pattern*. ART2 utilizes an unsupervised learning technique which attempts to discover the distributions and centroids of the categories for the patterns it is presented. ART2 can utilize a "winner-take-all" classification strategy, such as MAXNET [19]. Having selected the winner, an *Orienting Subsystem* is activated to determine whether the proposed winning neuron's Long Term Memory (LTM) traces sufficiently resemble the Short Term Memory (STM) pattern to be considered a match. A matching threshold called the *Vigilance Parameter* determines how similar the input pattern must be to the exemplar to

be considered a match. If the degree of match computed by the orienting subsystem exceeds the vigilance parameter, a state of resonance is attained and the STM pattern at F_1 is merged onto the winning neuron's LTM traces. Otherwise, the orienting subsystem sends a reset signal to the winning neuron, and inhibits it from competing again for the current input pattern. This search process is repeated until either an F_2 neuron passes the vigilance test or all established F_2 neurons have failed the test. In the latter case, a new category is established in the next available F_2 neuron.

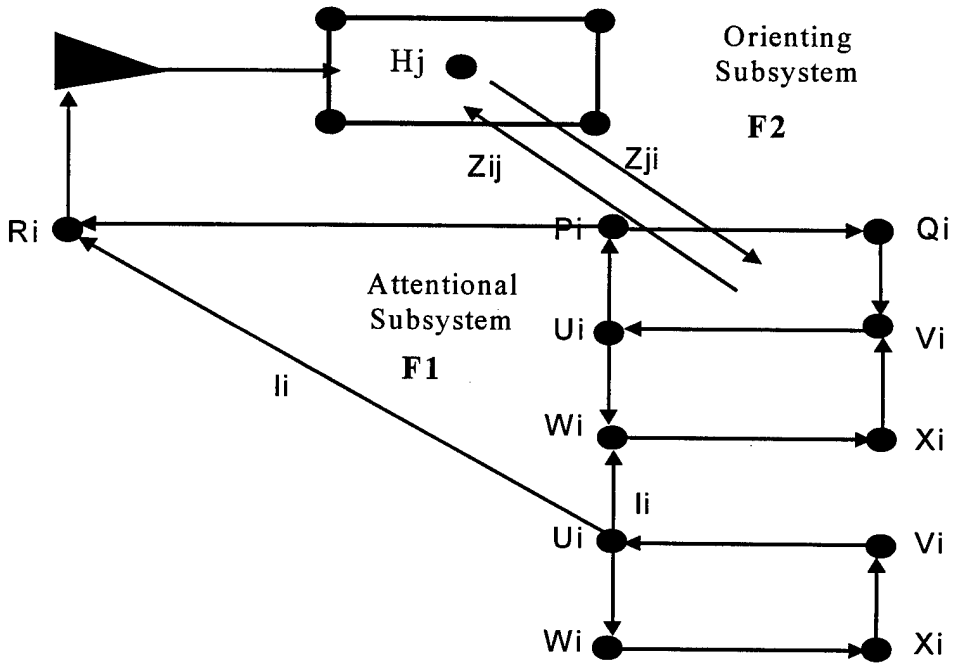


Figure 1. Basic ART2 Architecture

Learning is competitive with each F_2 neuron attempting to include the current input pattern in its category code. The actual learning process involves modification of the bottom-up and top-down LTM traces that join the winning F_2 neuron to the feature representation field. Learning either refines the code of a previously established class, based on any new information that is contained in the input pattern, or initiates code learning in a previously uncommitted F_2 neuron [2].

In either case, learning only occurs when the system is in a resonant state. This property ensures that an input pattern does not obliterate information that has been previously stored in an established class.

1.2 Fuzzy ART

The basic operation of "adaptive resonance" in the standard ART is simplified in the fuzzy ART. The basic equations which govern the fuzzy ART are based on the equations from the standard ART architecture, where the intersection operator is replaced by its

fuzzy counterpart, the minimum operator. An introduction of the mathematics governing the fuzzy ART is given here, based on [4,5,6,7,25]

The fuzzy ART system consists of three layers: the input layer (F0), processing layer (F1), and output category (F2) layer. Associated to connections between layers F1 and F2 are a set of weights directed from F1 to F2. A fundamental difference between the Fuzzy ART and prior continuous versions is the simplification of the "resonance criteria" by use of only bottom-up weights in the matching process. The matching process consists of two vector matching operations:

- Degree which input A matches output category c
- Degree which category c matches input A

The norm of a vector A, which gives an indication of its "size," is defined as

$$\|A\| = \sum |a_i| \tag{1}$$

The following three sections describe the fuzzy mathematical functions, operations and data structures associated within each of these layers.

1) Input Layer

Given an input vector A, $A = \{a_j\}$ or optionally, with the complement

$$A = \{a_j, a_j^C\}, \quad j = 1,2,\ldots,N_{in} \tag{2}$$

The addition of the complement of the input vector has the advantage that A is now self-normalized, using the definition of norm in Eq. 1:

$$\|A\| = \|(a_j, a_j^C)\| = \sum_{j=1}^{N_{in}} a_j + \sum_{j=1}^{N_{in}} (1 - a_j) = \sum_{j=1}^{N_{in}} = N_{in} \tag{3}$$

2) Output Layer

The output layer F2 consists of a set C of $N_{\max}$ active categories,

$$C = \{c_1, \ldots, c_{N_{\max}}\}$$

Each category vector $c_J \in C$ has an associated LTM weight set

$$W_J = \{w_{1,J}, w_{2,J}, \ldots, w_{2N_{in},J}\}$$

3) Processing Layer

A category *Choice Function* T_J measures the degree which input A is a match to a c_J and its associated W_J :

$$T_J = \frac{\|A \cap W_J\|}{\alpha + \|W_J\|} = \frac{\|MIN(A, W_J)\|}{\alpha + \|W_J\|} \tag{4}$$

where $\alpha > 0$ is a choice parameter.

T is the best category choice, and is calculated as the union of all T_J.

$$T = \bigcup_J T_J = \underset{J}{MAX}(T_J) \tag{5}$$

There are two cases which can occur once a category choice is attempted:

Case 1. *Equation 5 produces a choice J.* A test is performed on the preliminary choice J to test if it meets a threshold criteria called the vigilance test, where the degree to which the preliminary category matches the input A is compared against a threshold ρ

$$\frac{\|A \cap W_J\|}{\|A\|} = \frac{\|MIN(A, W_J)\|}{\|A\|} \geq \rho \tag{6}$$

If the vigilance criteria of Eq. 6 is not met, the preliminary choice [J] is said to be "reset," and another category choice according to Eqs. 4 and 5 is made from the set of active categories in C.

If the vigilance criteria is met, then the system is said to be in a state of resonance, and the input A is incorporated into category J by the following:

$$w_J^{new} = \beta\left(A \wedge w_J^{old}\right) + (1-\beta) w_J^{old} \tag{7}$$

Fast learning is said to occur when $\beta = 1$.

Case 2. *Equation 5 produces no choice.* If no category choice can be made, a new category C_{N+1} is created in C with

$$w_{N+1}^{new} = w_{N+1}^{old} = A. \tag{8}$$

Initialization: N=0

A simplified fuzzy ART architecture is described by Kasuba [17]. In the remainder of this report, we will describe a classifier based on ART2, fuzzy ART, simplified fuzzy ART, and fuzzy hypercube ART. Section 2 will introduce and define the fuzzy hypercube classifier. Section 3 will compare simplified fuzzy ART and fuzzy hypercube ANN using standard classification sets. Section 4 will give an application of the fuzzy hypercube classifier to Speaker Recognition, and Section 5 will briefly summarize the results.

2. FUZZY HYPERCUBE ARTIFICIAL NEURAL NETWORK

Modifications to ART and Fuzzy ART were done to:

1) Improve performance since they generally suffered from poor noise tolerance
2) Provide workable functional basis for hypothesis/test of neural systems

The Fuzzy Hypercube ANN (FHANN) is a conceptual-neural model based on the following general features:

1) Hypothesize and Test paradigm,
2) Hypercube category (concept) representation
3) Category Overlap
4) Fuzzy Information representation and processing
5) Information Fusion (optional)
6) Category Match Test
7) Hypervolume category limit

An FHANN, which implements the above general features, will be described. First an overview will be given of the network layer structure, followed by a detailed description of each layer. Also, global category merge and ANN initialization are described.

2.1 Fuzzy Hypercube ANN Structure

The FHANN has seven layers of processing. Figure 2 shows the functional structure, interconnection, and data flows of the ANN structure. Notice the network bi-directionality in its processing and information flows. Specific category information is fed back to the Fusion and Transform layers for state-dependent information adjustment, and to the Functional layer to performing category adjustment and learning.

The function of each of the seven layers in the FHANN is:

Input:	Input fuzzification, scaling, optional functional expansion
Transform:	Evaluation of category choice functions over active categories
Fusion:	Fusion to final ratings of category choice evaluation [Optional Config]
Hypothesize:	Choice of single hypothesis category
Test:	Pass/fail match test of hypothesis category
Functional:	Category creation, hypervolume adjustment, or category learning
Category:	Representation of clusters as fuzzy hypercube

The FHANN has several distinct differences from the basic, fuzzy, and simplified fuzzy ART. It retains the basic data structures using A and x vectors. The concept of bottom up and top down match and the learning rule are different.
Each of the layers are described in the following sections. A detailed view of the network is shown in Figure 3, where each of the blocks from Figure 2 is expanded to show more details.

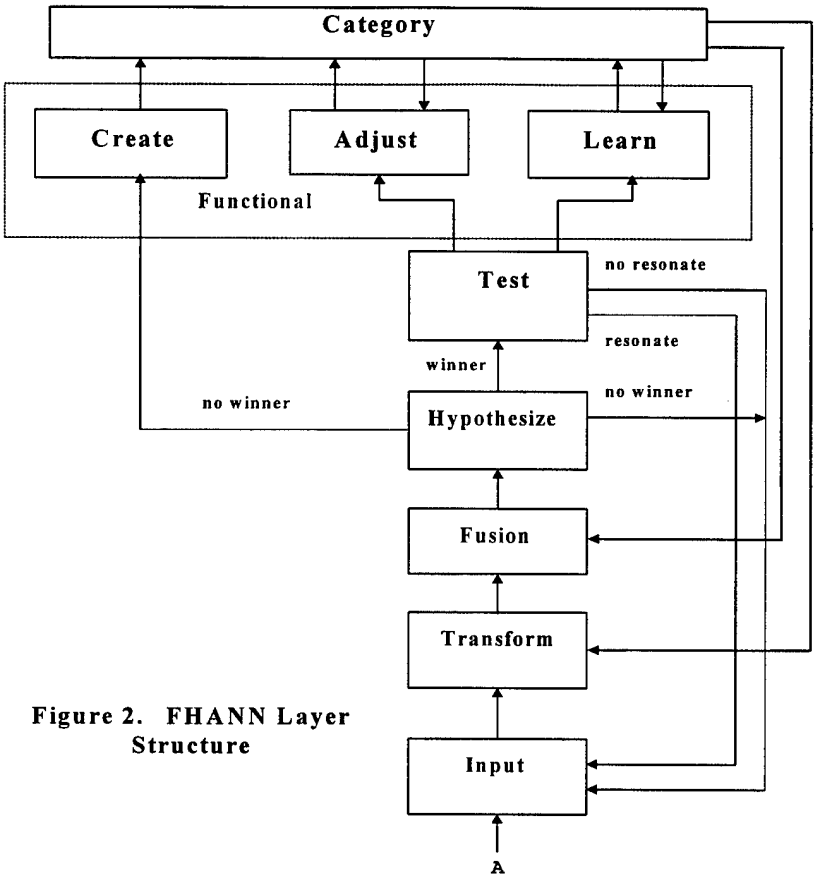


Figure 2. FHANN Layer Structure

2.1.1 Input Layer

The Input layer has three inputs and $3 * N_{in}$ outputs. The inputs are vector A, scaling values for A, and enable/disable. The vector A and associated scaling generate fuzzy set representations of the input as "Feature Vector". The enable/disable inputs control the Input layer and overall network resonance/dissonance.

2.1.1.1 Fuzzy Set Generation

The input vector and its optional complement are described by Eqs. 2 and 3.

A scaling function is defined based on the range of inputs between an expected minimum and maximum for each element of the input set A. The scaling function maps input values to the closed interval [0,1].

A discrete fuzzy set "Feature Vector" (FV) results from the set of mapped input values of A. This fuzzy set characterization is represented by a crisp set of individual membership functions s(a), one for each input feature value *a*. Note that this standard representation of a fuzzy set uses the plus sign to indicate the union of the elements.

$$FV = \frac{s(a_1)}{1} + \frac{s(a_2)}{2} + \ldots + \frac{s(a_{N_{in}})}{N_{in}}$$

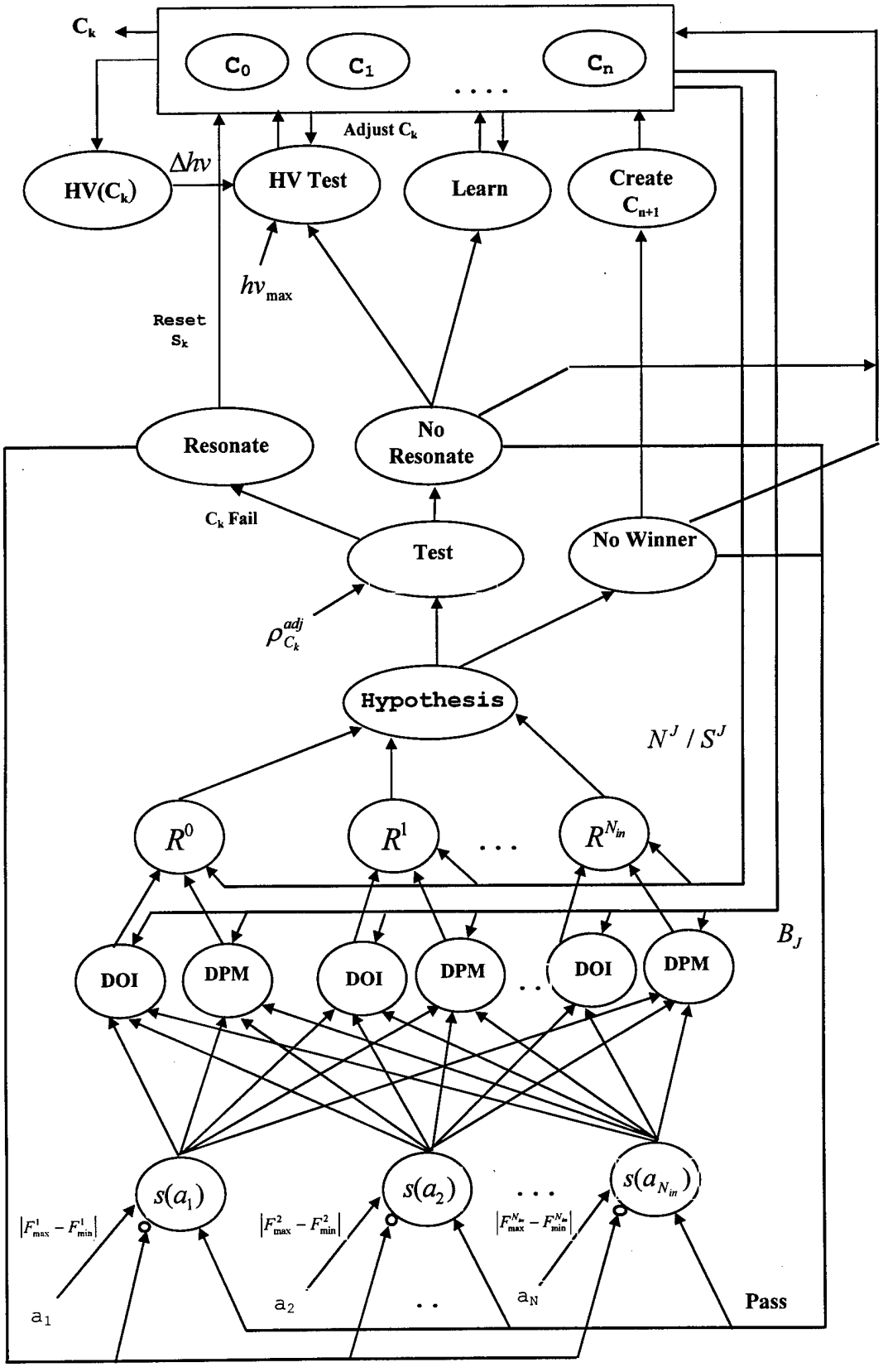


Figure 3. FHANN Functional Diagram

The function s translates input features into membership values.

$$s(a_j) = \begin{cases} 0, & a_j \le F_{Min,j} \\ z_j, & F_{Min,j} < a_j \le F_{Max,j} \\ 1, & a_j > F_{Max,j} \end{cases} \qquad over\ all \quad j = 1,2,...,N_{in} \qquad (9)$$

where $z_j = \dfrac{a_j - F_{Min,j}}{|F_{Max,j} - F_{Min,j}|}$

Individual values of F_{Min}, F_{Max} are initialized as either

- Constants
- Learned Variables

A simple weighted learning procedure can be implemented to dynamically adjust the values of F_{Min}, F_{Max}. Although , in this report, only constant values are used.

2.1.1.2 Layer/Network Resonance/Dissonance Control

The enable/disable input controls iteration/search cycling in the network. The network either continues to cycle with current input A when a suitable category is not found by the Hypothesize/Test layers, or stops cycling and begins processing of the current input, while enabling the acceptance of the next input.

2.1.2 Category Layer

The Category layer produces the output of the network and consists of a set of neurons with associated states and LTM weight values which descibe them. The LTM weights are described using a min-max feature hypercube representation of the associated J-categories defined by Simpson [21]. Strictly speaking, the categories are not hypercubes but "hyperboxes," since they do not have equal dimensions.
The category layer C is a set defined by:

$$\begin{aligned} &C = \left\{B^J, N^J, T^J, S^J\right\}, \quad J = 1,2,..., N_{\max} \\ &with \quad B^J = \left\{V_j, W_j\right\}, \quad j = 1,2,..., N_{in}, \quad V_j, W_j \in [0,1] \end{aligned} \qquad (10)$$

where $N_{\max}$ the total number of J-categories, N_{in} is the j-dimensionality of the hypercube representation according to Eq. 2.

The category layer consists of B^J as the hypercube representation of category J. B^J contains a jth dimensional description of V_j the minimum point, and W_j the maximum point. Additionally, category J consists of:

- N^J is a count of category adjustments

- T^J is an overall confidence
- S^J is the state of category J

2.1.3 Transform Layer

FV values from the Input layer are passed to the Transform layer, where they are processed by a match function and generate a degree of match between the input A and each member of the output category set C. A membership of the input in each of the output category classes is produced.

A wide range of choices is available for the matching functions such as those in ART2 or fuzzy ART (Eq. 4). Several authors examined various fuzzy match functions. Eq. 4 has been expanded by Carpenter and Gjaja [7] to Choice-by-difference, providing a more conservative learning.

$$T_j = (|W_j| - |A \wedge W_j|) + \varepsilon(|I \vee W_j| - |W_j|)$$

where ε is analogous to α of Eq. 4.

Simpson [21] developed a membership function which measures the average amount of max point and min point violations. He defines a function b which approaches 1 as the point approaches a hypercube,

$$b_h(A_h, V_j, W_j) = \frac{1}{N_{in}} \sum_{i=1}^{N_{in}} [1 - f(a_{hi} - w_{ji}, \gamma) - f(v_{ji} - a_{hi}, \gamma)]$$

where f() is the ramp function,

$$f(x,\gamma) = \begin{cases} 1 & if \quad x\gamma > 1 \\ x\gamma & if \quad 0 \le x\gamma \le 1 \\ 0 & if \quad x\gamma < 0 \end{cases}$$

The variable γ regulates the speed of membership decrease when an input is separated from a hyperbox core [21].

The idea of a smooth membership function which approaches one, as a point is within a target hypercube and steadily decreasing as it leaves the hypercube, has conceptual support. However, the function b_h above does not provide a clear indication of goodness once inside the hypercube.

2.1.3.1 Matching Functions.

A general and exact hypercube match is formulated as the linear combination of two functions [11]. The functions measure fuzzy hypercube "Degree of Inclusion" (DOI) and

"Degree of Perfect Match" (DPM). They are combined to give a measure of the level at which the scaled input matches to each feature category hypercube.

2.1.3.2 Degree of Inclusion

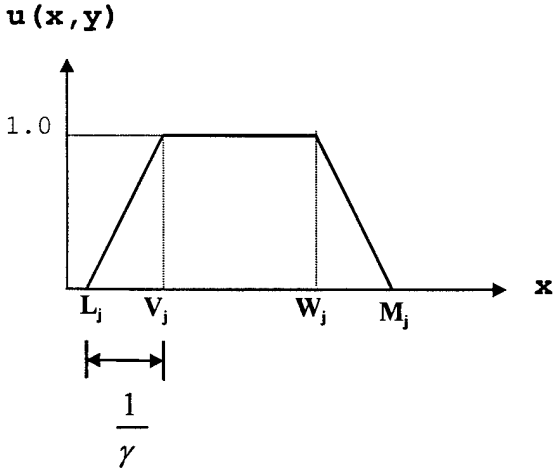


Figure 4. Trapezoidal Degree of Inclusion

DOI measures the level at which each input A_j matches to the external dimensions of, or is similar to, the j-th dimension of a J-category hypercube.

This method is similiar in function to Simpson's [21]. A trapezoidal function is used as a model for a match which gives full membership when an element of A_j is included in a category, and less than full membership outside, depending on the distance to the hypercube. Figure 4 shows the trapezoidal membership function.

The set H, consisting of upper and lower fuzzification values is defined for each hypercube B^J, as

$$H^J = \{B^J, L^J, M^J\}, \quad J = 1,2,\ldots,N_{Max}$$
$$L^J = \{L_j\}, \quad M^J = \{M_j\}, \quad j = 1,2,\ldots,N_{in} \tag{11}$$

where the sets L^J, M^J represent the fuzzification sets for the upper and lower limits.

The overall membership for DOI, $\mu^{DOI}(A)$ is defined for each j-dimension of the subsets of H^J for hypercube J (Eqs. 10,11) as a trapezoidal function $\mu_j^{DOI}(A)$.

$$\begin{aligned} &IF \quad W_j \geq a_j \geq V_j && \mu_j^{DOI}(A) = 1 \\ &IF \quad M_j \geq a_j > W_j && = 1-(a_j - W_j)\gamma \\ &IF \quad V_j > a_j \geq L_j && = 1-(V_j - a_j)\gamma \\ &IF \quad L_j > a_j \; OR \quad M_j < a_j && = 0 \end{aligned} \tag{12}$$

As a practical matter we set $|l_j - v_j| = |w_j - m_j| = \frac{1}{\gamma}$ to evenly fuzzify the hypercube. The value $\frac{1}{\gamma}$ is chosen as a constant range of "fuzzy" border around V and W for each value of A.

The oveall membership function for the DOI, $\mu^{DOI}(A)$, is defined as the following:

$$\mu^{DOI}(A) = \frac{1}{N_{in}} \sum_j \mu_j^{DOI} \tag{13}$$

Note that $0 \le \sum \mu_j^{DOI} \le N_{in}$

2.1.3.3 Degree of Perfect Match (DPM)

The DPM measures how close an input is to the mean of each J-category. The measure of the distance from the mean of each dimension of H_j is defined as DPM. DPM is a resemblance relation between the input A and an individual J-category. First, the mean M^J of a category J is defined.

$$M^J = \{m_j\} = \left| V_j - W_j \right| / 2 , \quad j = 1,2,\ldots,N_{in} \tag{14a}$$

The dissimilarity is defined as the difference between the value A and the mean of the category J, M^J,

$$Dissimilarity = |A - M^J|$$

and for each j-dimension of A and M^J,

$$Dissimilarity_j = |a_j - m_j|$$

The similarity is taken as the fuzzy complement of the dissimilarity,

$$Similarity = Dissimilarity_j^C$$

and, for each j-dimension element, we have

$$Sim_j = 1 - |a_j - m_j| \tag{14b}$$

Using the mean and similarity from Eqs. 14a and 14b, the individual j-dimension membership functions for DPM, $\mu_j^{DPM}(a_j)$ are defined as follows

$$\mu_j^{DPM}(a_j) = \begin{cases} 1 - |a_j - V_j| & if \quad W_j = V_j \\ 1 - |a_j - m_j| & if \quad W_j \neq V_j \end{cases} \tag{14c}$$

The overall membership function $\mu^{DPM}(A)$ for a single J-category is the mean of the sum of the individual values,

$$\mu_J^{DPM}(A) = \frac{1}{N_{in}} \sum_j \mu_j^{DPM}(a_j) \tag{15}$$

Again, note that $0 \le \sum \mu_j^{DPm}(a_j) \le N_{in}$ and $\mu_J^{DPM}(A) \in [0,1]$.

2.1.4 Fusion Layer

The Fusion layer integrates the best knowledge in the network concerning the matching between the current input A and each of the current categories. The knowledge used in the matching process is the DOI and DPM from the Transform layer, as well as certain feedback category information that comprise the input to the Fusion layer. A fusion function R^J is defined as the linear combination of the DPM and DOI that are dynamically weighted. The dynamic weighting is done to compensate for low DOI at the start of a hypercube matching cycle, and is based on the number of inputs representing a resultant category.

$$R^J(A) = k_1^J \mu_J^{DPM}(A) + k_2^J \mu_J^{DOI}(A) \tag{16a}$$

where

$$k_2^J = \min(K * NC(J),1), \qquad k_1^J + k_2^J = 1 \tag{16b}$$

and $R^J \le 1$

$0 \le K \le 1$, *K constant*

The functions k_1^J, k_2^J are dynamic weight functions which give the relative importance of DPM and DOI to the value R^J. Note that DPM and DOI are weighted complementary, to allow for $R^J \le N_{in}$.

The node constant NC is a dynamic weight function dependent on N^J in C_J which is the number j of input vectors which are assimilated through learning into a J-category hypercube.

$$NC(J) = \begin{cases} 0.65, & N^J = 1 \\ 0.85, & N^J = 2 \\ 0.95, & N^J = 3 \\ 1.00, & N^J > 3 \end{cases} \tag{17}$$

2.1.5 Hypothesis Layer

Inputs from the Fusion layer form a hypothesis space from which a single winnning category hypothesis is either chosen or no winner is chosen meaning no current category fits the current input and a new category is then created. Hypotheses are formed and a Winner/No Winner is chosen by first finding the maximum over all the active J-category fusion values, R^J. An active category is one which has not yet been processed using the current input as a hypothesis; if it has already been processed it is marked inactive.

$$\textit{Winner:} \quad C_W = \max_J\{R^J\} \ \textit{if} \ \ R^J \ \textit{is active and} \ \ R^J > 0, \qquad J = 1,\ldots,N_{\max}$$

No Winner: *if* R^J *is inactive ,or* $R^J \leq 0$, *over all* J

The resultant hypothesis "*Winner*" is passed with the winning category node to the Test layer, while a "*No Winner*" hypothesis is passed back to the Input layer to halt resonation of the network, as well as to notify the Functional layer to create a new category node for the current input A.

2.1.6 Test Layer

The Test layer performs a match test on the curent active winning category C_W as an input hypothesis. The value of R^J associated with C_W measures the degree of match between the category hypercube B_J and the input vector A. A modified form of the vigilance test (Eq. 6) is performed as shown in the following equations to measure if the current hypothesis passes the vigilance threshold.

$$R^J \geq \rho^{adj} \tag{18}$$

where

$$\rho^{adj} = \rho NC(J) \tag{19}$$

and NC(J) is described by Eq. 17.

Additionally, any current category which fails the vigilance test is inhibited from competing with the current input by making it inactive.

2.1.7 Functional Layer

The Functional layer performs a series of services on the Category layer including Hypervolume Measure, Hypervolume Test, Hypervolume Adjust, Hypercube Learning, and Hypercube Creation.

2.1.7.1 Hypervolume Measure

There are two methods described to calculate hypervolume: by a product and by a sum. The product hypervolume, Phv, which corresponds to the "geometric" interpretation of volume, is found by the product of the "length" in each dimension of B_J.

$$Phv = \left|\prod_{i=1}^{N_{in}} (W_i - V_i)\right| \tag{20}$$

where $0 \leq Phv \leq 1$

The product measure, however, does not allow for much expansion in the measure of a hypercube with variable length concepts or inclusion of noise, etc. Another measure of the hypervolume which increases the effect of each dimension is the unweighted sum of each of the hypercube dimensions, Shv,

$$Shv = \sum_{i=1}^{N_{in}} |W_i - V_i| \tag{21}$$

where $0 \leq Shv \leq N_{in}$

2.1.7.2 Hypervolume Test

Hypervolume limit testing and adjustment are necessary for the stability of the network. In FHANN, for each J-category, the hypervolume measure is constrained to be less than a maximum limit, Θ_{max}. The hypervolume Θ of a hypercube is bounded to keep it from expanding to infinite volume.

$$0 < \Theta \le \Theta_{max} \quad for\ each\ J \quad J = 1,...,N_{max} \tag{22}$$

where the operation which determines Θ is selected from the set of hypervolume measure operations defined by Eq. 19 or Eq. 20.

$$\Theta = \{Phv, Shv\}$$

2.1.7.3 Hypervolume Adjust

If the limit Θ_{max} is exceeded, the category hypervolume is adjusted. The excessive volume $\Delta\Theta$ is found from the current hypervolume, Θ, by the following:

$$\Delta\Theta = \begin{cases} (\Theta - \Theta_{max}) / N_{in} & \Theta > \Theta_{max} \\ 0 & Otherwise \end{cases} \tag{23}$$

where N_{in} is the current input dimensionality as in Eq. 11. The hypervolume of category J is adjusted whenever $\Delta\Theta > 0$ by

$$\begin{aligned} W_J^{new} &= \max\{(W_J^{old} - \Delta\Theta / 2), 0\} \\ V_J^{new} &= \min\{(V_J^{old} + \Delta\Theta / 2), 1\} \end{aligned} \tag{24}$$

This operation brings the hypervolume measure of category J within the value of Θ_{max} as required by Eq. 22.

2.1.7.4 Hypercube Learning

The inclusion of input A into the winning category hypercube B_J is done through a learning algorithm which adjusts the category J hypercube. In general, each feature of A, a_j, selectively adjusts its respective limits in W_J and V_J through the following hypercube learning algorithm. A learning adjustment factor r is used to set the algorithm's rate of learning.

Hypercube Learning Algorithm. Given an input vector A, a hypercube B_J, and a learning adjustment factor r, learning is performed for each dimension of A.

1. Determine if input is inside or outside of interval B_J
2. DO: Respective Case 1 or Case 2 and subcases below.

Case 1: Input is outside of interval B_J

Case 1.1 Input above W_J: Increment W upwards

$$IF \quad a_i > W_{Ji}^{old} \; THEN \quad W_{Ji}^{new} = W_{Ji}^{old} + r(a_i - W_{Ji}^{old}), \quad V^{new} = V^{old}$$

Case 1.2 Input below V_J: Decrement V downwards.

$$IF \quad a_i < V_{Ji}^{old} \; THEN \quad V_{Ji}^{new} = V_{Ji}^{old} + r(a_i - V_{Ji}^{old}), \quad W^{new} = W^{old}$$

Case 2: Input is inside of interval B_J

Case 2.1 Input is closer to V; move W closer in to V.

$$IF (a_{ii} \geq V_{Ji}^{old} \quad AND \quad a_i \leq W_{Ji}^{old} \; AND \quad W_{Ji}^{old} - a_i \geq a_i - V_{Ji}^{old})$$

$$THEN \quad W_{Ji}^{new} = W_{Ji}^{old} - r(W_{Ji}^{old} - a_{ii}), \quad V_{Ji}^{new} = V_{Ji}^{old}$$

Case 2.2 Input is closer to W; move V closer to W.

$$IF (a_{ii} \geq V_{Ji}^{old} \quad AND \quad a_i \leq W_{Ji}^{old} \; AND \quad a_i - V_{Ji}^{old} > W_{Ji}^{old} - a_i)$$

$$THEN \quad V_{Ji}^{new} = V_{Ji}^{old} + r(a_i - V_{Ji}^{old}), \quad W_{Ji}^{new} = W_{Ji}^{old}$$

2.1.7.5 Hypercube Creation.

The creation of a hypercube requires that the overall hypervolume limit is adjusted through the hypercube dimension, hd_{max}, which is defined as the maximum hypercube dimension, assuming equal size in each dimension:

$$hd_{\max} = \frac{\Theta_{\max}}{N_{in}} \tag{25}$$

Upon Creation:

1. Initial settings are equal.

$$W_{Ji}^{new} = V_{Ji}^{new} = a_i$$

2. Number of active J-categories is incremented by one.

$$N_{\max} = N_{\max} + 1$$

2.2 Auxiliary Network Functions

The FHANN has optional and required auxiliary functions. The required functions include FHANN Initialization of system, and FHANN parameter initializations. Optional function is the Global Category merge which was tested, but results are not reported here.

2.2.1 Category Merge

A global merge is defined as a combination of hypercube cluster classes produced by the FHANN which are very "close" to one another. This operation is performed outside of the neural network processing and does not affect any of the internal operations of the network. It does utilize detail parameters generated by the network.

This process occurs over time between FHANN cycles and can be considered a long term averaging process. There are two measures which are used to indicate whether a global merge is to take place:

a) Volume difference between hypercube categories and
b) Magnitude of rating R from Eq.16 between two categories.

The volume parameter is defined as follows:

$$N_{in} \times \Delta \leq volume \tag{26}$$

The value N_{in} is the number of input nodes and Δ the hypervolume per node.

A category merge function is defined. First, merge parameters are obtained over all possible different pairs of the current categories defined. Next, the merge criteria are applied and partition current categories into a final set which is compacted. [Note that during testing, the compacting occurred very rarely]. The criteria are expressed in terms of acceptance/rejection regions in the volume difference/rating mapping.

$$\begin{aligned} &0.0 \leq \Delta vol(c1,c2) \leq 1.10 \; and \quad R(c1,c2) > 1.00 \quad OR \\ &1.1 < \Delta vol(c1,c2) \leq 1.50 \; and \quad R(c1,c2) > 1.00 \quad OR \\ &1.5 < \Delta vol(c1.c2) \leq 1.75 \; and \quad R(c1,c2) > 1.40 \end{aligned} \tag{27}$$

These were experimentally derived and were only used to evaluate the concept of global clustering criteria within the context of the hypercube structure.

2.2.2 FHANN Initialization

The initialization is performed on the FHANN as follows:

Step 1. Enable all categories, set count, and confidence is "none".

$$N^J = 0, \quad T^J = none, \quad S^J = enabled$$

Step 2. Set available category count to zero.

$$N_{avail} = 0$$

2.2.3 FHANN Parameters

The FHANN contains several variables and parameters which are under the control of the designer of the system. The parameters are to be set before NN execution, and are as follows, with references to defining equations:

P1: Size of input layer, N_{in} (Eq. 2)

P2: Maximum number of output categories, $N_{\max}$ (Eq. 10)

P3: Complement of input A in input layer (Eqs. 2, 3)

P4: Maximum, minimum input scaling values in A, $F_{\max}, F_{\min}$ (Eq. 9)

P5: Hypercube fuzzification, $\frac{1}{\gamma}$, for fuzzy border around each a_j (Eq. 12)

P6: Fusion function weight, K, the fraction of DPM (Eq. 16b)

P7: Base value of vigilance, ρ (Eq. 19).

P8: Hypervolume maximum, $hv_{\max}$ (Eq. 22)

P9: Learning adjustment factor, r (in Hypercube Learning Algorithm)

P10: Hypercube Measure Type (Eq. 19 or Eq. 20)

One optional parameter, if known, is:

O1: Known Number of input classes, N_{class}

3. NETWORK COMPARATIVE PERFORMANCE ANALYSIS

The performances of fuzzy ART and the FHANN are compared using two standard classification data sets: Iris Flower and Wisconsin Diagnostic Breast Cancer. A simulation of the FHANN's was made using MATLAB. This section provides details on the structures for the simulator and results of the comparative testing performed.

3.1 Test Measurement Definitions

Test data items which are measured to quantify the performance of the FHANN's are defined in this section. Note that for all data tested, the correct responses are known but are used only in the calculation of the ANN performance.

The definitions for CC, TC, and IC are test measurements made on the FHANN's of the number of categories created by the networks.

CC="Correct Category": Count of categories containing data designated as correctly classified.
TC="All Created Categories": Count of all categories created
IC=TC-CC="Incorrect Categories": Count of categories containing data designated as incorrectly classified

The definitions for the following are test measurements made on the FHANN's of the number of input data elements placed in the various classes defined as CC, TC, or IC. The prefix function "D" indicates the measure of data in a specific category.

D="Data": Function indicating individual data sets in a specific class/category.

D(CC)=Data in Correct Category
D(TC)=Data in All Created Categories
D(IC)=Data in Incorrect Categories

Correct Classifications Vs Incorrect Classifications in "Correct Categories"

One measure of the performance of the FHANN being used is the fraction of how many of the data in the created categories are correct or incorrect, with respect to the sum of the correct and incorrect data input to the system. These are defined by Eqs. 28 and 29.

$$P_{CC}^{C} = \frac{\sum D(CC)}{\sum \{D(CC) + D(IC)\}} \tag{28}$$

$$P_{IC}^{C} = \frac{\sum D(IC)}{\sum \{D(CC) + D(IC)\}} \tag{29}$$

Note that $P_{CC}^{C} \cap P_{IC}^{C} = \varnothing$

Correct Classifications Vs Incorrect Classifications in "All Created Categories"

A measure of the performance of the FHANN in all categories is a measure of the fraction of how many of the data that are in correct or incorrect categories, with respect to all the data input to the system. These are defined by Eqs. 30 and 31.

$$P_{CC}^{A} = \frac{\sum D(CC)}{\sum D(TC)} \tag{30}$$

$$P_{IC}^{A} = \frac{\sum D(IC)}{\sum D(TC)} \tag{31}$$

Note: The following inequality may exist due to the fact that several potential categories may have been eliminated prior to classification:

$$D(TC) \geq D(CC) + D(IC)$$

Total Count of Incorrect Categories

This measure is the absolute number of incorrect categories created. It gives an indication of the degree of useless categories which the network creates for a given set of conditions. It is given by Eq. 32.

$$P_{IC} = \sum IC \tag{32}$$

Correct Vs. Incorrect Categories

A measure of the performance of the FHANN, with respect to the number of correct categories it creates and the number of categories created, is measured as a fraction of the number of correct categories, if the value of N_{class} is known. It is given by Eq. 33.

$$P_{CC}^{N} = \frac{\sum CC}{N_{class}} \tag{33}$$

Likewise, for number of incorrect categories, Eq. 34.

$$P_{IC}^{C} = \frac{P_{IC}}{N_{class}} \tag{34}$$

3.2 Function Descriptions

The analysis was performed using a system running under MATLAB with the following functions:

(1) Anode: Creates an output node for ANN. An output node contains: category number (Cat), count of adjustments (Nj) and B. Nj is initialized to one, the max points W to zeros and min points V to ones. Category number is equal to the number of categories.

(2) Volume: Calculates the volume of a hypercube. The volume is defined as the sum of all sides of a hypercube (Eq. 21).

(3) Min-max: Finds minimum and maximum values of each input feature from an input data file (Eq. 9)

(4) Scale: Scales input vector to be within [0,1] (Eq. 9).

(5) NC: Generates node constants for different Nj's (Eq. 17).

(6) DOI: Calculates degree of inclusion (Eq. 13).

(7) DPM: Calculates degree of perfect match (Eq. 15).

(8) Adjust: Adjusts vector B so that the overall hypercube volume is within limit (HV_max). It requires two inputs: Vector B, and the difference between the max and the actual volume Delta (Eq. 23).

(9) Learn: Updates and returns the values of B (which contain the min (V) and max (W) of the hypercube) using input vector A.

(10) Trans: Transforms input data into a membership function R using DOI and DPM (Eq. 16a).

(11) Test: Performs a match test (Eq. 18) on the input vector and category input hypothesis.

(12) Stat: Calculates the statistics of the output and does some output routines.

3.3 Main Program Description

Interactive input values are requested for Vigilance, Gamma, Learning Adjustment Factor r, K (in Eq. 16b) and maximum hypercube volume Θ_{max}.

The input data file which contains the feature vectors is then scanned using Function *Min_Max* to obtain F_{max}, F_{min} the maximum and minimum values. These are used to scale the input data set using the Function *scale*.

The first data set is read and classified as category one. An initial output node is created for this data set using Function Anode. Once a node is created, the number of nodes (Num.) is incremented by one. A scan loop reads input feature sets from the input file until an end of file is detected.

Inside the scan loop, data sets are read and scaled. The scaled input data goes through the transform layer using Function *Trans* which takes the input vector and generates a membership vector R. Values of vector R are then sorted in ascending order. The category node which has largest R value is the current winner.

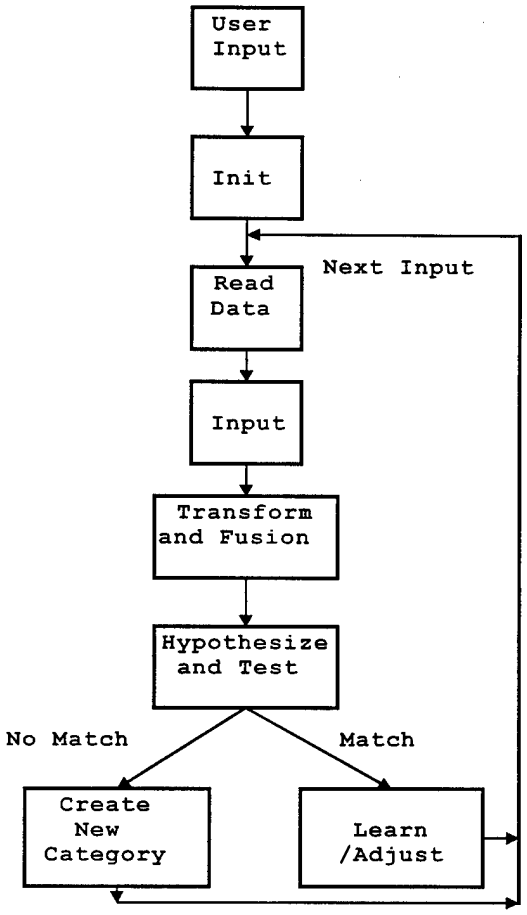


Figure 5.
FHANN Functional Block Diagram

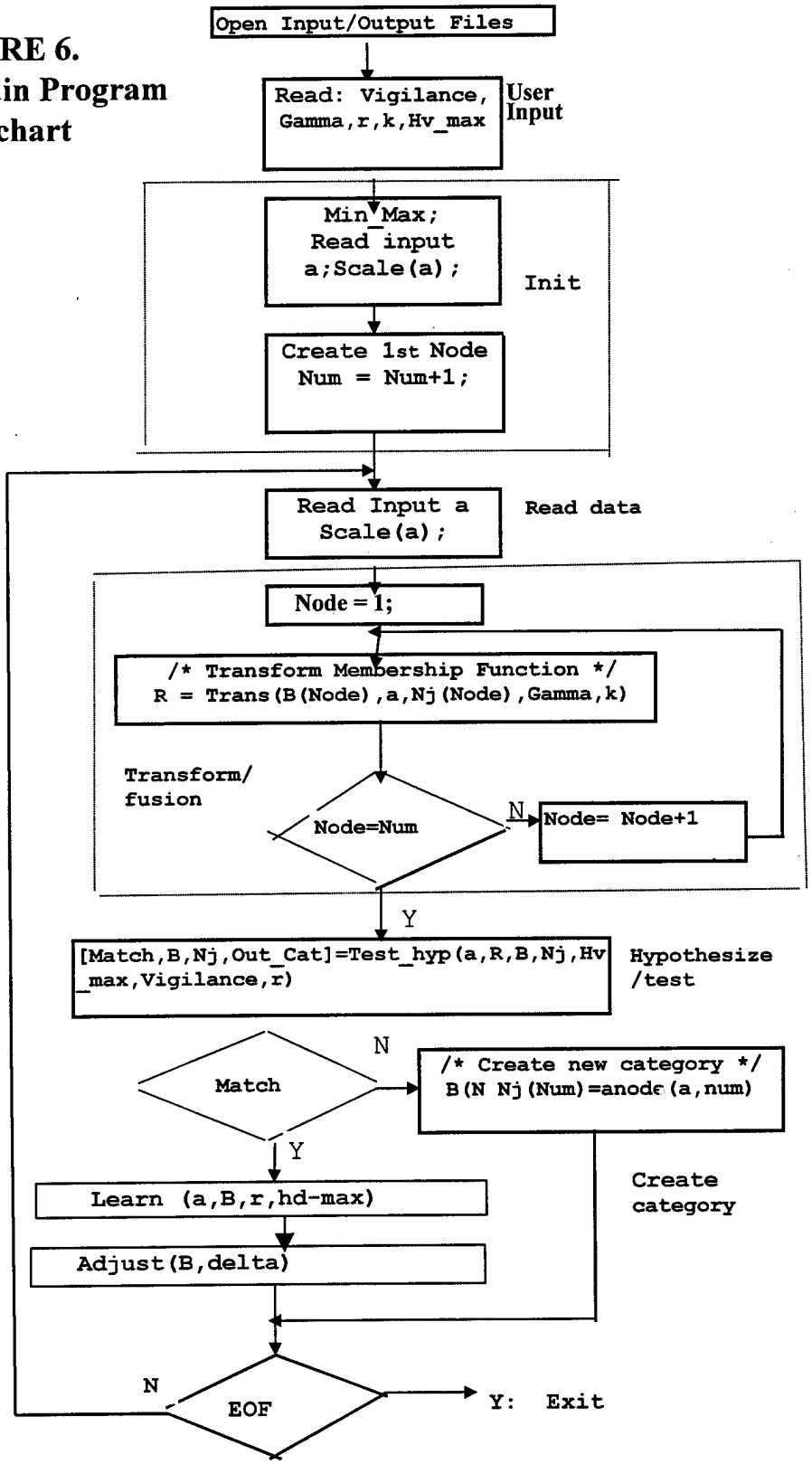
The input data set is set at the winning category node. A match function is performed against the input data and the winning category's maximum and minimum points using the Function *test*.

If it matches, then a hypercube volume test is performed to see if it is within the specified maximum hypercube volume. If not, the min and max points are adjusted. The winning category node learns the input data through the Function *learn*. If it does not match, that winning category node is disabled and the category node with second largest R value is activated.

The process continues until the last category node is encountered with no match. At this point, a new category node is created for that input data set.

Figure 5 shows the Main program's Functional block diagram. For a more detailed description, see the Main program's flow chart in Figure 6.

FIGURE 6.
FHANN Main Program Flowchart



3.4 Testing Methodology

The FHANN was tested using a parametric testing method. A set of baseline values was chosen and the program was run while varying a single parameter in the baseline and keeping the others constant. The purpose was to see the effects of one variable on the output of the network. The data files that were used as inputs were: iris.dat (a 3-feature, 150-cases of the Iris flower classification set) and WDBC.dat (a 2-feature, 569-cases of the Wisconsin Diagnostic Breast Cancer).

For each run, the following measures of performance were generated and graphed:

- Percentage of correct classification in all classes (Eq. 28)
- Percentage of correct classification in correct classes (Eq. 30)
- Number of incorrect classes formed.

From results of the parametric test, the best value for each parameter was chosen to maximize the output of the network.

For Iris.dat, the following optimal output was obtained:

- Percentage of correct classification in all classes: 76%
- Percentage of correct classification in correct classes: 79%
- Number of incorrect classes formed: 1

when Gamma = 5; Vigilance = 0.7; r = 0.18; k = 0.1; Hv_max = 1.5

For WDBC.dat,

- Percentage of correct classification in all classes: 88%
- Percentage of correct classification in correct classes: 90%
- Number of incorrect classes formed: 1

when Gamma = 5; Vigilance = 0.6; r = 0.1; k = 0.1; Hv_max > 0.5

Figures 7 and 8 show a sample run of the output from the network. A confusion matrix is generated for each case where columns are the actual input categories and rows are the output categories formed. Fig. 9 plots the percent of correct, number of correct classifications, and number of correct categories versus vigilance for both iris.dat and WDBC.dat.

3.5 Comparison Between SFAM and FHANN

Simplified Fuzzy ART Map (SFAM) [17] has higher percentage of correct classifications than FHANN in the Iris case and about the same in WDBC case. However, the numbers of incorrect classes, N_incor, are much higher for SFAM over FHANN in both cases. This occurs because SFAM is more scattered than FHANN as it is more sensitive to changes in the input features. FHANN appears more stable, displaying greater insensitivity to noise in the input features

INPUT DATA FILE: iris.dat

Vigilance = 0.700000
Gamma = 5.000000 ; k = 0.100000
Learning adjustment factor r = 0.180000
Maximum hypercube volume = 1.500000

Confusion matrix:

Out	1	2	3	Total	Percent
1	50	0	0	50	0.33
2	0	29	43	72	0.48
3	0	21	2	23	0.15
4	0	0	5	5	0.03
Total:	50	50	50	150	
Percent:	1.00	0.42	0.86		1.00

Number of correct classifications: 114
Number of incorrect classifications: 36
Percentage of correct in all classes: 76/100
Percentage of correct in correct classes: 79/100
Percentage of correct classes: 100/100
Number of incorrect classes: 1

Figure 7. FHANN's Optimal Output for Iris.dat

INPUT DATA FILE: WDBC.dat

Vigilance = 0.600000
Gamma = 5.000000 ; k = 0.100000
Learning adjustment factor r = 0.100000
Maximum hypercube volume = 1.000000

Confusion matrix:

Out	1	2	Total	Percent
1	157	16	173	0.30
2	39	341	380	0.67
3	16	0	16	0.03
Total:	212	357	569	
Percent:	0.74	0.96		1.00

Number of correct classifications: 498
Number of incorrect classifications: 71
Percentage of correct in all classes: 88/100
Percentage of correct in correct classes: 90/100
Percentage of correct classes: 100/100
Number of incorrect classes: 1

Figure 8. FHANN's Optimal Output for WDBC.dat

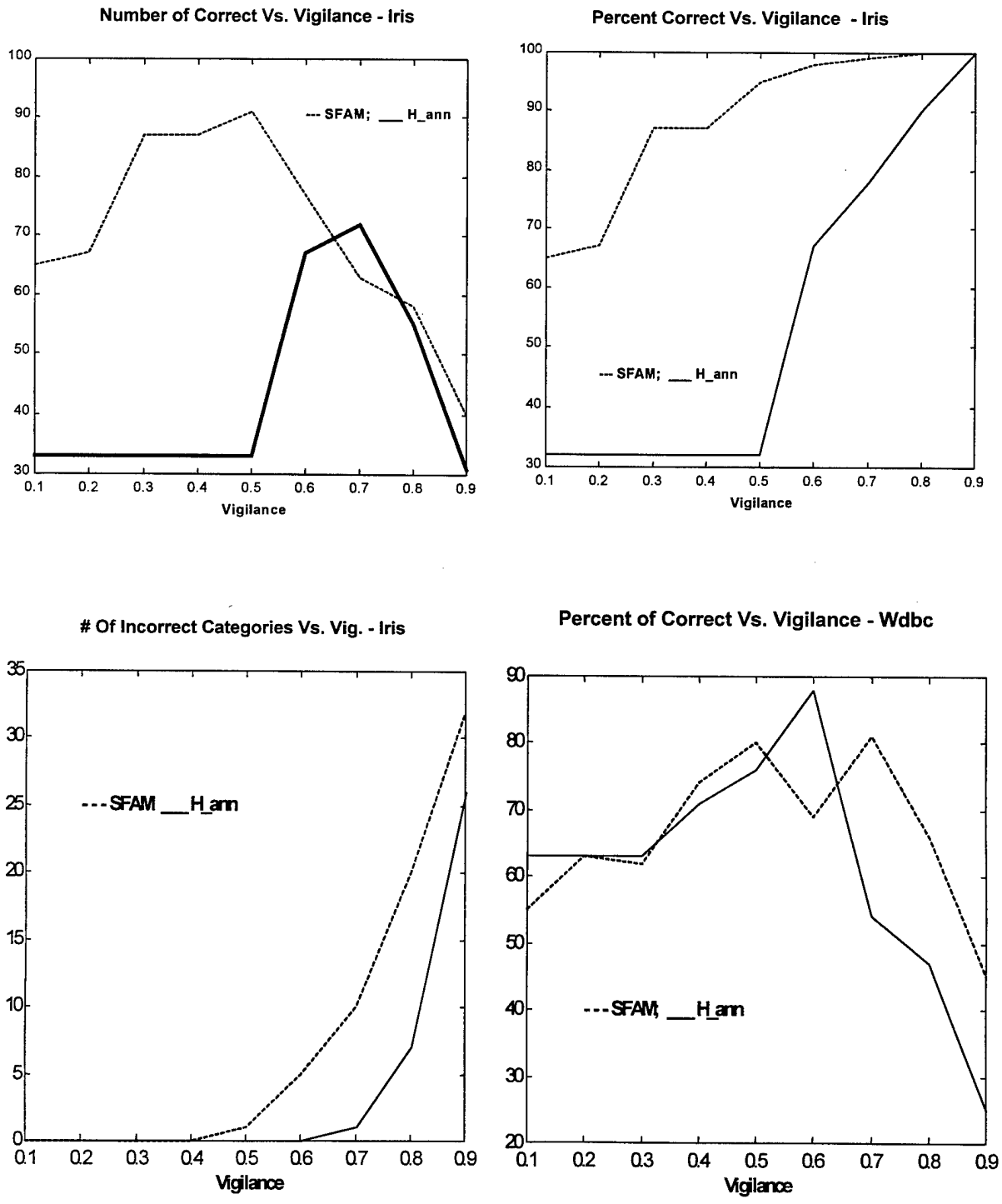


Figure 9. FHANN and SFAM comparison

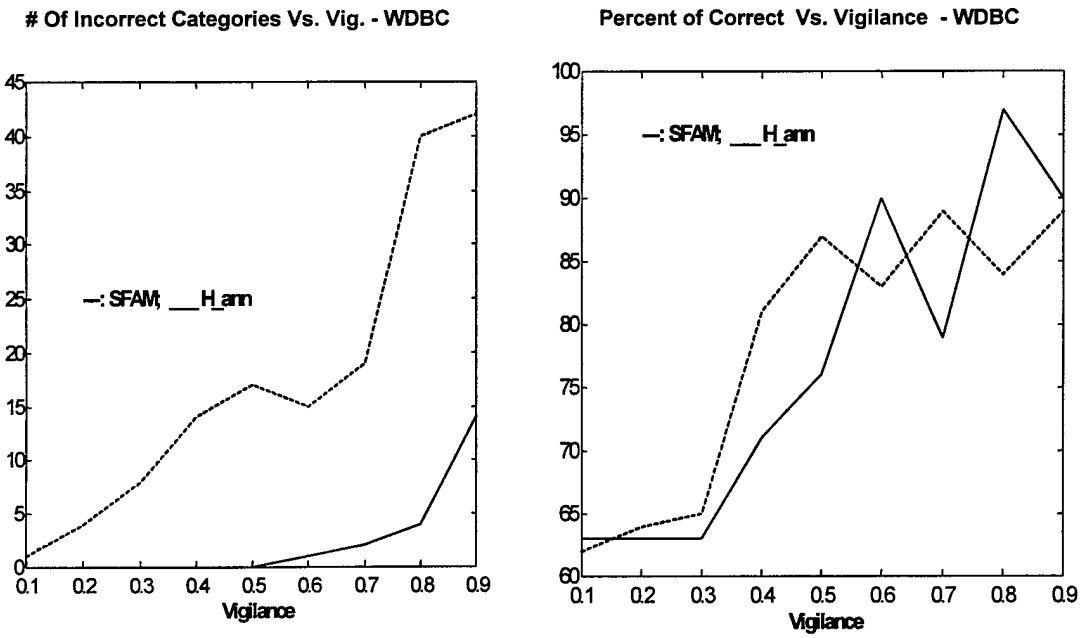


Figure 9. FHANN and SFAM comparison (Contd)

4 SPEAKER RECOGNITION TEST

This section describes specific testing of the FHANN for recognition of speakers.

4.1 Introduction

Speaker recognition from speakers' voices requires a clustering/classification process which can group signal feature representations of the voices into reliable speaker groups. The clustering/classification should be able to form any number of classes dynamically, and tolerate the noisy and overlapping domain of speakers' feature vectors. ART architectures did not perform well during prior speaker recognition testing [13,14,16]; hence, the fuzzy hypercube classifier was employed in this test in an attempt to overcome this and other problems.

There were two speaker data sets used for the formal testing of the system, the Switchboard [22] and the Greenflag [23]. Switchboard data consists of 26 speakers and Greenflag 41 speakers in a tactical environment. Both data sets were interfaced using the NIST/SPHERE V2.2 software. Speaker test data were grouped into non-overlapping sets of 8, 12, and 16 speakers.

The Feature Processor is described in detail in [15]. Features were analyzed for two characteristics, separability, and maximum / minimum values. For an introductory background in this area, see Pellisier [20]. This report deals only with the classification performance of sets of input features provided by the Feature Processor.

There were a number of fixed and varied parameters corresponding to specific subsystems [15].

FHANN-Variable Parameters

Vigilance
Maximum Hypervolume

Overall System Variable Data and Parameters

Test Data Sets
Number of Speakers
Number of correct and incorrect classifications per Test Set

4.2 Speaker Testing

The following are defined parameters which were measured during the speaker testing.

1. Average, Minimum, and Maximum of Total Number of Actual Speakers Correctly Identified. Using P_{CC}^{C} from Eq. 28, the average is defined as:

$$\overline{P_{CC}^{C}} = \frac{\sum_{N_{tests}} P_{CC}^{C}}{N_{tests}} \tag{34}$$

where $D(CC) > 1$ in Eq. 28, and N_{tests} are the total number of speaker recognition tests performed. The Minimum and Maximum are defined as:

$$Min(P_{CC}^{C}), \quad Max(P_{CC}^{C}) over \quad all\, N_{tests} \tag{35}$$

Figures 10 and 11 both display the Average, Minimum, and Maximum of P_{CC}^{C} as a function of the vigilance parameter and the maximum hypervolume [within a small range of values.]

Figure 10. Percent of Correct Classification vs. Vigilance Values

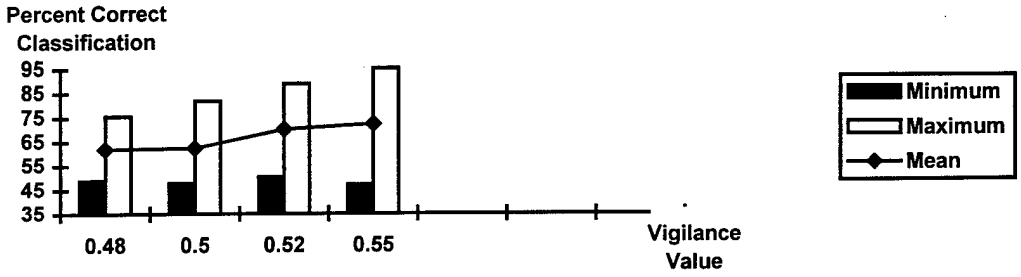
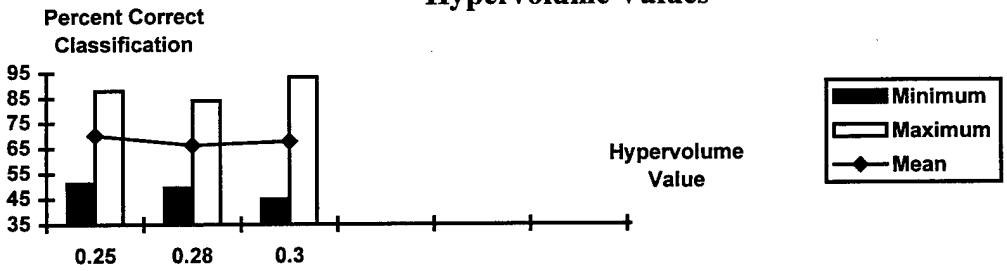


Figure 11. Percent of Correct Classification vs. Hypervolume Values



2. Average, Minimum, and Maximum of Total Number of Actual Speakers Incorrectly Identified. Using P_{IC}^{C} from Eq. 29, the average is defined as:

$$\overline{P_{IC}^{C}} = \frac{\sum \frac{P_{IC}^{C}}{N_{tests}}}{N_{tests}} \tag{36}$$

where $D(IC) > 2$ in Eq. 36. The Minimum and Maximum are defined as:

$$Min(P_{IC}^{C}), \quad Max(P_{IC}^{C}) over \quad all\, N_{tests} \tag{37}$$

3. Average, Minimum, and Maximum of Spurious Categories[1] Generated. Test values are generated using P_{IC}^{N} Eq. 33 with the condition $D(IC) \leq 2$:

$$\overline{P_{IC}^{N}} = \frac{\sum \frac{P_{IC}^{N}}{N_{tests}}}{N_{tests}} \tag{38}$$

[1] Spurious Categories are speaker categories with a maximum of 1 or 2 entries over all time. This is extended to time-dependent spurious categories where the entry count decreases fractionally and is proportional to age.

where $D(IC) \leq 2$ in Eq. 38 and 39. The Minimum and Maximum are defined as:

$$Min(P_{IC}^{N}), \quad Max(P_{IC}^{N}) over \quad all\ N_{tests} \tag{39}$$

Figures 12 and 13 display spurious speaker category creation in the network as a function of vigilance parameter and maximum hypervolume.

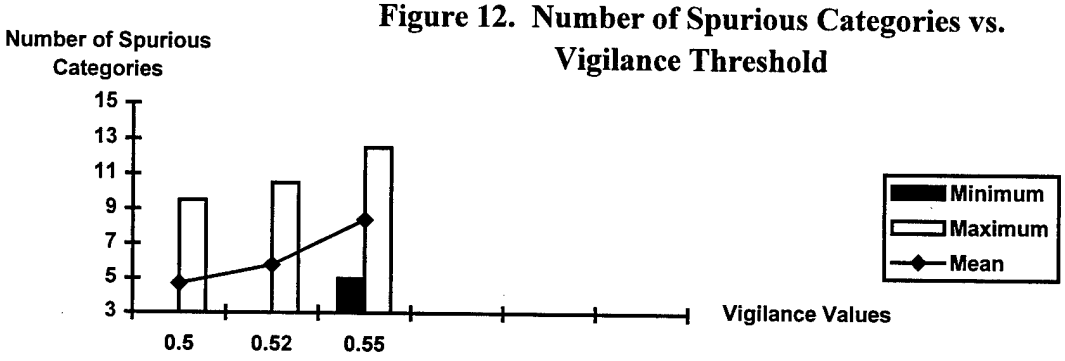


Figure 12. Number of Spurious Categories vs. Vigilance Threshold

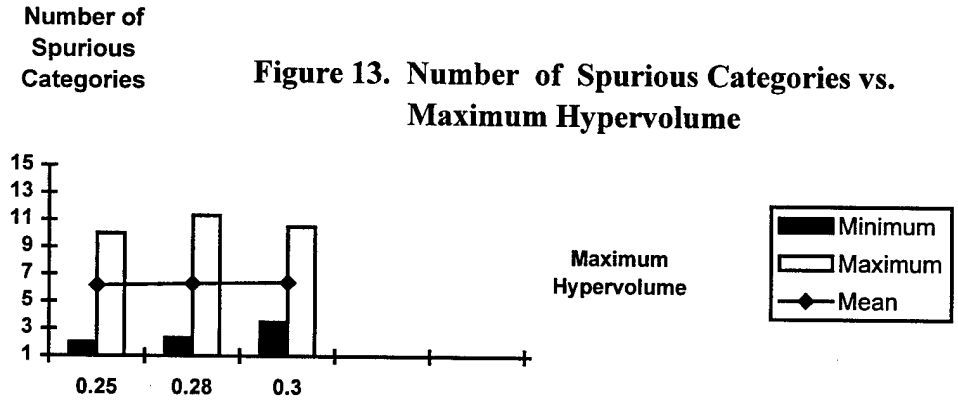


Figure 13. Number of Spurious Categories vs. Maximum Hypervolume

4. Average, Minimum, and Maximum of the Number of Correct Speaker Categories is defined as:

$$\overline{P_{CC}^{N}} = \frac{P_{CC}^{N}}{N_{tests}} \tag{40}$$

$$Min(P_{CC}^{N}), \quad Max(P_{CC}^{N}) over \quad all\ N_{tests}$$

Figures 14 and 15 show the development of correct speaker categories for the 8 speaker test.

Figure 14. Number of Correct Categories vs. Vigilance Threshold

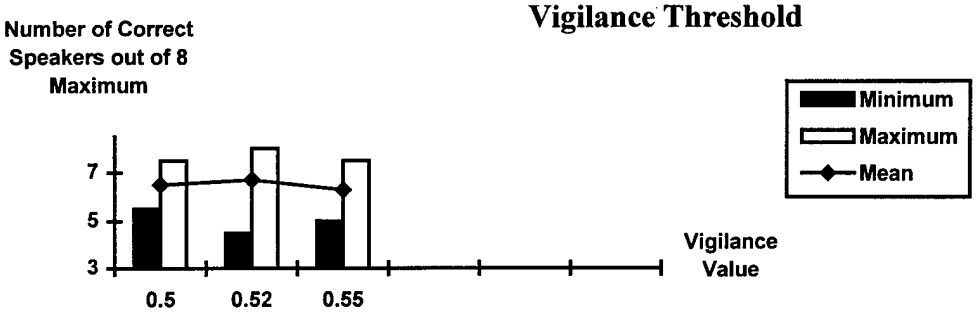
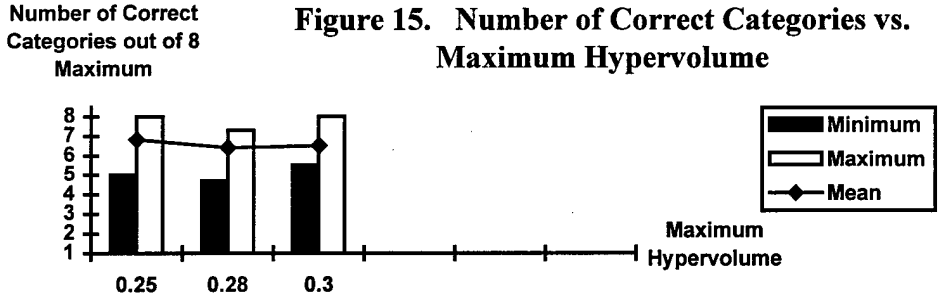


Figure 15. Number of Correct Categories vs. Maximum Hypervolume



Summarized test results for the FHANN performance are shown in Tables 1 and 3 for the 8 speaker tests, and Table 2 for the 12 speaker tests.

TABLE 1. FHANN, 8-Speaker Test

Test Data Set for 8 Speakers	*Average Number of Correct Categories Generated (8 max)*	*Average Number of False Categories Generated (8 max)*	*Average Number of False Categories Deleted per Data Set*
Switchboard May 95	6.29	0.29	1.86
Greenflag	6.57	0.23	5.77

TABLE 2. FHANN, 12-Speaker Test

Test Data Set for 12 Speakers	*Total Number of Speakers in Test*	*Total voiced Speaking Time (mins)*	*Overall Correct Classification (%)*
Switchboard May 95	12	16.71	67.25
Greenflag	23	4.77	68.75

TABLE 3. FHANN, Summary of Test Results

Test Data Set for 8 Speakers	*Total Number of Speakers in Test*	*Total Voiced Speaking Time (hrs)*	*Overall Correct Classification (%)*
Switchboard May 95	26	2.69	69.7
Greenflag	41	2.96	70.3

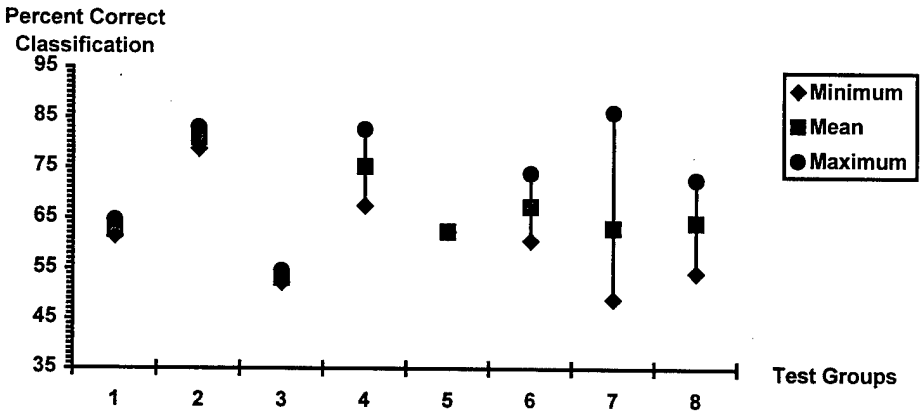


Figure 16. Percent correct vs. Greenflag Test Groups

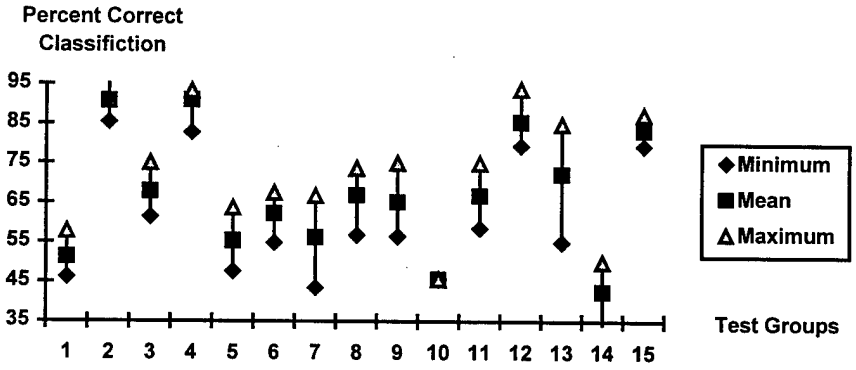


Figure 17. Percent correct vs. Switchboard 95 Test Groups

A breakout of the Switchboard and the Greenflag individual test groups is shown in Figures 16 and 17. Here the results are given for each group individually, with associated absolute minimum, maximum and mean. Values are derived using Eq. 34 and Eq. 35.

The overall system test results are shown in Table 4. This includes all speaker groups.

TABLE 4. Overall System Test Results

Test Data Set for all Speaker Groups	*Total Number of Speakers*	*Total Voiced Speaking Time (hrs)*	*Overall Correct Classification (%)*	*Standard Deviation (avg)*	*Maximum-Minimum (avg)*
Switchboard [14]	26	3.0	66.9	5.0	14.5
Greenflag [15]	41	3.0	66.6	6.6	13.4

4.3 Test Results Discussion

The speaker tests performed in the previous section will be described. They will be grouped into ANN Tests and Overall System Test. The Overall system test included the testing of the Feature Processor with the FHANN.

4.3.1 FHANN Specific Tests

FHANN parameters modified during testing were maximum hypervolume and the vigilance parameter. In Figures 10 and 11, the effects of the hypervolume (HV) and vigilance on the system performance are demonstrated for a very limited set of values. The bar charts indicate that the HV mean value does not change much, but the spread between the maximum and minimum increases with increasing hypervolume. Thus, one would like to keep the HV as small as possible, while maintaining an acceptable classification performance level.

Spurious categories are displayed in Figures 12 and 13. Here an increase in vigilance produced an increasing number of spurious categories, as well, and wider swings on the maximum and minimum values. This makes sense since increasing the vigilance produces a more "specific" network, and hence more extra category nodes would seem to be created under these conditions. The effects of hypervolume, in contrast, seemed to have little effect on the spurious category creation.

The bar chart in Figure 14 shows a slight decrease in the number of correct speaker categories with an increase in vigilance, but a narrowing of the max/min values. For the hypervolume, in Figure 15, there is a slight decrease also, but not as clear as that for vigilance.

Figures 10 to 13 relate vigilance and hypervolume to the number of categories generated in the category layer in the network, both correct and spurious. In the case of vigilance, there is a slight downward trend to performance for a correct number of categories, and a strong positive increase in the number of spurious categories generated as it increases. In the case of hypervolume maximum, increasing values have little effect on the number of correct categories and spurious categories.

Note that the tests should not be used to draw generalizations over the entire data space due to the fact that data sets are narrow in range.

4.3.2 Overall System Tests

Overall testing results are shown in Figures 16 and 17 and Tables 2 to 4. In Figures 16 and 17, system performance is plotted for each speaker group, and is shown for its mean, minimum, and maximum. The results are synopsized in Tables 2 and 3, giving the standard deviation averaged over all groups for each group, as well as the maximum to minimum value spread averaged over all the groups.

From these data, it can be seen that Greenflag had a smaller minimum to maximum spread, and, with the exception of group number 7, all appear well behaved. In the switchboard case, the spread is much more in all groups with group number 13 the greatest. However, the switchboard data was still more well behaved and better clustered as is shown by its smaller standard deviation in Table 4.

The performance of the test groups is nearly identical at 67% for an 8 speaker group maximum.

5. CONCLUSION

A seven layer neural network architecture is described which performs a hypothesize and matching test between an input vector and a fuzzy hypercube category representation of the input vectors.

The network has a variable hypervolume limit to accommodate noisy feature hypercubes.

Testing was performed to compare the fuzzy hypercube classifier with fuzzy ART using Iris Flower and breast cancer standard data sets. The fuzzy hypercube classifier displayed better tolerance to noise in these tests.

The fuzzy hypercube classifier was also tested with Switchboard and Greenflag data sets. The performance for 8 speaker groups is 67% overall correct classification.

6. BIBLIOGRAPHY

[1] J. Bezdek and S. Pal (Eds.), Fuzzy Models For Pattern Recognition, IEEE Press (1992).

[2] G. Carpenter and S. Grossberg, "ART2: Self-organization of Stable Category Recognition Codes for Analog Input Patterns," Applied Optics, Vol. 26, No. 23, pp. 4919-4930 (1987).

[3] G. Carpenter and S. Grossberg, "ART3: Hierarchical Search Using Chemical Transmitters in Self-Organizing Pattern Recognition Architectures," Neural Networks, Vol. 3, pp. 129-152 (1990).

[4] G. Carpenter, et al. , "ART and ARTMAP Neural Networks for Applications: Self-Organizing Learning, Recognition, and Prediction," Tech Report CAS/CNS-96-009 (1996).

[5] G. Carpenter and S. Grossberg, "Learning, Categorization, Rule Formation, and Prediction by Fuzzy Neural Networks," Tech Report CAS/CNS-94-028 (1994).

[6] G. Carpenter, S. Grossberg and D. Rosen, "Fuzzy ART: An Adaptive Resonance Algorithm for Rapid, Stable Classification of Analog Patterns," Tech Report CAS/CNS-TR-91-006 (1991).

[7] G. Carpenter and M. Gjaja, "Fuzzy ART Choice Functions," CAS/CNS-TR-93-060 (1993).

[8] P. Devijver and J. Kittler, Pattern Recognition A Statistical Approach, Prentice Hall (1982).

[9] D. Dubois and H. Prade, Fuzzy Sets and Systems: Theory and Applications, Academic Press (1980).

[10] L. Franks, Signal Theory, Prentice Hall, Englewood Cliffs, NJ, Ch. 2 (1969).

[11] M.M. Gupta and E. Sanchez, Approximate Reasoning in Decision Analysis, North-Holland (1982).

[12] J. Karakowski, "A Fuzzy Expert System for Real Time Sidelobe Suppression," 24th Asilomar Conference on Signals, Systems, and Computers (1990).

[13] J. Karakowski, "An Automatic Text-Independent Speaker Recognition System," 26th Asilomar Conference on Signals, Systems, and Computers (1992).

[14] J. Karakowski, "Communication Net Sorting: An Automatic Text Independent Speaker Recognition System," US Army CECOM Report CECOM/EW-TR- 92-2 (1992).

[15] J. Karakowski and H. Phu, "Text Independent Speaker Recognition Using a Fuzzy Hypercube Classifier," US Army CECOM Report, CECOM-TR-98-5 (Oct 1998).

[16] J. Karakowski, "Final Report of MMB Signal Processing Unit," US Army CECOM Report, CECOM/EW-TR-90-4 (1990).

[17] T. Kasuba, "Simplified Fuzzy ARTMAP," AI Expert, Nov. 1993.

[18] R. Krishnapuram and J.M. Keller, " A Possibilistic Approach to Clustering," IEEE Trans. Fuzzy Systems, Vol. 1, No. 2 (1993).

[19] Y. Pao, Adaptive Pattern Recognition and Neural Networks, Addison-Wesley Publishing Company, Inc., New York (1989).

[20] S.V. Pellisier, "Open-Set, Text-Independent Speaker Recognition," Thesis, AFIT/GE/ENG/95D, Air Force Institute of Technology, WPAFB, Ohio (1995).

[21] P.K. Simpson, "Fuzzy Min-Max Neural Networks - Part 2: Clustering," IEEE Trans Fuzzy Systems, Vol. 1 No. 1 (Feb 1993).

[22] "Switchboard Speaker Evaluation-Training Segments & Conversation Sides and Test Target & Background Segments," National Institute of Standards and Technology, Gaithersburg, MD (May 1995).

[23] Rome Air Development Center, Greenflag Data Base.

[24] H. Nii, "Rule-Based Understanding of Signals" in Pattern Directed Inference Systems, P. Waterman and F. Hayes-Roth (Eds.), Academic Presss (1978).

[25] G. Bantfai, "An Improved Learning Algorithm for the Fuzzy ARTMAP Neural Network," TR CS-TR-95/10 (1995), Victoria University of Wellington at TechReports@comp.vuw.ac.nz

[26] D. Aha, Machine Learning Database (1993) at ftp.ics.edu/pub/machine-learning-databases.

Appendix A1 - MATLAB Toplevel Code Listings of SFAM

```
%*************************************************
%*   Simplified Fuzzy ArtMap (SFAM) program      *
%*   Programmer:  Hai Phu                        *
%*   Date: 22/08/96                              *
%*                                               *
%*   Input data file:  Iris.dat                  *
%*   Output data file: Out.dat                   *
%*************************************************

clear;
Infile = 'h:\matlab\iriscnvt.dat';
Outfile = 'h:\matlab\sfam\out.dat';

fid_in = fopen(Infile);            % Open a input data file
fid_out  = fopen(Outfile,'w');     % Open an output file

today = date; time = clock;
fprintf(fid_out,'DATE: %s   %2d:%2d\n',today,time(4:5));
fprintf(fid_out,'INPUT DATA FILE: %s \n\n', Infile);

% ******************** Start of program ********************
m = iris_mm(fid_in);
I = 0;
Vigilance = .1
while Vigilance <= .9
 I = I + 1;
 status = fseek(fid_in,0,'eof');    % set indicator to end of file
 eof = ftell(fid_in);
 status = fseek(fid_in,0,'bof');  % reset indicator to beginning of file

% Initialization

 Num_nodes = 0;   pos = 0;
 wrong = 0;
 count = 1;
 Category = 0;

 %******************* Read first data set *******************
 Data  = fscanf(fid_in,'%3f %3f %3f %3f %d /n');
 a = Data(1:4)';
 In_Cat(count) = Data(5) + 1;
 Out_Cat(count) = 1;          %first data is categorized as one
 pos = ftell(fid_in);
 b = scale(a,m);              % scaling the input
 Num_nodes = Num_nodes + 1;    % create an initial output node.
 [Category(Num_nodes) , w(Num_nodes, :)] = fnode(b,Num_nodes);
```

```
%********** Reading input data till end of file ************
while pos < eof
  count = count + 1;
  Data  = fscanf(fid_in,'%3f %3f %3f %3f %d /n');
  a = Data(1:4)';
  In_Cat(count) = Data(5) + 1;
  pos = ftell(fid_in);
  b = scale(a,m);              % scaling the input

% *************** Hypothesize input data *****************
  [node,i] = hypo(Num_nodes,w,b);

% ****************** Test hypothesis **********************
  [w,switch,Out_Cat(count)] = Test_hyp(node,i,w,Vigilance);

 %****************** Create new category ********************

   if ~switch            % if never seen this category before
     Num_nodes = Num_nodes + 1;  % then create an output node.
     [Category(Num_nodes) , w(Num_nodes, :)] = fnode(b,Num_nodes);
     Out_Cat(count) = Num_nodes;
   end;
end                      % end while eof

 %********************* Output routine **********************

fprintf(fid_out,'\nVigilance = %f\n',Vigilance);

[Pc_ac(I) Pc_cc(I) T_Incor(I)] = stat2(In_Cat,Out_Cat,fid_out);
%print out statistics
Values(I) = Vigilance;
Vigilance = Vigilance + .1
clear w Category node;          % clear out all vars for next run


end                             % end while Vigilance


plots(Values,Pc_cc,T_Incor,Pc_ac)
fclose(fid_in);
fclose(fid_out);
```

Appendix A2 - MATLAB Toplevel Code Listings of FANN

```
%*************************************************
%*  Fuzzy Hypercube Artificial Neural Network program   *
%*  Programmer:  Hai Phu                                *
%*  Date: 22/10/96                                      *
%*                                                      *
%*  Input data file:  Iris.dat                          *
%* Output data file: Out.dat                            *
%*************************************************

clear;
Infile = 'h:\matlab\iriscnvt.dat';
Outfile = 'h:\matlab\h_ann1\out.dat';

fid_in = fopen(Infile);              % Open a data file
fid_out  = fopen(Outfile,'w');       % Open an output file


today = date; time = clock;
fprintf(fid_out,'DATE: %s   %2d:%2d\n',today,time(4:5));
fprintf(fid_out,'INPUT DATA FILE: %s \n\n', Infile);


% ******************** Start of program ********************
%********************** User inputs ************************

runs = 0;


Vig = input('Enter a value for vigilance > ');
Gamma = input('Enter a value for gamma > ');

r = input('Enter the Learning Adjustment Factor, r > ');
k = input('Enter a value for k > ');

Hv_max = input('Enter the maximum hypercube volumn > ');



while Hv_max <= 1.5


runs = runs + 1
%************************************************************

m = mm_iris(fid_in);


status = fseek(fid_in,0,'eof');    % set indicator to end of file
eof = ftell(fid_in);
```

```
status = fseek(fid_in,0,'bof');  % reset indicator to beginning of file


% Initialization


Num = 0;
count = 1;


%****************** Read first data set ******************

%read_iris;

Data  = fscanf(fid_in,'%3f %3f %3f %3f %d /n');
a = Data(1:4)';
In_Cat = Data(5) + 1;
pos = ftell(fid_in);

a = scale(a,m);              % scaling the input

Num = Num + 1;     % create an initial output node.
[B(Num,:), Nj(Num)] = anode(a);
Out_Cat(count) = 1;          %first data is categorized as one

%********** Reading input data till end of file ***********

while pos < eof
 count = count + 1;
 Data  = fscanf(fid_in,'%3f %3f %3f %3f %d /n');
 a = Data(1:4)';
 In_Cat(count) = Data(5) + 1;
 pos = ftell(fid_in);

 a = scale(a,m);              % scaling the input


% *************** Transform/Fusion layer ******************

 for category = 1:Num
   R(category) = Trans(B(category,:),a,Nj(category),Gamma,k);
 end;

% ************** Hypothesize and test layer ***************

 % Test an hypothesis;  function test_hyp
 [winner,node] = max(R);
 Adj = Nc(Nj(node))* Vig;
 Out_Cat(count) = node;
 Match = winner > Adj;

% ********************** Learn ************************

 if Match
```

```
      B(node,:) = learn2(a,B(node,:),r);  % Hypercube learning
      Nj(node) = Nj(node) + 1;

      % ********************** Adjust ***********************

      Shv = volume(B(node,:));

      if Shv > Hv_max            % if volume exceeds Hv_max then
        delta = (Shv - Hv_max)/Num
        B(node,:) = Adjust(B(node,:),delta);  % adjust Hypervolume
      end;

      %****************** Create new category ***************

    else                    % if never seen this category before
      Num = Num + 1;            % create an output node.
      [B(Num, :), Nj(Num)] = Anode(a);
      Out_Cat(count) = Num;

    end;

  end                       % end while eof


  %******************** Output routine **********************

  fprintf(fid_out,'\nVigilance = %f\n',Vig);
  fprintf(fid_out,'Gamma = %f ;  k = %f\n',Gamma,k);
  fprintf(fid_out,'Learning adjustment factor r = %f\n',r);
  fprintf(fid_out,'Maximum hypercube volume = %f\n',Hv_max);

 %save parameters for plotting purposes

  [P_ac(runs),P_cc(runs),N_icc(runs)] = stat(In_Cat,Out_Cat,fid_out);

  Values(runs) = Hv_max;
  Hv_max = Hv_max + 0.25;
 clear B category Nj R;          % clear out all vars for next run

end %while

  plots(Values,P_cc,N_icc,P_ac,'Hv_max');

  saveit = input('Do you want to save all working variables ?','s');
  if saveit == 'Y' | saveit == 'y'
    filename = input('Enter filename -> ','s');
    filename = ['h:\matlab\sfam4\' filename];
    save(filename);
  end;

    fclose(fid_in);
  fclose(fid_out);
```

Distribution List

Defense Technical Information Center
ATTN: DTIC-OCC
8725 John J. Kingman Rd., STE 0944
Fort Belvoir, VA 22060-6218
(*Note: 2 DTIC copies will be sent
from STINFO Office, Ft Monmouth, NJ)

Commander, U.S. Army CECOM
R&D Technical Library
Fort Monmouth, NJ 07703-5703
(1) AMSEL-IM-BM-I-L-R (Tech Lib)
(2) AMSEL-IM-BM-I-L-R (STINFO)

Commander, U.S. Army CECOM
Research, Development and Eng. Center
ATTN: AMSEL-RD
Fort Monmouth, NJ 07703-5000

Commander, U.S. Army CECOM
Director, C2SID
ATTN: AMSEL-RD-C2-ED
Fort Monmouth, NJ 07703

Commander, AFRL/CC
1864 4th Street, Suit 1
WPAFB, OH 45433

Director, Rome Research Site
26 Electronic Parkway, Building 106
Rome, NY 13442-4514

Commander, Naval Research Laboratory
Code 1000
Washington DC 20375-5320

Commander, AFIT/CC
2950 P Street
WPAFB, OH 45433-6583