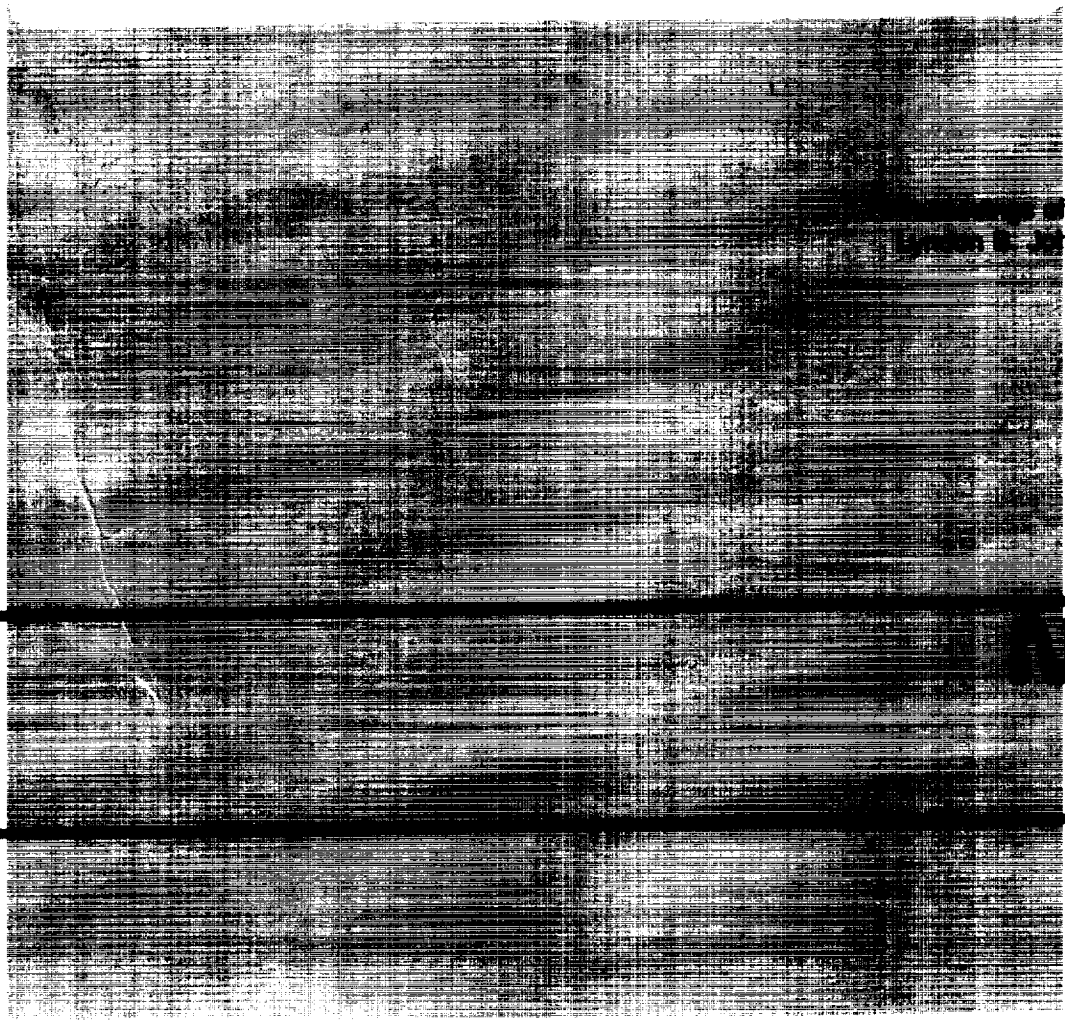
NASA Conference Publication 10111

Proceedings of the Third International Workshop on Neural Networks and Fuzzy Logic

Volume II



a workshop held at
Lyndon B. Johnson Space Center
Houston, Texas
June 1 - 3, 1992

NASA

NASA Conference Publication 10111

Proceedings of the Third International Workshop on Neural Networks and Fuzzy Logic

Volume II

Christopher J. Culbert, Editor
NASA Lyndon B. Johnson Space Center
Houston, Texas

Proceedings of a workshop held at
Lyndon B. Johnson Space Center
Houston, Texas
June 1 - 3, 1992

National Aeronautics and
Space Administration

January 1993

THIRD INTERNATIONAL WORKSHOP ON NEURAL NETWORKS AND FUZZY LOGIC

Program Schedule

Monday June 1, 1992

7:30-8:00 Registration

8:00-8:30 Robert T. Savely, Chief Scientist, Information Systems Directorate, NASA/Lyndon B. Johnson Space Center, Houston, TX. **Welcoming Remarks.**

8:30-9:30 Jon Erickson, Chief Scientist, Automation and Robotics Division, NASA/Lyndon B. Johnson Space Center, Houston, TX. **Space Exploration Needs for Supervised Intelligent Systems.**

9:30-9:45 Break

Plenary Speakers

9:45-10:30 Piero P. Bonnisone, General Electric, **Fuzzy Logic Controllers: A Knowledge-Based Systems Perspective.**

10:30-11:15 Robert Farber, Los Alamos National Laboratory, **Efficiently Modeling Neural Networks on Massively Parallel Computers.**

11:15-1:00 Lunch

1:00-1:30 Lawrence O. Hall and Steve G. Romaniuk, University of South Florida, **Learning Fuzzy Information in a Hybrid Connectionist, Symbolic Model.**

1:30-2:00 Haluk Ogmen, University of Houston, **On the Neural Substrates Leading to the Emergence of Mental Operational Structures.**

2:00-2:30 Hao Ying, University of Texas Medical Branch, **A Fuzzy Controller with Nonlinear Control Rules is the Sum of a Global Nonlinear Controller and a Local Nonlinear PI-like Controller.**

2:30-2:45 Break

Parallel Sessions

2:45-3:15	Ron Maor and Yashvant Jani, Togai Infralogic, **Fuzzy Control of Electric Motors.**	Ronald Yager, Iona College, **A Hierarchical Structure for Representing and Learning Fuzzy Rules.**
3:15-3:45	Andre de Korvin and Margaret F. Shipley, University of Houston–Downtown, **Certain and Possible Rules for Decision Making using Rough Set Theory Extended to Fuzzy Sets.**	Jun Zhou and G. V. S. Raju, The University of Texas at San Antonio, **On Structuring the Rules of a Fuzzy Controller.**
4:15-4:45	Yashvant Jani, Togai Infralogic, Inc., Houston, TX, Gilberto Sousa, University of Tennessee, Knoxville, TN, Wayne Turner, Research Triangle Institute, Research Triangle Park, NC, Ron Spiegel and Jeff Chappell, U.S. EPA, Research Triangle Park, NC, **Fuzzy Efficiency Optimization of AC Induction Motors.**	Robert N. Pap, Mark Atkins, Chadwick Cox, Charles Glover, Ralph Kissel, and Richard Saeks, Accurate Automation Corporation, George C. Marshall Space Flight Center, **Advanced Telerobotic Control Using Neural Networks.**

6:00-7:00 **Wine and Cheese Reception**

7:00-9:00 **Banquet and Keynote Speaker**

Professor Bernard Widrow
Stanford University
Neural Controls

Tuesday, June 2, 1992

Plenary Speakers

8:00-8:45 Tomhiro Takagi, Laboratory for International Fuzzy Engineering Research, **Multilayered Reasoning Based by Means of Conceptual Fuzzy Sets.**

8:45-9:30 Michio Sugeno, Tokyo Institute of Technology, **Fuzzy Control of an Unmanned Helicopter.**

9:30-9:45 Break

9:45-10:15 Porter Sherman, Sikorsky Aircraft, **Fuzzy Logic Mode Switching in Helicopters**

10:15-10:45 James M. Urnes, Stephen E. Hoy, and Robert N. Ladage, McDonnell Aircraft Company, **A Neural Based Intelligent Flight Control System for the NASA F-15 Flight Research Aircraft.**

10:45-11:15 Capt. Gregory Walker, U.S. Army – NASA Langley Research Center, **A Teleoperated Unmanned Rotorcraft Flight Test Technique.**

11:15-11:45 James Villarreal, Robert N. Lea, Yashvant Jani, and Charles Copeland, NASA Lyndon B. Johnson Space Center, **Space Time Neural Network for Tether Operations In Space.**

11:45-1:00	**Lunch**
1:00-1:30	Hamid Berenji, Ames Research Center, **Structure Identification in Fuzzy Inference Using Reinforcement Learning.**
1:30-2:00	J. J. Buckley, University of Alabama at Birmingham, **Approximation Paper: Part 1.**
2:30-3:00	H. VanLangingham, A. Tsoukkas, V. Kreinovich, and C. Quintana, Virginia Polytechnic Institute and State University, University of Texas at El Paso, **Nonlinear Rescaling of Control Values Simplifies Fuzzy Control.**
3:00-3:15	**Break**
3:15-3:45	Robert Elliot Smith, The University of Alabama, **Genetic Learning in Rule-based and Neural Systems.**
3:45-4:15	Manuel Valenzuela-Rendon, Instituto Tecnologico de Estudios Superiores de Monterrey, **Evolving Fuzzy Rules in a Learning Classifier System.**
4:15-4:45	Charles L. Karr, U.S. Department of the Interior – Bureau of Mines, **Adaptive Process Control Using Fuzzy Logic and Genetic Algorithms.**
4:45-5:15	Hideyuki Takagi, University of California at Berkeley, **Design of Fuzzy Systems by Neural Networks and Realization of Adaptability.**
5:15-5:45	Paul P. Wang, Duke University, **Improvement of Fuzzy Controller Design Techniques.**

Wednesday, June 3, 1992

Plenary Speakers

8:00-8:45 Jim Bezdek, University of West Florida, **Two Generalizations of Kohonen Clustering.**

8:00-8:45 Jim Keller, University of Missouri, **Possibilistic Clustering for Shape Description.**

9:30-9:45 Break

Parallel Sessions

9:45-10:15	Todd Espy, Endre Vombrack, and Jack Aldridge, Togai Infralogic, **Application of Genetic Algorithms to Tuning Fuzzy Control Systems.**	Kaoru Hirota, Hosei University – Tokyo, **3D Image Recognition Based on Fuzzy Neural Network Technology.**
10:15-10:45	Isao Hayashi, Elichi Naito, Jun Ozawa, and Noboru Wakami, Matsushita Electric Industrial Co., Ltd., **A Proposal of Fuzzy Connective With Learing Function and its Applications to Fuzzy Retrieval Systems.**	Chul-Sung Kim, Exxon Production Research, **Application of Fuzzy Set and Dempster-Shafer Theory to Organic Geochemistry Interpretation.**
10:45-11:15	Sujeet Shenoi, Chun-Hsin Chen, and Arthur Ramer, University of Tulsa, University of New South Wales, **Towards Autonomous Fuzzy Control.**	Ben Jansen, University of Houston, **Determining the Number of Hidden Units in Multi-layer Perceptrons using F-Ratios.**
11:15-11:45	P.A. Ramamoorthy and Shi Zhang, University of Cincinnati, **A New Approach for Designing Self-Organi zing Systems and Applications to Adaptive Control**	Sunanda Mitra, Texas Tech University, **Adaptive Fuzzy System for 3D Vision.**

11:45-1:00	**Lunch**
1:00-1:30	Michael Murphy, University of Houston – Downtown, **Fuzzy Logic Path Planning System for Collision Avoidance by an Autonomous Rover Vehicle.**
1:30-2:00	Francois G. Pin and Yutaka Watanabe, Oak Ridge National Laboratory, **Driving a Car with Custom-Designed Fuzzy Inferencing VLSI Chips and Boards.**
2:00-2:30	Enrique Ruspini, Stanford Research Institute, **Autonomous Vehicle Motion Control, Approximate Maps, and Fuzzy Logic.**
2:30-2:45	**Break**
2:45-3:15	John Yen and Nathan Pfluger, Texas A&M University, **A Fuzzy Logic Controller for an Autonomous Mobile Robot.**
3:15-3:45	Sandeep Gulati and Raoul Tawel, Jet Propulsion Laboratory, **Intelligent Neuroprocessors for In Situ Propulsion Systems Health Management.**
3:45-4:15	Reza Langari, Texas A&M University, **Synthesis of Non-Linear Control Strategies from Fuzzy Logic Control Algorithms.**
4:15-4:45	P.Z. Wang, Hongmin Zhang, and Wei Xu, National University of Singapore, **Truth-Valued-Flow Inference (TVFI) and Its Applications in Approximate Reasoning.**

CONTENTS

Volume I

Fuzzy Logic Controllers: A Knowledge-Based System Perspective 1

Efficiently Modeling Neural Networks on Massively Parallel Computers 3

Learning Fuzzy Information in a Hybrid Connectionist, Symbolic Model 13

On the Neural Substrates Leading to the Emergence of Mental Operational Structures 30

A Fuzzy Controller with Nonlinear Control Rules Is the Sum of a Global Nonlinear Controller and a Local Nonlinear PI-like Controller 40

Fuzzy Control of Small Servo Motors 48

Hierarchical Structure for Representing and Learning Fuzzy Rules 49

Certain & Possible Rules for Decision Making Using Rough Set Theory Extended to Fuzzy Sets 54

On Structuring the Rules of a Fuzzy Controller 69

Robust Algebraic Image Enhancement for Intelligent Control Systems 73

Design Issues for a Reinforcement-based Self-Learning Fuzzy Controller 75

Fuzzy Efficiency Optimization of AC Induction Motors 76

Advanced Telerobotic Control Using Neural Networks 93

Multi-layered Reasoning by means of Conceptual Fuzzy Sets 95

Fuzzy Control of an Unmanned Helicopter 107

Fuzzy Logic Mode Switching in Helicopters 108

A Neural Based Intelligent Flight Control System for the NASA F-15 Flight Research Aircraft 109

Teleoperated Unmanned Rotorcraft Flight Test Technique 113

Space Time Neural Networks for Tether Operations in Space 127

Structure Identification in Fuzzy Inference Using Reinforcement Learning 169

Approximation Paper: Part I 170

Nonlinear Rescaling of Control Values Simplifies Fuzzy Control 174

Genetic Learning in Rule-based and Neural Systems 183

Evolving Fuzzy Rules in a Learning Classifier System 184

Adaptive Process Control Using Fuzzy Logic and Genetic Algorithms 186

Design of Fuzzy System by NNs and Realization of Adaptability 194

Improvement on Fuzzy Controller Design Techniques 196

Volume II

Two Generalizations of Kohonen Clustering 199

Possibilistic Clustering of Shape Recognition 227

Application of Genetic Algorithms to Tuning Fuzzy Control Systems 237

3D Image Recognition Based on Fuzzy Neural Network Technology ..... 249 -4

A Proposal of Fuzzy Connective with Learning Function and its Application to Fuzzy Retrieval System ..... 257 -5

Application of Fuzzy Set and Dempster-Shafer Theory to Organic Geochemistry Interpretation ..... 273 -6

Towards Autonomous Fuzzy Control ..... 282 -7

Determining the Number of Hidden Units in Multi-Layer Perceptrons using F-Ratios ..... 285 -8

A New Approach for Designing Self-Organizing Systems and Application to Adaptive Control ..... 295 -9

Adaptive Fuzzy System for 3-D Vision ..... 309 -10

Fuzzy Logic Path Planning System for Collision Avoidance by an Autonomous Rover Vehicle ..... 328 -11

Driving a Car with Custom-Designed Fuzzy Inferencing VLSI Chips and Boards ..... 330 -12

Autonomous Vehicle Motion Control, Approximate Maps, and Fuzzy Logic ..... 343 -13

A Fuzzy Logic Controller for an Autonomous Mobile Robot ..... 344 -14

Intelligent Neuroprocessor for In-Situ Launch Vehicle Propulsion Systems Health Management ..... 346 -15

Synthesis of Nonlinear Control Strategies from Fuzzy Logic Control Algorithms ..... 348 -16

Truth-Valued-Flow Inference (TVFI) and Its Applications in Approximate Reasoning 354

TWO GENERALIZATIONS OF KOHONEN CLUSTERING

James C. Bezdek*
Nikhil R. Pal+
Eric C.K.Tsao

Division of Computer Science
The University of West Florida
Pensacola, FL 32514

*Research supported by NSF Grant Number IRI-9003252
+On leave from Indian Statistical Institute, Calcutta

KEYWORDS

Fuzzy Clustering
Fuzzy LVQ
Generalized LVQ
Learning Vector Quantization

ABSTRACT

This paper discusses the relationship between the sequential hard c-means (SHCM), learning vector quantization (LVQ), and fuzzy c-means (FCM) clustering algorithms. LVQ and SHCM suffer from several major problems. For example, they depend heavily on initialization. If the initial values of the cluster centers are outside the convex hull of the input data, such algorithms, even if they terminate, may not produce meaningful results in terms of prototypes for cluster representation. This is due in part to the fact that they update only the winning prototype for every input vector. We also discuss the impact and interaction of these two families with Kohonen's self-organizing feature mapping (SOFM), which is not a clustering method, but which often lends ideas to clustering algorithms. Then we present two generalizations of LVQ that are explicitly designed as clustering algorithms; we refer to these algorithms as generalized LVQ = GLVQ; and fuzzy LVQ = FLVQ. Learning rules are derived to optimize an objective function whose goal is to produce "good clusters". GLVQ/FLVQ (may) update every node in the clustering net for each input vector. Neither GLVQ nor FLVQ depends upon a choice for the update neighborhood or learning rate distribution - these are taken care of automatically. Segmentation of a gray tone image is used as a typical application of these algorithms to illustrate the performance of GLVQ/FLVQ .

1. INTRODUCTION : LABEL VECTORS AND CLUSTERING

Clustering algorithms attempt to organize unlabeled feature vectors into clusters or "natural groups" such that points within a cluster are more similar to each other than to vectors belonging to different clusters. Treatments of many classical approaches to this problem include the texts by Kohonen[1], Bezdek[2], Duda and Hart[3], Tou and Gonzalez[4], Hartigan[5], and Dubes and Jain[6]. Kohonen's work has become timely in recent years because of the widespread resurgence of interest in the theory and applications of neural network structures [7].

Label Vectors. To characterize solution spaces for clustering and classifier design, let c denote the number of clusters, $1 < c < n$, and set :

$$N_{fcu} = \{\mathbf{y} \in \Re^c \mid y_k \in [0, 1] \ \forall\ k\} \quad = \text{(unconstrained) } \textit{fuzzy labels} \quad ; \quad (1a)$$

$$N_{fc} = \{\mathbf{y} \in N_{fcu} \mid \Sigma y_k = 1\} \quad = \text{(constrained) } \textit{fuzzy labels} \quad ; \quad (1b)$$

$$N_c = \{\mathbf{y} \in N_{fc} \mid y_k \in \{0, 1\} \ \forall\ k\} \quad = \textit{hard labels} \text{ for c classes} \quad . \quad (1c)$$

N_c is the canonical basis of Euclidean c-space; N_{fc} is its convex hull; and N_{fcu} is the unit hypercube in $\Re^c$. Figure 1 depicts these sets for c=3. For example, the vector $\mathbf{y} = (.1, .6, .3)^T$ is a typical constrained fuzzy label vector; its entries lie between 0 and 1, and sum to 1. And because its entries sum to 1, $\mathbf{y}$ may also be interpreted as a *probabilistic* label. The cube $N_{fcu} = [0, 1]^3$ is called *unconstrained* fuzzy label vector space; vectors such as $\mathbf{z} = (.7, .2, .7)^T$ have each entry between 0 and 1, but are otherwise unrestricted.

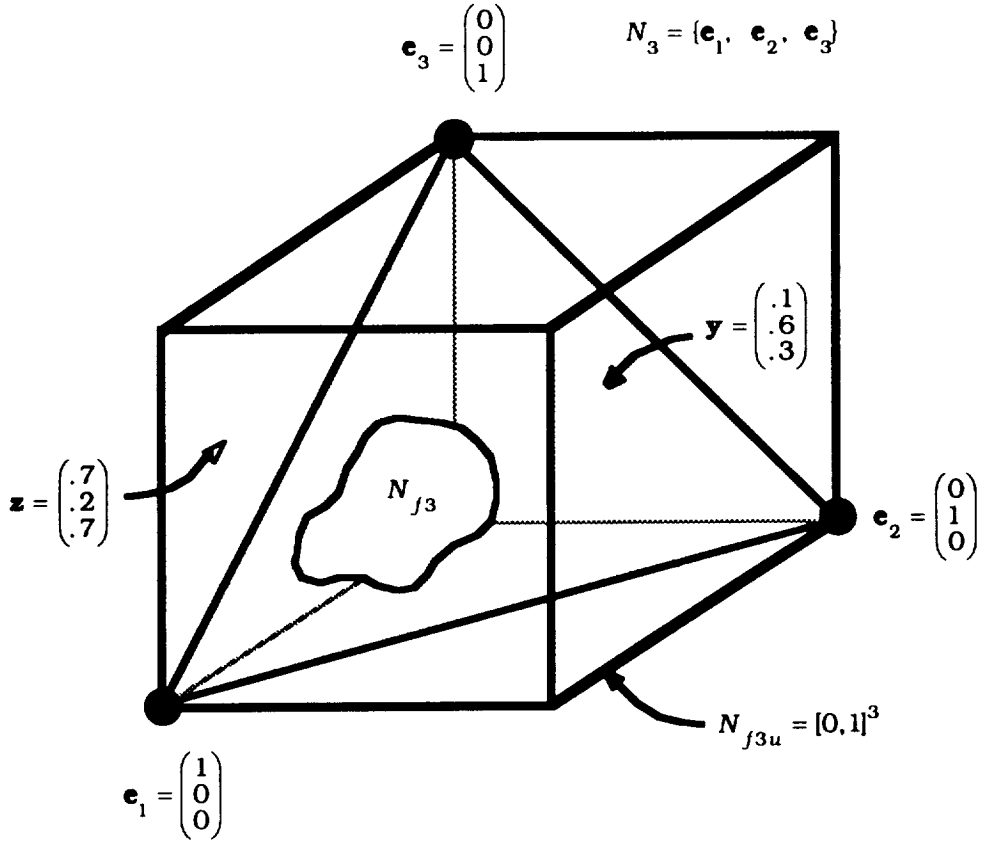
Cluster Analysis. Given unlabeled data $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ in $\Re^p$, *clustering* in X is assignment of (hard *or* fuzzy) label vectors to the objects generating X. If the labels are hard, we hope that they identify c "natural subgroups" in X. Clustering is also called *unsupervised learning* , the word *learning* referring here to learning the correct labels (and possibly vector prototypes or quantizers) for "good" subgroups in the data. *c-partitions* of X are characterized as sets of (cn) values $\{u_{ik}\}$ satisfying some or all of the following conditions :

$$0 \le u_{ik} \le 1 \qquad \forall\ i,k \quad ; \quad (2a)$$

$$0 < \Sigma u_{ik} < n \qquad \forall\ i \quad ; \quad (2b)$$

$$\Sigma u_{ik} = 1 \qquad \forall\ k \quad . \quad (2c)$$

Fig. 1. Hard, fuzzy and probabilistic label vectors (for c = 3 classes).



Using equations (2) with the values $\{u_{ik}\}$ arrayed as a (cxn) matrix $U = [u_{ik}]$, we define:

$$M_{fcnu} = \{U \in \Re^{cn} \mid u_{ik} \text{ satisfies (2a) and (2b) } \forall \text{ i, k}\} \quad ; \quad \text{(3a)}$$

$$M_{fcn} = \{U \in M_{fcnu} \mid u_{ik} \text{ satisfies (2c) } \forall \text{ i and k}\}. \quad ; \quad \text{(3b)}$$

$$M_{cn} = \{U \in M_{fcn} \mid u_{ik} = 0 \text{ or } 1 \ \forall \text{ i and k}\} \quad . \quad \text{(3c)}$$

Equations (3a), (3b) and (3c) define, respectively, the sets of unconstrained fuzzy, constrained fuzzy (or probabilistic), and crisp *c-partitions of X*. We represent clustering algorithms as mappings $\mathcal{A} : X \rightarrow M_{fcnu}$. Each column of U in M_{fcnu} (M_{fcn}, M_{cn}) is a label vector from N_{fcu} (N_{fc} , N_c). The reason these matrices are called *partitions* follows from the interpretation of u_{ik} as the *membership* of $\mathbf{x}_k$ in the i-th partitioning subset (cluster) of X. M_{fcnu} and M_{fcn} can be more realistic physical models than M_{cn}, for it is common experience that the boundaries between many classes of real objects (e.g., tissue types in magnetic resonance images) are in fact very badly delineated (i.e., really fuzzy) , so M_{fcnu} provides a much richer means for

representing and manipulating data that have such structures. We give an example to illustrate hard and fuzzy c-partitions of X. Let $X = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$ = {peach, plum, nectarine}, and let c=2. Typical 2-partitions of these three objects are shown in Table 1:

Table 1. 2-partitions of $X = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$ = {peach, plum, nectarine}

	Hard $U_1 \in M_{23}$	Fuzzy $U_2 \in M_{f23}$	Fuzzy $U_3 \in M_{f23u}$
Object	$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$	$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$	$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$
Peaches	$\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.9 & 0.2 & 0.4 \end{bmatrix}$	$\begin{bmatrix} 0.9 & 0.5 & 0.5 \end{bmatrix}$
Plums	$\begin{bmatrix} 0 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 0.1 & 0.8 & 0.6 \end{bmatrix}$	$\begin{bmatrix} 0.6 & 0.8 & 0.7 \end{bmatrix}$

The nectarine, $\mathbf{x}_3$, is shown as the last column of each partition, and in the hard case, it must be (erroneously) given full membership in one of the two crisp subsets partitioning this data; in U_1 $\mathbf{x}_3$ is labeled "plum". Fuzzy partitions enable algorithms to (sometimes!) avoid such mistakes. The final column of the first fuzzy partition in Table 1 allocates most (0.6) of the membership of $\mathbf{x}_3$ to the plums class; but also assigns a lesser membership of 0.4 to $\mathbf{x}_3$ as a peach. The last partition in Table 1 illustrates an unconstrained set of membership assignments for the objects in each class. Columns like the one for the nectarine in the two fuzzy partitions serve a useful purpose - lack of strong membership in a single class is a signal to "take a second look". Hard partitions of data cannot suggest this. In the present case, the nectarine is an *hybrid* of peaches and plums, and the memberships shown for it in the last column of either fuzzy partition seem more plausible *physically* than crisp assignment of $\mathbf{x}_3$ to an incorrect class. It is appropriate to note that statistical clustering algorithms - e.g., unsupervised learning with maximum likelihood - also produce solutions in M_{fcn}. Fuzzy clustering began with Ruspini[8] ; see Bezdek and Pal[9] for a number of more recent papers on this topic. Algorithms that produce unconstrained fuzzy partitions of X are relatively new; for example, see the work of Krishnapuram and Keller[10].

Prototype classification is illustrated in Figure 2. Basically, the vector $\mathbf{v}_i$ is taken as a prototypical representation for all the vectors in the hard cluster $X_i \subset X$. There are many synonyms for the word prototype in the literature: for example, quantizer (hence LVQ), signature, template, paradigm, exemplar. In the context of clustering, of course, we view $\mathbf{v}_i$ as the cluster center of hard cluster $X_i \subset X$. Each of the clustering algorithms discussed in this paper will produce a set of c prototype vectors $V = \{\mathbf{v}_k\}$ from any unlabeled or labeled input data

set X in $\Re^p$. Once the prototypes are found (and possibly relabeled if the data have physical labels), they define a hard nearest prototype (NP) classifier, say $\mathbf{D}_{NP,V}$:

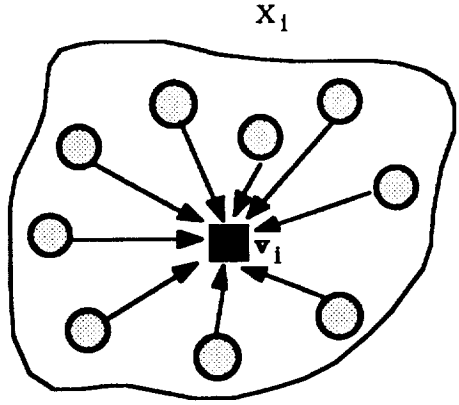
Crisp Nearest Prototype (1-NP) Classifier. Given prototypes $V = \{\mathbf{v}_k \mid 1 \le k \le c\}$ and $\mathbf{z} \in \Re^p$:

$$\text{Decide } \mathbf{z} \in i \Leftrightarrow \mathbf{D}_{NP,V}(\mathbf{z}) = \mathbf{e}_i \quad \Leftrightarrow \quad \|\mathbf{z} - \mathbf{v}_i\|_A \le \|\mathbf{z} - \mathbf{v}_j\|_A : 1 \le j \le c, j \ne i \tag{4}$$

In (4) A is any *positive definite* pxp weight matrix - it renders the norm in (4) an inner product norm. That is, the distance from $\mathbf{z}$ to any $\mathbf{v}_i$ is computed as $\|\mathbf{z} - \mathbf{v}_i\|_A = \sqrt{(\mathbf{z} - \mathbf{v}_i)^T A (\mathbf{z} - \mathbf{v}_i)}$.

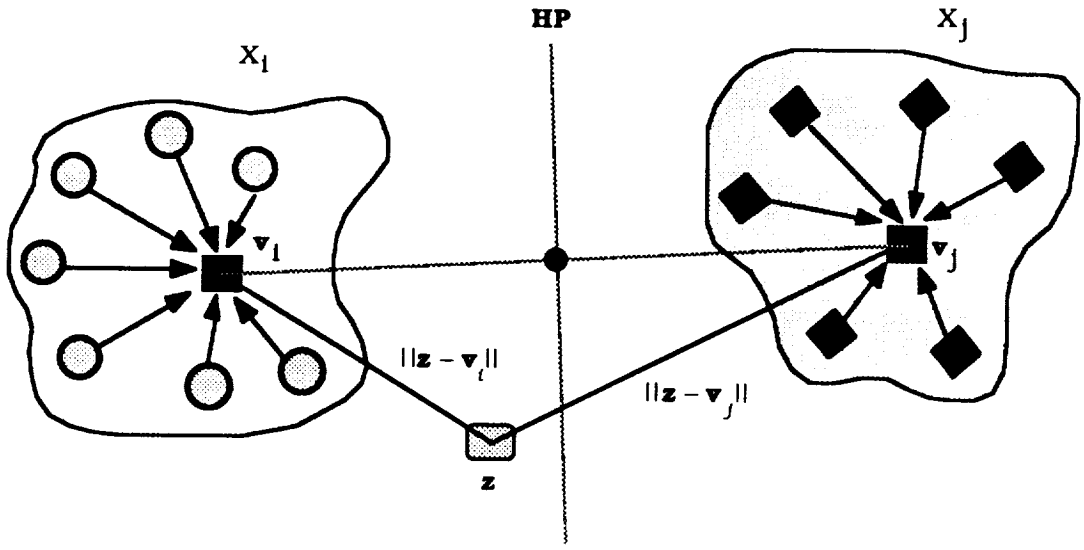
Equation (4) defines a hard classifier, even though its parameters may come from a fuzzy algorithm. It would be careless to call $\mathbf{D}_{NP,V}$ a fuzzy classifier just because fuzzy c-means produced the prototypes, for example, because (4) can be implemented, and has the same geometric structure, using prototypes $\{\mathbf{v}_k\}$ from *any* algorithm that produces them. The $\{\mathbf{v}_k\}$ can be sample means of hard clusters (HCM); cluster centers of fuzzy clusters (FCM); weight vectors attached to the nodes in the competitive layer of a Kohonen clustering network (LVQ); or estimates of the (c) assumed mean vectors $\{\mu_k\}$ in maximum likelihood decomposition of mixtures.

Figure 2. Representation of many vectors by one prototype (vector quantizer).



The geometry of the 1-NP classifier is shown in Figure 3, using Euclidean distance for (4) - that is A=I, the pxp identity matrix. The 1-NP design erects a linear boundary halfway between and orthogonal to the line connecting the i-th and j-th prototypes, viz., the hyperplane HP through the vector $(\mathbf{v}_i - \mathbf{v}_j)/2$ perpendicular to it. All NP designs defined with inner product norms use (piecewise) linear decision boundaries of this kind.

Figure 3. Geometry of the Nearest Prototype Classifier for Inner Product Norms



Clustering algorithms imaged in M_{fcnu} eventually "defuzzify" or "deprobabilize" their label vectors, usually using the maximum membership (or maximum probability) strategy on the terminal fuzzy (or probabilistic) c-partitions produced by the data:

***Maximum membership* (MM) conversion** of U in M_{fcnu} to U_{MM} in M_{fc} :

$$u_{MM_{ik}} = \begin{cases} 1; & u_{ik} \geq u_{sk},\ 1 \leq s \leq c, s \neq i \\ 0; & \textit{otherwise} \end{cases} \qquad 1 \leq i \leq c;\ 1 \leq k \leq n \quad . \qquad (5)$$

U_{MM} is always a hard c-partition; we use this conversion to generate a confusion matrix and error statistics when processing labeled data with FCM and FLVQ. For HCM/FCM/LVQ/FLVQ, using (5) instead of (4) with the terminal prototypes secured is fully equivalent- that is, U_{MM} *is* the hard partition that would be created by applying (5) with the final cluster centers to the unlabeled data. This is not true for GLVQ.

2. LEARNING VECTOR QUANTIZATION AND SEQUENTIAL HARD C-MEANS

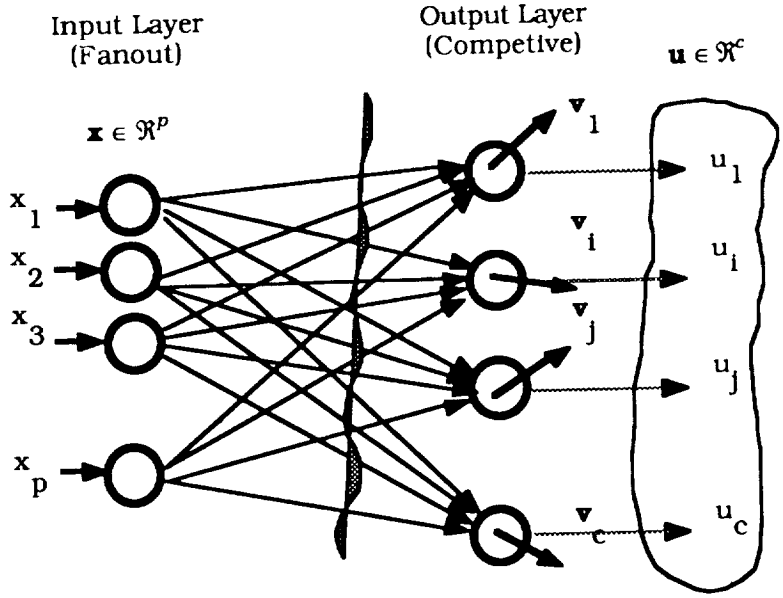
Kohonen's name is associated with two very different, widely studied and often confused families of algorithms. Specifically, Kohonen initiated study of the prototype generation algorithm called *learning vector quantization* (LVQ); and he also introduced the concept of *self-organizing feature maps* (SOFM) for visual display of certain one and two dimensional

data sets[1]. LVQ is not a clustering algorithm per se; rather, it can be used to generate crisp (conventional or hard) c-partitions of unlabeled data sets in using the 1-NP classifier designed with its terminal prototypes. LVQ is applicable to p dimensional unlabeled data. SOFM, on the other hand, attempts to find topological structure hidden in data and display it in one or two dimensions.

We shall review LVQ and its c-means relative carefully, and SOFM in sufficient detail to understand its intervention in the development of generalized network clustering algorithms. The primary goal of LVQ is representation of many points by a few prototypes; identification of clusters is implicit, but not active, in pursuit of this goal. We let $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n\} \subset \Re^p$ denote the samples at hand, and use c to denote the number of nodes (and clusters in X) in the competitive layer.

The salient features of the LVQ model are contained in Figure 5. The input layer of an LVQ network is connected directly to the output layer. Each node in the output layer has a weight vector (or prototype) attached to it. The prototypes $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c)$ are essentially a network array of (unknown) cluster centers, $\mathbf{v}_i \in \Re^p$ for $1 \le i \le c$. In this context the word *learning* refers to finding values for the $\{v_{ij}\}$. When an input vector $\mathbf{x}$ is submitted to this network, distances are computed between each $\mathbf{v}_r$ and $\mathbf{x}$. The output nodes "compete", a (minimum distance) "winner" node , say $\mathbf{v}_i$, is found ; and it is then updated using one of several update rules.

Figure 5. LVQ Clustering Networks



We give a brief specification of LVQ as applied to the data in our examples. There are other versions of LVQ; this one is usually regarded as the "standard" form.

The LVQ Clustering Algorithm[1]

LVQ1. Given unlabeled data set $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n\} \subset \Re^p$. Fix c, T, and $\varepsilon > 0$.

LVQ2. Initialize $\mathbf{V}_0 = (\mathbf{v}_{1,0}, \ldots, \mathbf{v}_{c,0}) \in \Re^{cp}$, and learning rate $\alpha_0 \in (1,0)$.

LVQ3. For t = 1,2, ..., T;

For k = 1,2, ..., n:

a. Find $\|\mathbf{x}_k - \mathbf{v}_{i,t-1}\| = \min_{1 \le j \le c} \{\|\mathbf{x}_k - \mathbf{v}_{j,t-1}\|\}$. (6)

b. Update the winner : $\mathbf{v}_{i,t} = \mathbf{v}_{i,t-1} + \alpha_t(\mathbf{x}_k - \mathbf{v}_{i,t-1})$ (7)

Next k.

d. Apply the 1-NP (nearest prototype) rule to the data :

$$u_{LVQ_{ik}} = \begin{cases} 1; & \|\mathbf{x}_k - \mathbf{v}_i\| \le \|\mathbf{x}_k - \mathbf{v}_j\|, 1 \le j \le c, j \ne i \\ 0; & \text{otherwise} \end{cases}, 1 \le i \le c \text{ and } 1 \le k \le n. \quad (8)$$

e. Compute $E_t = \|\mathbf{V}_t - \mathbf{V}_{t-1}\|_1 = \sum_{r=1}^{c} \|\mathbf{v}_{r,t} - \mathbf{v}_{r,t-1}\|_1 = \sum_{k=1}^{n} \sum_{r=1}^{c} |v_{rk,t} - v_{rk,t-1}|$.

f. If $E_t \le \varepsilon$ stop; Else adjust learning rate α_t;

Next t

The numbers $U_{LVQ} = [u_{LVQ_{ik}}]$ at (8) are a cxn matrix that define a hard c-partition of X using the 1-NP classifier assignment rule shown in (4). The vector **u** shown in Figure 1 represents a crisp *label vector* that corresponds to one column of this matrix; it contains a 1 in the winner row i at each k; and zeroes otherwise. Our inclusion of the computation of the hard 1-NP c-partition of X at the end of each pass through the data (step LVQ3.d) is **not** part of the LVQ algorithm - that is, the LVQ iterate sequence does not depend on cycling through U's. Ordinarily this computation is done once, non-iteratively, outside and after termination of LVQ. Note that LVQ uses the Euclidean distance in step LVQ3.a. This choice corresponds roughly to the update rule shown in (7), since $\nabla_{\mathbf{v}}(\|\mathbf{x} - \mathbf{v}\|_I^2) = -2I(\mathbf{x} - \mathbf{v}) = -2(\mathbf{x} - \mathbf{v})$. The origin of this rule comes about by assuming that each $\mathbf{x} \in \Re^p$ is distributed according to a probability density function $f(\mathbf{x})$. LVQ's objective is to find a set of $\mathbf{v}_i$'s such that the expected value of the square of the discretization error is minimized :

$$E\left(\left\|\mathbf{x}-\mathbf{v}_i\right\|^2\right)=\int\int\ldots\int_{\Re^p}\left\|\mathbf{x}-\mathbf{v}_i\right\|^2 f(\mathbf{x})d\mathbf{x} \quad . \tag{9}$$

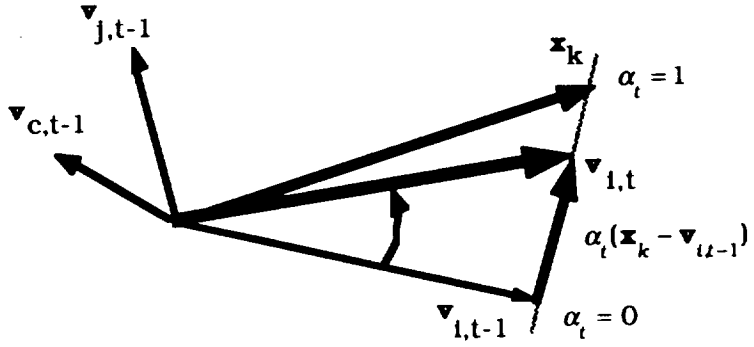
In this expression $\mathbf{v}_i$ is the winning prototype for each $\mathbf{x}$, and will of course vary as $\mathbf{x}$ ranges over $\Re^p$. A sample function of the optimization problem is $e=\left\|\mathbf{x}-\mathbf{v}_i\right\|^2$. An optimal set of $\mathbf{v}_i$'s can be approximated by applying local gradient descent to a finite set of samples drawn from f. The extant theory for this scheme is contained in Kohonen[12], which states that LVQ converges in the sense that the prototypes $\mathbf{V}_t = (\mathbf{v}_{1,t}, \mathbf{v}_{2,t}, \ldots, \mathbf{v}_{c,t})$ generated by the LVQ iterate sequence converge, i.e., $\{\mathbf{V}_t\} \xrightarrow{t\to\infty} \hat{\mathbf{V}}$, provided two conditions are met by the sequence $\{\alpha_t\}$ of learning rates used in (7) :

$$\sum_{t=0}^{\infty}\alpha_t=\infty \quad : \quad \text{and} \tag{10a}$$

$$\sum_{t=0}^{\infty}\alpha_t^2<\infty \quad . \tag{10b}$$

One choice for the learning rates that satisfies these conditions is the harmonic sequence $\alpha_t = 1/t$ for $t \geq 1$; $\alpha_0 \in (0,1)$. Kohonen has shown that (under some assumptions) steepest descent optimization of the average expected error function (9) is possible, and leads to the update rule (7). The update scheme shown in equation (7) has the simple geometric interpretation shown in Figure 6.

Figure 6. Updating the winning LVQ Prototype.



The winning prototype $\mathbf{v}_{i,t-1}$ is simply rotated towards the current data point by moving along the vector $(\mathbf{x}_k - \mathbf{v}_{i,t-1})$ which connects it to $\mathbf{x}_k$. The amount of shift depends on the value of a "learning rate" parameter α_t, which varies from 0 to 1. As seen in Figure 2, there is no update if $\alpha_t=0$, and when $\alpha_t=1$, $\mathbf{v}_{i,t}$ becomes $\mathbf{x}_k$ ($\mathbf{v}_{i,t}$ is just a convex combination of $\mathbf{x}_k$ and $\mathbf{v}_{i,t-1}$). This process continues until termination via LVQ3.f, at which time the terminal prototypes yield a "best" hard c-partition of X via (3).

Comments on LVQ :

1. Limit point property : Kohonen[12] refers to [13,14], and mentions that LVQ converges to a unique limit if and only if conditions (10) are satisfied. However, nothing was said about what sort or *type* of points the final weight vectors produced by LVQ are. Since LVQ does not model a well defined property of clusters (in fact, LVQ does not maintain a partition of the data at all), the fact that $\{\mathbf{V}_t\} \xrightarrow{t\to\infty} \hat{\mathbf{V}}$ does not insure that the limit vector $\hat{\mathbf{V}}$ is a good set of prototypes in the sense of representation of clusters or clustering tendencies. All the theorem guarantees is that the sequence HAS a limit point. Thus, "good clusters" in $\mathbf{X}$ will result by applying the 1-NP rule to the final LVQ prototypes only if, by chance, these prototypes are good class representatives. In other words, the LVQ model is not *driven* by a well specified clustering goal.

2. **Learning rate** α : Different strategies for α_t often produce different results. Moreover, LVQ seldom terminates unless $\alpha_t \to 0$ (i.e., it is *forced* to stop because successive iterates are necessarily close).

3. **Termination** : LVQ often runs to its iterate limit, and actually passes the optimal (clustering) solution in terms of minimal apparent label error rate. This is called the "over-training" phenomenon in the neural network literature.

Another, older, clustering approach that is often associated with LVQ is *sequential hard c-means* (SHCM). The updating rule of MacQueen's SHCM algorithm is similar to LVQ[15]. In MacQueen's algorithm the weight vectors are initialized with the first c samples in the data set X. In other words, $\mathbf{v}_{r,0} = \mathbf{x}_r$, r=1,...,c. Let $q_{r,0}=1$ for r=1,...,c ($q_{r,t}$ represents the number of samples that have so far been used to update $\mathbf{v}_{r,t}$). Suppose $\mathbf{x}_{t+1}$ is a new sample point such that $\mathbf{v}_{i,t}$ is closest (with respect to, and without loss, the Euclidean metric) to it. MacQueen's algorithm updates the $\mathbf{v}_r$'s as follows (again, index i identifies the winner at this t):

$$\mathbf{v}_{i,t+1} = (\mathbf{v}_{i,t}\ q_{i,t} + \mathbf{x}_{t+1})/(q_{i,t}+1) \quad ; \quad (11a)$$

$$q_{i,t+1} = q_{i,t} + 1 \quad ; \quad (11b)$$

$$\mathbf{v}_{r,t+1} = \mathbf{v}_{r,t} \quad \text{for } r \neq i, \quad ; \quad (11c)$$

$$q_{r,t+1} = q_{r,t} \quad \text{for } r \neq i. \quad . \quad (11d)$$

MacQueen's process terminates when all the samples have been used once (i.e., when t = n). The sample points are then labeled on the basis of nearness to the final mean vectors (that is, using (3) to find a hard c-partition U_{SHCM}). Rearranging (11a), one can rewrite Macqueen's update equation :

$$\mathbf{v}_{i,t+1} = \mathbf{v}_{i,t} + (\mathbf{x}_{t+1} - \mathbf{v}_{i,t}) / q_{i,t+1} . \tag{12}$$

Writing $1/q_{i,t+1}$ as $\alpha_{i,t+1}$, equation (12) takes exactly the same form as equation (7) . However, there are some differences between LVQ and MacQueen's algorithm: (i) In LVQ sample points are used repeatedly until termination is achieved, while in MacQueen's method sample points are used only once (other variants of this algorithm pass through the data set many times[16]. (ii) In MacQueen's algorithm $\alpha_{i,t+1}$ is inversely proportional to the number of points found closest to $\mathbf{v}_{i,t}$, so it is possible to have $\alpha_{i,t_1} < \alpha_{j,t_2}$ when $t_1 > t_2$. This is not possible in LVQ. MacQueen attempted to partition feature space $\Re^p$ into c subregions, say $(S_1,....,S_c)$, in such a way as to minimize the functional

$$J_M(\mathbf{x}, \hat{\mathbf{V}}) = \sum_{i=1}^{c} \int\int\ldots\int_{\Re^p} \left\| \mathbf{x} - \hat{\mathbf{v}}_i \right\|^2 df(\mathbf{x}),$$

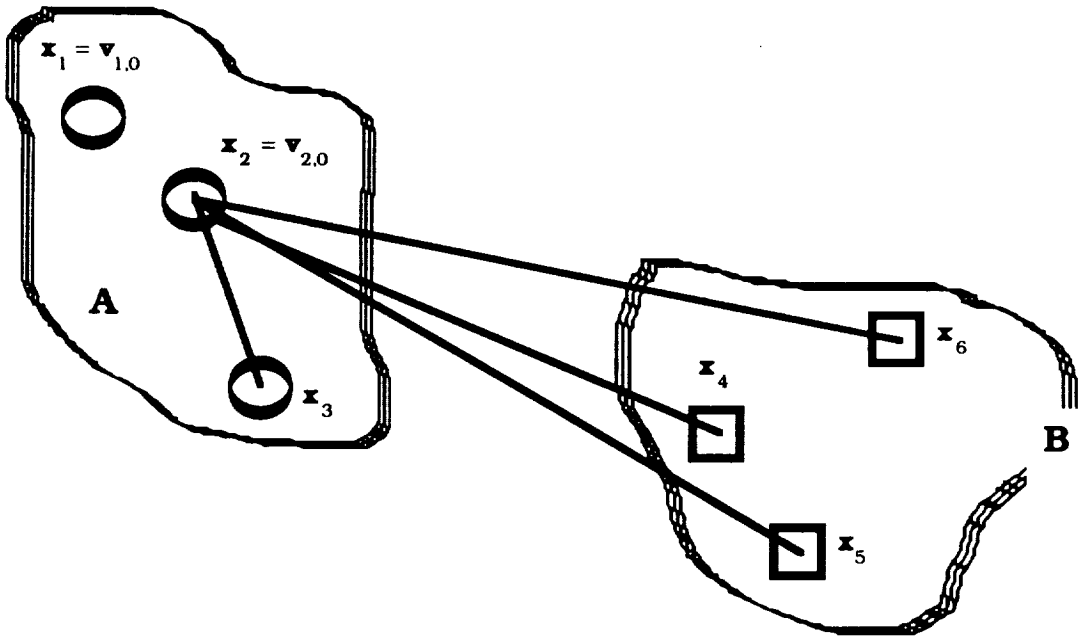
where f is a density function as in LVQ, and $\hat{\mathbf{v}}_i$ is the (conditional) mean of the pdf f_i obtained by restricting f to S_i, normalized in the usual way, i.e., $f_i(\mathbf{x}) = f(\mathbf{x})|_{S_i} / P(S_i)$; and $\hat{\mathbf{V}} = (\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2,\ldots, \hat{\mathbf{v}}_c) \in \Re^{cp}$. Let $\mathbf{V}_t = (\mathbf{v}_{1,t},\ldots,\mathbf{v}_{c,t})$; $S_t = (S_1(\mathbf{v}_t),\ldots,S_c(\mathbf{v}_t))$ be the minimum distance partition relative to $\mathbf{v}_t$; $P(S_j) = \text{prob}(\mathbf{x} \in S_j)$, $P_{j,t} = P(S_j(\mathbf{v}_t)) = \text{prob}(\mathbf{x} \in S_j(\mathbf{v}_t))$; and $\hat{\mathbf{v}}_{j,t}$, the conditional mean of $\mathbf{x}$ over $S_j(\mathbf{v}_t)$, is $\hat{\mathbf{v}}_{j,t} = \int_{S_j(\mathbf{v}_t)} \mathbf{x} df(\mathbf{x}) / P(S_j)$ when $P(S_j) > 0$, or $\hat{\mathbf{v}}_{j,t} = \mathbf{v}_{j,t}$ when $P(S_j) = 0$. MacQeen proved that for the algorithm described by equations (11a-d) ,

$$\lim_{n\to\infty} \left\{ \frac{\sum_{t=1}^{n} \left(\sum_{j=1}^{c} P_{j,t} \left\| \mathbf{v}_{j,t} - \hat{\mathbf{v}}_{j,t} \right\| \right)}{n} \right\} = 0 \quad .$$

Since $\{ \hat{\mathbf{v}}_j \}$ are conditional means, the partition obtained by applying the nearest prototype labeling method at (4) to them may not always be desirable from the point of view of clustering. Moreover, this result does not eliminate the possibility of slow but indefinite oscillation of the centroids (limit cycles).

LVQ and SHCM suffer from a common problem that can be quite serious. Suppose the input data $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6\} \subset \Re^2$ contains the two classes $\mathbf{A} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$ and $\mathbf{B} = \{\mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6\}$ as shown in Figure 7. The initial positions of the centroids $\mathbf{v}_{1,0}$ and $\mathbf{v}_{2,0}$ are also depicted in Figure 7. Since the initial centroid for class 2 ($\mathbf{v}_{2,0}$) is closer to the remaining four input points than $\mathbf{v}_1$, each of them will update (modify) $\mathbf{v}_2$ only; $\mathbf{v}_1$ will not be changed on the first pass through the data. Moreover, both update schemes result in the updated centroid being pulled towards the data point some distance along the line joining the two points. Consequently, the chance for $\mathbf{v}_{1,0}$ to get updated on succeeding passes is very low. Although this results in a locally optimal solution, it is hardly a desirable one.

Figure 7. An initialization problem for LVQ/SHCM



There are two causes for this problem : (i) an improper choice of the initial centroids, and (ii) each input updates only the winner node. To circumvent problem (i), initialization of the $\mathbf{v}_i$'s is often done with random input vectors; this reduces the probability of occurrence of the above situation, but does not eliminate it. Bezdek et. al[17] attempted to solve problem (ii) by updating the winner and some of its neighbors (not topological, but metrical neighbors in $\Re^p$) with each input in FLVQ. In their approach, the learning coefficient was reduced both with time and distance from the winner. FLVQ, in turn, raised general two issues : defining an appropriate

neighborhood system, and deciding on strategies to reduce the learning coefficient with distance from the winner node. These two issues motivated the development of the GLVQ algorithm.

We conclude this section with a brief description of the SOFM scheme, again using t to stand for iterate number (or time). In this algorithm each prototype $\mathbf{v}_{r,t} \in \Re^p$ is associated with a display node ,say $\mathbf{d}_{r,t} \in \Re^2$. The vector $\mathbf{v}_{i,t}$ that best matches (in the sense of minimum Euclidean distance in the feature space) an incoming input vector $\mathbf{x}_k$ is then identified as in (4). $\mathbf{v}_{i,t}$ has an "image" $\mathbf{d}_{i,t}$ in display space. Next, a topological (spatial) neighborhood $\mathcal{N}(\mathbf{d}_{i,t})$ centered at $\mathbf{d}_{i,t}$ is defined in display space, and its display node neighbors are located. Finally, the vector $\mathbf{v}_{i,t}$ and other prototype vectors in the inverse image $[\mathcal{N}(\mathbf{d}_{i,t})]^{-1}$ of spatial neighborhood $\mathcal{N}(\mathbf{d}_{i,t})$ are updated using a generalized form of update rule (7) :

$$\mathbf{v}_{r,t} = \mathbf{v}_{r,t-1} + \alpha_{rk,t}(\mathbf{x}_k - \mathbf{v}_{r,t-1}), \qquad \mathbf{d}_{r,t} \in \mathcal{N}(\mathbf{d}_{i,t}). \tag{13}$$

The function $\alpha_{rk,t}$ defines a *learning rate distribution* on indices (r) of the nodes to be updated for each input vector $\mathbf{x}_k$ at each iterate t. These numbers *impose* (by their definition) a sense of the strength of interaction between (output) nodes. If the $\{\mathbf{v}_{r,t}\}$ are initialized with random values and the external inputs $\mathbf{x}_k = \mathbf{x}_k(t)$ are drawn from a time invariant probability density function $f(\mathbf{x})$, then the point density function of $\mathbf{v}_{r,t}$ (the number of $\mathbf{v}_{r,t}$'s in the ball $\mathbf{B}(\mathbf{x}_k, \varepsilon)$ centered at the point $\mathbf{x}_k$with radius ε) tends to approximate $f(\mathbf{x})$. It has also been shown that the $\mathbf{v}_{r,t}$'s attain their values in an "orderly fashion" according to $f(\mathbf{x})^{12}$. This process is continued until the weight vectors "stabilize." In this method then, a learning rate distribution over time and spatial neighborhoods must be defined which decreases with time in order to force termination (to make $\alpha_{rk,t}$ =0). The update neighborhood also decreases with time. While this is clearly not a clustering strategy, the central tendency property of the prototypes often tempts users to assume that terminal weight vectors offer compact representation to clusters of feature vectors; in practice, this is often false.

4. GENERALIZED LEARNING VECTOR QUANTIZATION (GLVQ)

In this section we describe a new clustering algorithm which avoids or fixes several of the limitations mentioned earlier. The learning rules are derived from an optimization problem. Let $\mathbf{x} \in \Re^p$ be a stochastic input vector distributed according to a time invariant probability distribution $f(\mathbf{x})$, and let i be the best matching node as in (7). Let $L_{\mathbf{x}}$ be a loss function which measures the locally weighted mismatch (error) of $\mathbf{x}$ with respect to the winner :

$$L_{\mathbf{x}} = L(\mathbf{x} \; ; \; \mathbf{v}_1,\ldots,\mathbf{v}_c) = \sum_{r=1}^{c} g_{ir} \left\| \mathbf{x} - \mathbf{v}_r \right\|^2 \quad , \text{where} \tag{14a}$$

$$g_{ir} = \left\{ \begin{array}{ll} 1 & \text{if} \quad r = i \\ \dfrac{1}{\sum_{j=1}^{c} \left\| \mathbf{x} - \mathbf{v}_j \right\|^2} & , \; \text{otherwise} \end{array} \right\} . \tag{14b}$$

Let $\mathbf{X} = \{\mathbf{x}_1,\ldots, \mathbf{x}_n, \ldots\}$ be a set of samples from $f(\mathbf{x})$ drawn at time instants t=1,2,...,n, ... Our objective is to find a set of c $\mathbf{v}_r$'s , say $\mathbf{V} = \{\mathbf{v}_r\}$ such that the locally weighted error functional $L_{\mathbf{x}}$ defined with respect to the winner $\mathbf{v}_i$ is minimized over $\mathbf{X}$. In other words, we seek to

$$\text{Minimize} \; : \quad \Gamma(\mathbf{V}) = \int\int\ldots\int_{\Re^p} \sum_{r=1}^{c} g_{ir} \left\| \mathbf{x} - \mathbf{v}_r \right\|^2 f(\mathbf{x}) d\mathbf{x} \tag{15}$$

For a fixed set of points $\mathbf{X} = \{\mathbf{x}_1,\ldots, \mathbf{x}_n\}$ the problem reduces to the unconstrained optimization problem:

$$\text{Minimize} \; : \quad \Gamma(\mathbf{V}) = \frac{\sum_{t=1}^{n} \sum_{r=1}^{c} g_{ir} \left\| \mathbf{x}_t - \mathbf{v}_r \right\|^2}{n} \tag{16}$$

Here $L_{\mathbf{x}}$ is a random functional for each realization of $\mathbf{x}$, and $\Gamma(\mathbf{V})$ is its expectation. Hence exact optimization of Γ using ordinary gradient descent is difficult . We have seen that i , the index for the winner, is a function of $\mathbf{x}$ and all of $\mathbf{v}_r$ s. The function $L_{\mathbf{x}}$ is well defined. If we assume that $\mathbf{x}$ has a unique distance from each $\mathbf{v}_r$, then i and g_{ir} are uniquely determined, and hence $L_{\mathbf{x}}$ is also uniquely determined. However, if the above assumptions are not met, then i and g_{ir} will have discontinuities. In the following discussion we assume that g_{ir} does not have discontinuities so that the gradient of $L_{\mathbf{x}}$, exists. As most learning algorithms do[18], we

approximate the gradient of $\Gamma(\mathbf{V})$ by the gradient of the sample function $L_{\mathbf{x}}$. In other words, We attempt to minimize Γ by local gradient descent search using the sample function $L_{\mathbf{x}}$. It is our conjecture that the optimal values of $\mathbf{v}_r$'s can be approximated in an iterative, stepwise fashion by moving in the direction of gradient of $L_{\mathbf{x}}$. The algorithm is derived as follows (for notational simplicity the subscript for $\mathbf{x}$ will be ignored). First rewrite L as :

$$L = \sum_{r=1}^{c} g_{ir} \left\| \mathbf{x} - \mathbf{v}_r \right\|^2 = \left\| \mathbf{x} - \mathbf{v}_i \right\|^2 + \sum_{\substack{r=1 \\ r \neq i}}^{c} \left\| \mathbf{x} - \mathbf{v}_r \right\|^2 / \sum_{j=1}^{c} \left\| \mathbf{x} - \mathbf{v}_j \right\|^2$$

$$= \left\| \mathbf{x} - \mathbf{v}_i \right\|^2 + \sum_{r=1}^{c} \left\| \mathbf{x} - \mathbf{v}_r \right\|^2 / \sum_{j=1}^{c} \left\| \mathbf{x} - \mathbf{v}_j \right\|^2 - \left\| \mathbf{x} - \mathbf{v}_i \right\|^2 / \sum_{j=1}^{c} \left\| \mathbf{x} - \mathbf{v}_j \right\|^2$$

$$= \left\| \mathbf{x} - \mathbf{v}_i \right\|^2 + 1 - \left\| \mathbf{x} - \mathbf{v}_i \right\|^2 / \sum_{j=1}^{c} \left\| \mathbf{x} - \mathbf{v}_j \right\|^2 . \qquad (17)$$

Differentiating L with respect $\mathbf{v}_i$ yields (after some algebraic manipulations) :

$$\nabla_{\mathbf{v}} L(\mathbf{v}_i) = -2(\mathbf{x} - \mathbf{v}_i) \frac{D^2 - D + \left\| \mathbf{x} - \mathbf{v}_i \right\|^2}{D^2} \qquad (18)$$

where $D = \sum_{r=1}^{c} \left\| \mathbf{x} - \mathbf{v}_r \right\|^2$. On the other hand, differentiation of L with respect to $\mathbf{v}_j$ ($j \neq i$) yields:

$$\nabla_{\mathbf{v}} L(\mathbf{v}_j) = -2(\mathbf{x} - \mathbf{v}_j) \frac{\left\| \mathbf{x} - \mathbf{v}_i \right\|^2}{D^2} \qquad (19)$$

Update rules based on (17) and (18) are :

$$\mathbf{v}_{i,t} = \mathbf{v}_{i,t-1} + \alpha_t \, (\mathbf{x} - \mathbf{v}_{i,t-1}) \frac{D^2 - D + \left\| \mathbf{x} - \mathbf{v}_{i,t-1} \right\|^2}{D^2} \quad \text{for the winner node i, and} \qquad (20)$$

$$\mathbf{v}_{j,t} = \mathbf{v}_{j,t-1} + \alpha_t \, (\mathbf{x} - \mathbf{v}_{j,t-1}) \frac{\left\| \mathbf{x} - \mathbf{v}_{i,t-1} \right\|^2}{D^2} \quad \text{for the other (c-1) nodes, } j \neq i . \qquad (21)$$

To avoid possible oscillations of the solution, the amount of correction should be reduced as iteration proceeds. Moreover, like optimization techniques using subgradient descent search, as one moves closer to an optimum the amount of correction should be reduced (in fact, α_t should satisfy the following two conditions : as $t \to \infty$; $\alpha_t \to 0$ and $\Sigma\, \alpha_t \to \infty$)[19]. On the other hand, in the presence of noise, under a suitable assumption about subgradients, the search becomes successful if the conditions in (10) are satisfied. We recommend a decreasing sequence of α_t ($0 < \alpha_t < 1$) satisfying (10) , which insure that α_t is neither reduced too fast nor too slow. From the point of view of learning, the system should be stable enough to remember old learned patterns, and yet plastic enough to learn new patterns (Grossberg calls it the *stability-plasticity dilemma*)[20]. Condition (10a) enables plasticity, while (10b) enforces stability . In other words, an incoming input should not affect the parameters of a learning system too strongly, thereby enabling it to remember old learned patterns (stability); at the same time, the system should be responsive enough to recognize any new trend in the input (plasticity). Hence, α_t can be taken as $\alpha_0(1\text{-}t/T)$, where T is the maximum number of iterations the learning process is allowed to execute and α_0 is the initial value of the learning parameter. Referring to (20), we see that when the match is perfect then nonwinner nodes are not updated; in other words, this strategy then reduces to LVQ. On the other hand, as the match between $\mathbf{x}$ and the winner node $\mathbf{v}_i$ decreases, the impact on other (nonwinner) nodes increases. This seems to be an intuitively desirable property. We summarize the GLVQ algorithm as follows:

GLVQ Clustering Algorithm:

GLVQ1. Given unlabeled data set $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n\} \subset \Re^p$. Fix c, T, and $\varepsilon > 0$.

GLVQ2. Initialize $\mathbf{V}_0 = (\mathbf{v}_{1,0}, \ldots, \mathbf{v}_{c,0}) \in \Re^{cp}$, and learning rate $\alpha_0 \in (1,0)$.

GLVQ3. For t = 1, 2, ..., T.

a. Compute $\alpha_t = \alpha_0 (1\text{-}t/T)$.

While k≤n

b. Find $\left\|\mathbf{x}_k - \mathbf{v}_{i,t-1}\right\| = \min_{1 \le j \le c} \left\{\left\|\mathbf{x}_k - \mathbf{v}_{j,t-1}\right\|\right\}$.

c. Update all (c) weight vectors $\{\mathbf{v}_{r,t}\}$ with

$$\mathbf{v}_{i,t} = \mathbf{v}_{i,t-1} + \alpha_t \,(\mathbf{x}_k - \mathbf{v}_{i,t-1})\, \frac{D^2 - D + \left\|\mathbf{x}_k - \mathbf{v}_{i,t-1}\right\|^2}{D^2} \quad \cdot D = \sum_{r=1}^{c} \left\|\mathbf{x} - \mathbf{v}_r\right\|^2$$

$$\mathbf{v}_{r,t} = \mathbf{v}_{r,t-1} + \alpha_t \, (\mathbf{x}_k - \mathbf{v}_{r,t-1}) \, \frac{\left\| \mathbf{x}_k - \mathbf{v}_{r,t-1} \right\|^2}{D^2} \quad (r \neq i) \qquad , D = \sum_{r=1}^{c} \left\| \mathbf{x} - \mathbf{v}_r \right\|^2$$

Wend

d. Compute $\left\| \mathbf{V}_t - \mathbf{V}_{t-1} \right\| = \sum_{r=1}^{c} \left\| \mathbf{v}_{r,t} - \mathbf{v}_{r,t-1} \right\|_1 = \sum_{k=1}^{n} \sum_{r=1}^{c} \left| v_{rk,t} - v_{rk,t-1} \right|$.

e. If $E_t \leq \varepsilon$ stop; Else

Next t.

GLVQ4. Compute non-iteratively the nearest prototype GLVQ c-partition of X :

$$u_{GLVQ_{ik}} = \begin{cases} 1; & \left\| \mathbf{x}_k - \mathbf{v}_i \right\| \leq \left\| \mathbf{x}_k - \mathbf{v}_j \right\| , 1 \leq j \leq c, j \neq i \\ 0; & \textit{otherwise} \end{cases} \qquad , 1 \leq i \leq c \text{ and } 1 \leq k \leq n.$$

Comments on GLVQ :

1. There is no need to choose an update neighborhood .

2. Reduction of the learning coefficient with distance (either topological or in $\Re^p$) from the winner node is not required. Instead, reduction is done automatically and adaptively by the learning rules.

3. For each input vector, either all nodes get updated or no node does. When there is a perfect match to the winner node, no node is updated. In this case GLVQ reduces to LVQ.

4. The greater the mismatch to the winner (i.e., the higher the quantization error), the greater the impact to weight vectors associated with other nodes. Quantization error is the error in representing a set of input vectors by a prototype - in the above case the weight vector associated with the winner node.

5.The learning process attempts to minimize a well-defined objective function.

6. Our termination strategy is based on small successive changes in the cluster centers. This method of algorithmic control offers the best set of centroids for compact representation (quantization) of the data in each cluster.

4. FUZZY LEARNING VECTOR QUANTIZATION (FLVQ)

Huntsberger and Ajjimarangsee[11] used SOFMs to develop clustering algorithms. Algorithm 1 in [11] is the SOFM algorithm with an additional layer of neurons. This additional set of neurons does not participate in weight updating. After the self-organizing network terminates, the additional layer, for each input, finds the weight vector (prototype) closest to it and assigns the input data point to that class. A second algorithm in their paper used the necessary conditions for FCM to assign a membership value in [0,1] to each data point. Specifically, Huntsberger and Ajjimarangsee suggested fuzzification of LVQ by replacing the learning rates $\{\alpha_{ik,t}\}$ usually found in rules such as (7) with fuzzy membership values $\{u_{ik,t}\}$ computed with the FCM formula [2]:

$$\alpha_{ik,t} = u_{ik,t} = \left(\sum_{j=1}^{c} \frac{D_{ik,t}}{D_{jk,t}} \right)^{\frac{-2}{m-1}} \quad , \qquad (22)$$

where $D_{ik,t} = \left\| \mathbf{x}_k - \mathbf{v}_{i,t} \right\|_A$. Numerical results reported in Huntsberger and Ajjimarangsee suggest that in many cases their algorithms and standard LVQ produce very similar answers. Their scheme was a partial integration of LVQ with FCM that showed some interesting results. However, it fell short of realizing a *model* for LVQ clustering; and no properties regarding terminal points or convergence were established. Moreover, since the objective of these LVQ is to find cluster centroids (prototypes), and hence clusters, there is no need to have a topological ordering of the weight vectors. Consequently, the approach taken in [11] seems to mix two objectives, feature mapping and clustering, and the overall methodology is difficult to interpret in either sense.

Integration of FCM with LVQ can be more fully realized by defining the learning rate for Kohonen updating as :

$$\alpha_{ik,t} = (u_{ik,t})^{m_t} = \left(\sum_{j=1}^{c} \frac{D_{ik,t}}{D_{jk,t}} \right)^{\frac{-2m_t}{m_t-1}} \quad , \quad \text{where} \qquad (23a)$$

$$m_t = m_0 + t[(m_f - m_0)/T] = m_0 + t\Delta m \; ; \qquad m_f, m_0 \geq 1; \qquad t=1,2,\ldots T. \qquad (23b)$$

m_t replaces the (fixed) parameter m in (22). This results in three families of *Fuzzy LVQ* or FLVQ algorithms, the cases arising by different treatments of paramerer m_t. In particular, for

$t \in \{1,2,\ldots,T\}$, we have three cases depending on the choice of the initial (m_0) and final (m_f) values of m:

1. $m_0 > m_f \Rightarrow \{m_t\} \downarrow m_f$: Descending FLVQ (24a)
2. $m_0 < m_f \Rightarrow \{m_t\} \uparrow m_f$: Ascending FLVQ (24b)
3. $m_0 = m_f \Rightarrow m_t \equiv m_0 \equiv m$: FLVQ ≡ FCM (24c)

Cases 1 and 3 are discussed at length by Bezdek et. al.[17]. Case 2 is fully discussed in Tsao et. al.[21]. Equation (24c) asserts that when $m_0 = m_f$, FLVQ reverts to FCM; this results from defining the learning rates via (23a), and using them in FLVQ3.b below. FLVQ is not a direct generalization of LVQ because it does not revert to LVQ in case all of the $u_{ik,t}$'s are either 0 or 1 (the crisp case). Instead, if $m_0 = m_f = 1$, FCM reverts to HCM, and the HCM update formula, which *is* driven by finding unique winners, as is LVQ, is a different formula than (7). FLVQ is perhaps the closest possible link between LVQ and c-Means type algorithms. We provide a formal description of FLVQ :

Fuzzy LVQ (FLVQ)

FLVQ1. Given unlabeled data set $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$. Fix c, T, $\|\ \|_A$ and $\varepsilon > 0$.

FLVQ2. nitialize $\mathbf{v}_0 = (\mathbf{v}_{1,0}, \ldots, \mathbf{v}_{c,0}) \in \Re^{cp}$. Choose $m_0, m_f \geq 1$.

FLVQ3. For t = 1, 2, ..., T.

a. Compute all (cn) learning rates $\{\alpha_{ik,t}\}$ with (23).

b. Update all (c) weight vectors $\{\mathbf{v}_{i,t}\}$ with $\mathbf{v}_{i,t} = \mathbf{v}_{i,t-1} + \sum_{k=1}^{n} \alpha_{ik,t}(\mathbf{x}_k - \mathbf{v}_{i,t-1}) / \sum_{s=1}^{n} \alpha_{is,t}$

c. Compute $E_t = \left\|\mathbf{v}_t - \mathbf{v}_{t-1}\right\| = \sum_{i=1}^{c} \left\|\mathbf{v}_{i,t} - \mathbf{v}_{i,t-1}\right\|$.

d. If $E_t \leq \varepsilon$ stop; Else

Next t.

For fixed c, $\{\mathbf{v}_{i,t}\}$ and m_t, the learning rates $\alpha_{ik,t} = (u_{ik,t})^{m_t}$ at (23a) satisfy the following :

$$\alpha_{ik,t} = (u_{ik,t})^{m_t} = \left(\frac{\kappa}{D_{ik,t}}\right)^{\frac{2m_t}{m_t - 1}} \quad , \tag{25}$$

where κ is a positive constant. Apparently the contribution of $\mathbf{x}_k$ to the next update of the node weights is inversely proportional to their distances from it. The "winner" in (29) is the $\mathbf{v}_{i,t-1}$

closest to $\mathbf{x}_k$, and it will be moved further along the line connecting $\mathbf{v}_{i,t-1}$ to $\mathbf{x}_k$ than any of the other weight vectors. Since $\sum u_{ik,t} = 1 \Rightarrow \sum \alpha_{ik,t} \leq 1$, this amounts to distributing partial updates across all c nodes for each $\mathbf{x}_k \in X$. This is in sharp contrast to LVQ, where only the winner is updated for each data point.

In *descending* FLVQ (24a), for large values of m_t (near m_0), all c nodes are updated with lower individual learning rates, and as $m_t \rightarrow 1$, more and more of the update is given to the "winner" node. In other words, the lateral distribution of learning rates is a function of t, which in the descending case "sharpens" at the winner node (for each $\mathbf{x}_k$) as $m_t \xrightarrow{+} 1$. Finally, we note again that for fixed m_t, FLVQ updates the $\{\mathbf{v}_{i,t}\}$ using the conditions that are necessary for FCM; each step of FLVQ is one iteration of FCM.

Figure 8. Updating Feature Space Prototypes in FLVQ Clustering Nets.

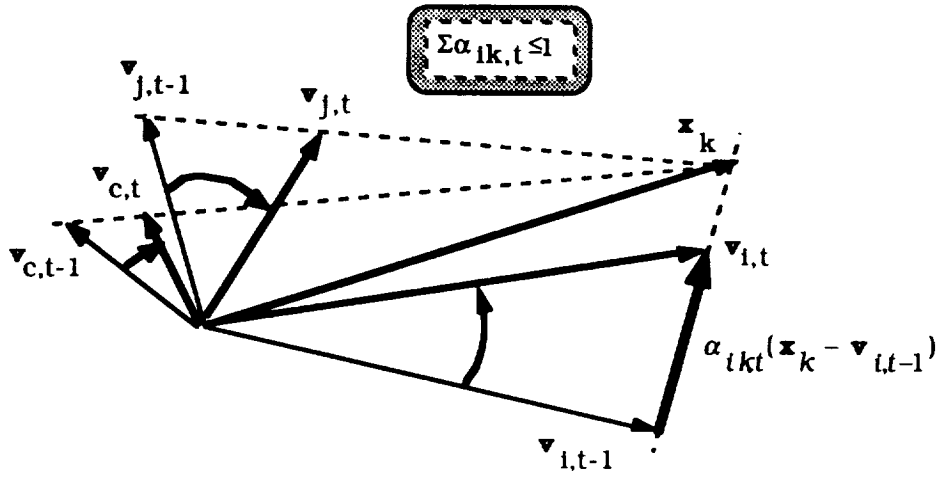


Figure 8 illustrates the update geometry of FLVQ; note that *every* node is (potentially) updated at every iteration, and the sum of the learning rates is always less than or equal to one.

Comments on FLVQ :

1. There is no need to choose an update neighborhood .

2. Reduction of the learning coefficient with distance (either topological or in $\Re^p$) from the winner node is not required. Instead, reduction is done automatically and adaptively by the learning rules.

3. The greater the mismatch to the winner (i.e., the higher the quantization error), the *smaller* the impact to the weight vectors associated with other nodes (recall (25) and (2c)). This is directly opposite to the situation in GLVQ.

4.The learning process attempts to minimize a well-defined objective function (stepwise).

5. Our termination strategy is based on small successive changes in the cluster centers. This method of algorithmic control offers the best set of centroids for compact representation (quantization) of the data in each cluster.

6. This procedure depends on generation of a fuzzy c-partition of the data, so it is an iterative clustering model - indeed, stepwise, it is exactly fuzzy c-means [17].

5. IMAGE SEGMENTATION WITH GLVQ AND FLVQ

In this section we illustrate the (FLVQ and GLVQ) algorithms with image segmentation, which can be achieved either by finding spatially compact homogeneous regions in the image; or by detecting boundaries of regions, i.e., detecting the edges of each region. We have applied our clustering strategies to both paradigms. Image segmentation by clustering raises the important issue of feature extraction / selection. Generally, features relevant for identifying compact regions are different from those useful for the edge detection approach.

Feature selection for homogeneous region extraction

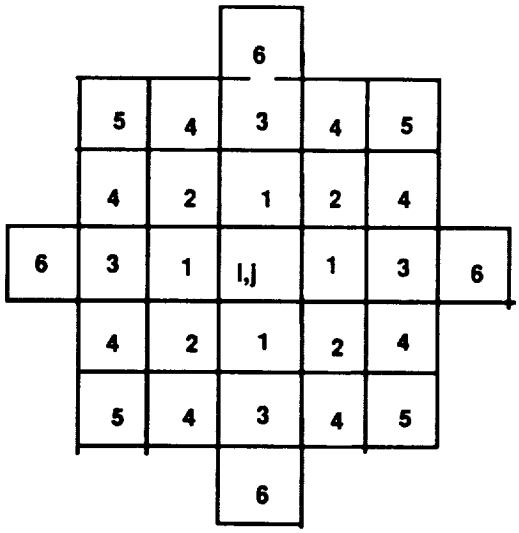
When looking for spatially compact regions, feature vectors should incorporate information about the spatial distribution of gray values. For pixel (i,j) of a digital image $F= \{(i,j) \mid 1 \le i \le M ; 1 \le j \le N\}$, we define the *$d^{th}$ order neighborhood* of (i,j), where $d > 0$ is an integer as ;

$$N_{i,j}^{d} = \{(k,l) \in F\} \text{ such that } (i,j) \notin N_{i,j}^{d} \text{ and if } (k,l) \in N_{i,j}^{d} \text{ then } (i,j) \in N_{k,l}^{d} . \qquad (26)$$

Several such neighborhoods are depicted in Figure 9, where $N_{i,j}^{d}$ consists of all pixels marked with an index $\le d$. For example N^1 is obtained by taking the four nearest neighbor pixels to (i,j). Similarly, N^2 is defined by its eight nearest neighbors, and so on. $N_{i,j}^{d}$ as defined in (26) is the standard neighborhood definition for modeling digital images using Gibbs or Markov Random Fields. To define feature vectors for segmentation, we extend the definition of a d-th order neighborhood at (26) to include the center pixel (i,j):

$$N_{i,j}^{d'} = N_{i,j}^{d} \cup \{(i,j)\} \quad ; \quad D_{ij} = \left| N_{i,j}^{d'} \right| \qquad (27)$$

Figure 9. An Ordered Neighborhood system



Next, let L= {1,2,..., G} be the set of gray values that can be taken by pixels in the image, and let f(i,j) be the intensity at (i,j) in F, that is, $f: F \mapsto L$. We define the collection of gray values of all pixels that belong to $N_{i,j}^{d'}$ as:

$$S_{i,j}^{d} = \{ f(k,l) \mid (k,l) \in N_{i,j}^{d'} \} \quad . \tag{28}$$

Note that $S_{i,j}^{d}$ may contain the same gray value more than once. We say two neighborhoods $N_{i,j}^{d'}$ and $N_{k,l}^{d'}$ are *equally homogeneous* in case $S_{i,j}^{d}$ and $S_{k,l}^{d}$ are identical up to a permutation. This assumption is natural and useful as long as the neighborhood size is small. To see this, consider two 100x100 neighborhoods that contain 5000 pixels with gray value 1 and 5000 with value G. Satisfaction of this property gives the impression of two perfectly homogeneous regions ; but in fact one of these neighborhoods might have all 5000 pixels of each intensity in, say, the upper and lower halves of the image, while other neighborhood has a completely random mixture of black and white spots. When the neighborhood size is small, however, spatial rearrangement of a few gray values among many more in the entire image will not create a much different impression to the human visual system as far as homogeneity of the region is concerned. Therefore, for small values of d we can derive features for (i,j) from $S_{i,j}^{d}$ which are relatively independent of permutation of its elements (typically, such features might include the mean, standard deviation, etc. of the intensity values in $S_{i,j}^{d}$). Subsequently, these features are arrayed into a *pixel vector* $\mathbf{x}_{ij}$ for each pixel. In this

investigation, we used the gray values in $S^d_{i,j}$ themselves as the feature vector for pixel (i,j); thus, each (i,j) in F (excluding boundaries) is associated with $\mathbf{x}_{ij}$ in $\Re^D$.

Since FLVQ and GLVQ both use distances between feature vectors, we *sorted* the values in $S^d_{i,j}$ to get each $\mathbf{x}_{ij}$. Sorting can be done either in ascending or in descending order, but the same strategy must be used for all pixels. We remark that an increase in the d-size of the neighborhood will obscure finer details in the segmented image; conversely, a very low value of d usually results in too many small regions. Experimental investigation suggests that $3 \le d \le 5$ provides a reasonable tradeoff between fine and gross structure.

Feature selection for edge extraction

Loosely speaking edges are regions of abrupt changes in gray values. Therefore, features used for extraction of homogeneous regions are not suitable for edge-nonedge classification. For this approach, we nominate a feature vector $\mathbf{x}_{ij}$ in $\Re^3$ with three components : standard deviation, gradient 1 and gradient 2. In other words, each pixel is represented by a 3-tuple $\mathbf{x}_{ij}$ = $(\sigma(i,j), G1(i,j), G2(i,j))$. The standard deviation is defined on $S^d_{i,j}$ as follows:

$$\sigma(i,j) = \{\frac{1}{|S^d_{i,j}|} \sum_{g \in S^d_{i,j}} (g - \mu_{i,j})^2\}^{1/2} \quad , \tag{29}$$

where $\mu_{i,j}$ is the average gray value over $S^d_{i,j}$. Since standard deviation measures variation of gray values over the neighborhood, using too large a neighborhood will destroy its utility for edge detection. The two gradients are defined as :

$$G1(i,j) = |f_{i+1,j} - f_{i-1,j}| + |f_{i,j-1} - f_{i,j+1}| \quad ; \text{and} \tag{30}$$

$$G2(i,j) = |f_{i+1,j+1} - f_{i-1,j-1}| + |f_{i+1,j-1} - f_{i-1,j+1}|. \tag{31}$$

Note that G1 measures intensity changes in the horizontal and vertical directions, while G2 takes into account diagonal edges; this justifies the use of both G1 and G2.

Implementation

FLVQ (ascending strategy) and GLVQ were used for segmentation of the house image depicted in Figure 10(a). This image is a very complex image for segmentation into homogeneous regions, because it has some textured portions (the trees) behind the house. For the region extraction

scheme we used neighborhoods of order d=3 and d=5. The number of classes chosen was c=8. The computing protocols used for different runs are summarized in Table 2.

Table 2. Computing protocols for the segmentations

Since FLVQ produces fuzzy labels for each pixel vector, the fuzzy label vector is defuzzified using the maximum membership rule at (5). Thus, each pixel receives a crisp label corresponding to one of the c classes in the segmented image. Coloring of the segmented image is done by using c distinct gray values, one for each class. Defuzzification is not required for the GLVQ algorithm as it produces hard labels.

Figure 10 contains some typical outputs of both FLVQ and GLVQ using the region-based segmentation approach. To show the effect of sorting we ran both algorithms with unsorted and sorted feature vectors. Figure 10(b) represents the segmented output produced by FLVQ with d=3 and unsorted features; while figure 10(c) displays the output under the same conditions, but with sorted features. Comparing figures 10(b) and (c) one sees that the noisy patches on the roof of the house that appear in Fig. 10(b) are absent in Fig. 10(c). Similar occurences can be found in other portions of the image. This demonstrates that sorted pixel vectors seem to afford some noise cleaning ability. Figure 10(d) was produced with FLVQ using sorted neighborhoods of size 5. Note that the textured tree areas have been segmented more compactly; this illustrates the effect of increasing the neighborhood size. Figures 10 (e) and (f) are produced by the GLVQ algorithm with sorted neighborhoods of orders 3 and 5, respectively.

FLVQ	**norm**	**c**	m_0	Δm	**T**	ε	**iterations**
Fig. 10(b)	Euclidean	8	1.05	0.2	80	0.5	25
Fig. 10(c)	Euclidean	8	1.05	0.2	80	0.5	24
Fig. 10(d)	Euclidean	8	1.05	0.2	80	0.5	29
Fig.11(a)	Euclidean	2	1.05	0.2	80	0.5	17
GLVQ	**norm**	**c**	α_0	$\Delta\alpha$	**T**	ε	**iterations**
Fig.10(e,f)	Euclidean	8	0.6	0.06	100	0.5	100
Fig. 11(b)	Euclidean	2	0.6	0.06	100	0.5	100

Comparing figures 10(c) and (e) we find that FLVQ and GLVQ are comparable for the house, but GLVQ extracts more compact regions for the tree areas. Another interesting thing to note is that for GLVQ with a window of size 5x5, the roof of the house is very nicely segmented with sharp inter-region boundaries; this is not true for all other cases using either algorithm.

Fig. 10 (a) Input image of a house

Fig. 10(b) FLVQ with N^3 (unsorted)

Fig. 10(c) FLVQ with N^3 (sorted)

Fig. 10(d) FLVQ with N^5 (sorted)

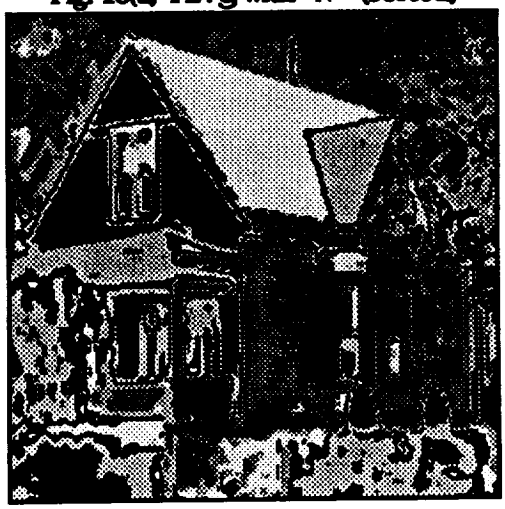


Fig. 10 (e) GLVQ with N^3 (sorted)

Fig. 10 (f) GLVQ with N^5 (sorted)

We used the same image (Figure 10 (a)) to test the edge-based approach. The results produced by FLVQ and GLVQ are shown in Figures 11(a) and (b), respectively. Comparing these two figures, one can see that both algorithms have extracted the compact regions nicely. A careful analysis of the images shows that FLVQ detects more edges than GLVQ. As a result of this FLVQ produces some noisy edges and GLVQ fails to extract some important edges. To summarize, both algorithms produce reasonably good results, but GLVQ has a tendency to produce larger compact (homogeneous) areas than that by the FLVQ. It appears that GLVQ is less sensitive to noise which might cause a failure to extract finer details.

Fig. 11(a) FLVQ (edge/nonedge)

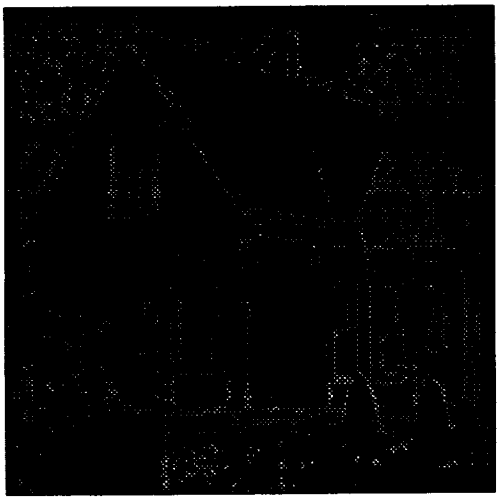
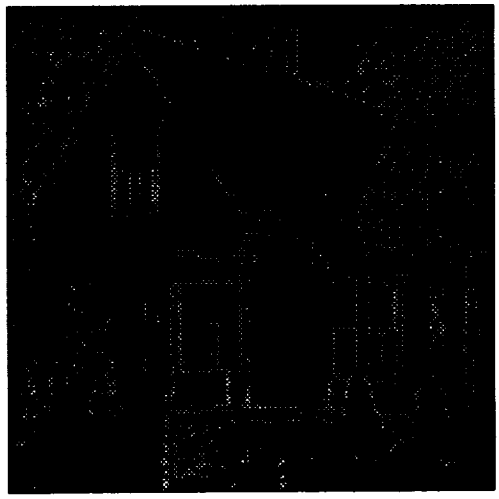


Fig. 11(b) GLVQ (edge/nonedge)



6. CONCLUSIONS

We have considered the role of and interaction between fuzzy and neural-like models for clustering, and have illustrated two generalizations of LVQ with an application in image segmentation. Unlike methods that utilize Kohonen's SOFM idea, both algorithms avoid the necessity of defining an update neighborhood scheme. Both methods are designed to optimize performance goals related to clustering, and both have update rules that allocate and distribute learning rates to (possibly) all c nodes at each pass through the data. Ascending and descending FLVQ updates all nodes at each pass, and learning rates are related to the fuzzy c-means clustering algorithm. This yields automatic control of the learning rate distribution and the update neighborhood is effectively all c nodes at each pass through the data. FLVQ can be considered a (stepwise) implementation of FCM. GLVQ needs only a specification of the

learning rate sequence and an initialization of the c protoytpes. GLVQ either updates all nodes for an input vector, or it does not update any. When an input vector exactly matches the winner node, GLVQ reduces to LVQ. Otherwise, all nodes are updated inversely proportionally to their distances from the input vector.

7. REFERENCES

1. Kohonen, T. *Self-Organization and Associative Memory*, 3rd Edition, Springer-Verlag, Berlin, 1989.

2. Bezdek, J. *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum,NY, 1981.

3. Duda, R. and Hart, P. *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.

4. Tou , J. and Gonzalez, R. *Pattern Recognition Principles*, Addison-Wesley, Reading, 1974.

5. Hartigan, J. *Clustering Algorithms*, Wiley, New York, 1975.

6. Dubes, R. and Jain, A. *Algorithms that Cluster Data*, Prentice Hall,

7. Pao, Y.H. *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley, Reading, 1989.

8. Ruspini, E.H. A New Approach to Clustering, *Inform. Control*, 15, 22-32, 1969.

9. Bezdek, J.C. and Pal, S.K. Fuzzy Models for Pattern Recognition, IEEE Press, Piscataway, 1992.

10. Krishnapuram, R. and Keller, J. A Possibilistic Approach to Clustering, in press, *IEEE Trans. Fuzzy Systems*, 1993.

11. Huntsberger, T. and Ajjimarangsee, P. Parallel Self-Organizing Feature Maps for Unsupervised Pattern Recognition, *Int'l. Jo. General Systems*, vol. 16, pp. 357-372, 1989.

12. Kohonen, T, Self-organizing maps: optimization approach, Artificial Neural networks, *Elsevier Sc. Pub.*, (Eds. T. Kohonen, K. Makisara, O. Simula and J. Kangas), 1991, 981-990.

13. Robbins, H. and Monro, S., A stochastic approximation method, *Ann. Math.Stat.*, 22, 400-407,1951.

14. Albert, A. E. and Gardner, L. A., Jr., Stochastic approximation and nonlinear regression, *MIT Press*, Cambridge, MA, 1967.

15. MacQueen J., Classification and analysis of multivariate observations, Some methods for classification and analysis of multivariate observations, *Proc. 5th Berkeley Symp.on Math. Stat. and Prob.*, 281-297, 1967.

16. Forgy, E. , Cluster analysis of multivariate dada : efficiency vs interpretability of classifications, WNAR meetings, Univ. of Cali - Riverside, June 22-23, 1965 (Biometrics, 21(3)).

17. Bezdek, J. C., Tsao, E. C. and Pal, N. R., Fuzzy Kohonen Clustering Networks, *Proc. IEEE Inter. Conf. of Fuzzy Syst.* , 1035-1041, March 1992, San Diego, USA .

18. Tsypkin, Y. Z., Foundations of the theory of learning systems, Trans. Z. J. Nikolic, *Academic Press*, NY, 1973.

19. Polyak, B.T., Introduction to Optimization, *Optimization Software Inc.*, New York, 1987.

20. Grossberg, S., *Studies of mind and brain*, Reidel, Boston, 1982.

21. Tsao, E.C.K., Bezdek, J.C. and Pal, N.R. Image Segmentation using Fuzzy Kohonen Clustering Networks, in press, *Proc. NAFIPS*, 1992.

22. Anderson, E., The IRISes of the Gaspe Peninsula, *Bulletin of the American IRIS Society*, 59, 2-5, 1939.

23. Pal, N.R. , Bezdek, J. C. and Tsao, E.C.K., Generalized Clustering networks and Kohonen's Self-Organizing Scheme, in press, *IEEE Trans. NN*, 1992.

Possibilistic Clustering for Shape Recognition[1]

James M. Keller and Raghu Krishnapuram
Department of Electrical and Computer Engineering
University of Missouri, Columbia, MO 65211

Abstract

Clustering methods have been used extensively in computer vision and pattern recognition. Fuzzy clustering has been shown to be advantageous over crisp (or traditional) clustering in that total commitment of a vector to a given class is not required at each iteration. Recently fuzzy clustering methods have shown spectacular ability to detect not only hypervolume clusters, but also clusters which are actually "thin shells", i.e., curves and surfaces. Most analytic fuzzy clustering approaches are derived from Bezdek's Fuzzy C-Means (FCM) algorithm. The FCM uses the probabilistic constraint that the memberships of a data point across classes sum to one. This constraint was used to generate the membership update equations for an iterative algorithm. Unfortunately, the memberships resulting from FCM and its derivatives do not correspond to the intuitive concept of degree of belonging, and moreover, the algorithms have considerable trouble in noisy environments. Recently, we cast the clustering problem into the framework of possibility theory. Our approach was radically different from the existing clustering methods in that the resulting partition of the data can be interpreted as a possibilistic partition, and the membership values may be interpreted as degrees of possibility of the points belonging to the classes. We constructed an appropriate objective function whose minimum will characterize a good possibilistic partition of the data, and we derived the membership and prototype update equations from necessary conditions for minimization of our criterion function. In this paper, we show the ability of this approach to detect linear and quartic curves in the presence of considerable noise.

[1]Research performed for NASA/JSC through a subcontract from the RICIS Center at the University of Houston - Clear Lake

I. Introduction

Clustering has long been a popular approach to unsupervised pattern recognition. It has become more attractive with the connection to neural networks, and with the increased attention to fuzzy clustering. In fact, recent advances in fuzzy clustering have shown spectacular ability to detect not only hypervolume clusters, but also clusters which are actually "thin shells", i.e., curves and surfaces [1-7]. One of the major factors that influences the determination of appropriate groups of points is the "distance measure" chosen for the problem at hand. Fuzzy clustering has been shown to be advantageous over crisp (or traditional) clustering in that total commitment of a vector to a given class is not required at each iteration.

Boundary detection and surface approximation are important components of intermediate-level vision. They are the first step in solving problems such as object recognition and orientation estimation. Recently, it has been shown that these problems can be viewed as clustering problems with appropriate distance measures and prototypes [1-7]. Dave's Fuzzy *C* Shells (FCS) algorithm [2] and the Fuzzy Adaptive C-Shells (FACS) algorithm [7] have proven to be successful in detecting clusters that can be described by circular arcs, or more generally by elliptical shapes. Unfortunately, these algorithms are computationally rather intensive since they involve the solution of coupled nonlinear equations for the shell (prototype) parameters. These algorithms also assume that the number of clusters are known. To overcome these drawbacks we recently proposed a computationally simpler Fuzzy *C* Spherical Shells (FCSS) algorithm [6] for clustering hyperspherical shells and suggested an efficient algorithm to determine the number of clusters when this is not known. We also proposed the Fuzzy C Quadric Shells (FCQS) algorithm [5] which can detect more general quadric shapes. One problem with the FCQS algorithm is that it uses the algebraic distance, which is highly nonlinear. This results in unsatisfactory performance when the data is not very "clean" [7]. Finally, none of the algorithms can handle situations in which the clusters include lines/planes and there is much noise. In [8], we addressed those issues in a new approach called Plano-Quadric Clustering. In this paper, we show how that algorithm, coupled with our new possibilistic clustering, can accurately find linear and quadric curves in the presence of noise.

Most analytic fuzzy clustering approaches are derived from Bezdek's Fuzzy C-Means (FCM) algorithm [9]. The FCM uses the probabilistic constraint that the memberships of a data point across classes must sum to one. This constraint came from generalizing a crisp C-Partition of a data set, and was used to generate the membership update equations for an iterative algorithm. These equations emerge as necessary conditions for a global minimum of a least-squares type of criterion function. Unfortunately, the resulting memberships do not represent one's intuitive notion of degrees of belonging, i. e., they do not represent degrees of "typicality" or "possibility".

There is another important motivation for using possibilistic memberships. Like all unsupervised techniques, clustering (crisp or fuzzy) suffers from the presence of noise in the data. Since most distance functions are geometric in nature, noise points, which are often quite distant from the primary clusters, can drastically influence the estimates of the class prototypes, and hence, the final clustering. Fuzzy methods ameliorate this problem when the number of classes is greater than one, since the noise points tend to have somewhat smaller membership values in all the classes. However, this difficulty still remains in the fuzzy case, since the memberships of unrepresentative (or noise) points can still be significantly high. In fact, if there is only one real cluster present in the data, there is essentially no difference between the crisp and fuzzy methods.

On the other hand, if a set of feature vectors is thought of as the domain of discourse for a collection of independent fuzzy subsets, then there should be no constraint on the sum of the memberships. The only real constraint is that the assignments do really represent fuzzy membership values, i.e., they must lie in the interval [0,1]. In [10], we cast the clustering problem

into the framework of possibility theory. We briefly review this approach, and show it's superiority to recognize shapes from noisy and incomplete data.

II. Possibilistic Clustering Algorithms

The original FCM formulation minimizes the objective function given by

$$J(L,U) = \sum_{i=1}^{C} \sum_{j=1}^{N} (\mu_{ij})^m d_{ij}^2 \text{, subject to } \sum_{i=1}^{C} \mu_{ij} = 1 \text{ for all } j. \qquad (1)$$

In (1), $L = (\lambda_1,...,\lambda_C)$ is a C-tuple of prototypes, d_{ij}^2 is the distance of feature point $\boldsymbol{x}_j$ to cluster λ_i, N is the total number of feature vectors, C is the number of classes, and $U = [\mu_{ij}]$ is a $C \times N$ matrix called the fuzzy C-partition matrix [9] satisfying the following conditions:

$$\mu_{ij} \in [0,1] \text{ for all } i \text{ and } j, \qquad \sum_{i=1}^{C} \mu_{ij} = 1 \text{ for all } j, \text{ and}$$

$$0 < \sum_{j=1}^{N} \mu_{ij} < N \text{ for all } i.$$

Here, μ_{ij} is the grade of membership of the feature point $\boldsymbol{x}_j$ in cluster λ_i, and $m \in [1,\infty)$ is a weighting exponent called the fuzzifier. In what follows, λ_i will also be used to denote the ith cluster, since it contains all of the parameters that define the prototype of the cluster.

Simply relaxing the constraint in (1) produces the trivial solution, i. e., the criterion function is minimized by assigning all memberships to zero. Clearly, one would like the memberships for representative feature points to be as high as possible, while unrepresentative points should have low membership in all clusters. This is an approach consistent with possibility theory [11]. The objective function which satisfies our requirements may be formulated as:

$$J_m(L,U) = \sum_{i=1}^{C} \sum_{j=1}^{N} (\mu_{ij})^m d_{ij}^2 + \sum_{i=1}^{C} \eta_i \sum_{j=1}^{N} (1-\mu_{ij})^m . \qquad (2)$$

where η_i are suitable positive numbers. The first term demands that the distances from the feature vectors to the prototypes be as low as possible, whereas the second term forces the μ_{ij} to be as large as possible, thus avoiding the trivial solution. The following theorem, proved in [9], gives necessary conditions for minimization, hence, providing the basis for an iterative algorithm.

Theorem:

Suppose that $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$ is a set of feature vectors, $L = (\lambda_1,...,\lambda_C)$ is a C-tuple of prototypes, d_{ij}^2 is the distance of feature point $\mathbf{x}_j$ to the cluster prototype λ_i, ($i = 1, ..., C$; $j = 1, ..., N$), and $U = [\mu_{ij}]$ is a $C \times N$ matrix of possibilistic membership values. Then U

may be a global minimum for $J_m(L,U)$ only if $\mu_{ij} = \left[1 + \left(\frac{d_{ij}^2}{\eta_i} \right)^{\frac{1}{m-1}} \right]^{-1}$. The necessary conditions on the prototypes are identical to the corresponding conditions in the FCM and its derivatives.

Thus, in each iteration, the updated value of μ_{ij} depends only on the distance of $\boldsymbol{x}_j$ from λ_i, which is an intuitively pleasing result. The membership of a point in a cluster should be determined solely by how far it is from the prototype of the class, and should not be coupled to its location with respect to other classes. The updating of the prototypes depends on the distance measure chosen, and will proceed exactly the same way as in the case of the FCM algorithm and its derivatives.

The value of η_i determines the distance at which the membership value of a point in a cluster becomes 0.5 (i. e., "the 3 dB point"). Thus, it needs to be chosen depending on the desired "bandwidth" of the possibility (membership) distribution for each cluster. This value could be the same for all clusters, if all clusters are expected to be similar. In general, it is desirable that η_i relates to the overall size and shape of cluster λ_i. Also, it is to be noted that η_i determines the relative degree to which the second term in the objective function is important compared to the first. If the two terms are to be weighted roughly equally, then η_i should be of the order of d_{ij}^2 . In practice we find that the following definition works best.

$$\eta_i = \frac{\sum_{j=1}^{N} \mu_{ij}^m \, d_{ij}^2}{\sum_{j=1}^{N} \mu_{ij}^m} \tag{3}$$

This choice makes η_i the average fuzzy intra-cluster distance of cluster λ_i. The value of η_i can be fixed for all iterations, or it may be varied in each iteration. When η_i is varied in each iteration, care must be exercised, since it may lead to instabilities. Our experience shows that the final clustering is quite insensitive to large (an order of magnitude) variations in the values of η_i.

III. The Possibilistic C Plano-Quadric Shells Algorithm

Suppose that we are given a second degree curve λ_i characterized by a prototype vector

$$\boldsymbol{p}_i^{\mathrm{T}} = [p_{i1}, p_{i2}, \ldots, p_{ir}]$$

to which it is desired to fit points $\boldsymbol{x}_j$ obtained through the application of some edge detection algorithm. $\boldsymbol{p}_i^{\mathrm{T}}$ contains the coefficients of the second-degree curve that describes cluster i.. If a point $\boldsymbol{x}$ has coordinates $[x_1, \ldots, x_n]$, then let

$$\boldsymbol{q} = [x_1^2,\ x_2^2, \ldots, x_n^2, x_1x_2, \ldots, x_{(n-1)}x_n, x_1, x_2, \ldots, x_n,\ 1]^T .$$

The equation of the second-degree curve that describes cluster i is given by $\boldsymbol{p}_i^T \boldsymbol{q} = 0$.

When the exact (geometric) distance has no closed-form solution, one of the methods suggested in the literature is to use what is known as the "approximate distance" which is the first-order approximation of the exact distance. It is easy to show [12] that the approximate distance of a point from a curve is given by

$$d_{Aij}^2 = d_A^2(\boldsymbol{x}_j, \lambda_i) = \frac{\delta_{Qij}^2}{|\nabla d_{Qij}^2|^2} = \frac{d_{Qij}^2}{\boldsymbol{p}_i^T \boldsymbol{D}_j \boldsymbol{D}_j^T \boldsymbol{p}_i} , \tag{4}$$

where ∇d_{Qij}^2 is the gradient of the distance functional

$$\boldsymbol{p}_i^T \boldsymbol{q} = [p_{i1}, p_{i2}, \ldots, p_{ir}][x_1^2,\ x_2^2, \ldots, x_n^2, x_1x_2, \ldots, x_{(n-1)}x_n, x_1, x_2, \ldots, x_n,\ 1]^T \tag{5}$$

evaluated at $\boldsymbol{x}_j$. In (4) the matrix $\boldsymbol{D}_j$ is simply the Jacobian of $\boldsymbol{q}$ evaluated at $\boldsymbol{x}_j$.

One can easily reformulate the quadric shell clustering algorithm with d_{Aij}^2 as the underlying distance measure. It was shown in [8] that the solution to the parameter estimation problem is given by the generalized eigenvector problem

$$\boldsymbol{F}_i \boldsymbol{p}_i = l_i \boldsymbol{G}_i \boldsymbol{p}_i , \tag{6}$$

where

$$\boldsymbol{F}_i = \sum_{j=1}^{N} (\mu_{ij})^m \boldsymbol{M}_j ,$$

$$\boldsymbol{M}_j = \boldsymbol{q}_j \boldsymbol{q}_j^T , \text{ and}$$

$$\boldsymbol{G}_i = \sum_{j=1}^{N} (\mu_{ij})^m \boldsymbol{D}_j \boldsymbol{D}_j^T ,$$

which can be converted to the standard eigenvector problem if the matrix $\boldsymbol{G}_i$ is not rank-deficient. Unfortunately this is not the case. In fact, the last row of $\boldsymbol{D}_j$ is always [0, . . . ,0]. Equation (6) can still be solved using other techniques that use the modified Cholesky decomposition [13], and the solution is computationally quite inexpensive when the feature space is 2-D or 3-D. Another advantage of this constraint is that it can also fit lines and planes in addition to quadrics. Our experimental results show that the resulting algorithm, which we call the Possibilistic C Plano-Quadric Shells (PCPQS) algorithm, is quite robust in the presence of poorly defined boundaries (i. e., when the edge points are somewhat scattered around the ideal boundary curve in the 2-D case and when the range values are not very accurate in the 3-D case). It is also very immune to impulse noise and outliers. Of course, if the type of curves required are restricted to a single type, e.g., lines, or circles, or ellipses, simpler algorithms can be used with possibilistic updates, as will be seen.

IV. Determination of Number of Clusters

The number of clusters C is not known *a priori* in some pattern recognition applications and most computer vision applications. When the number of clusters is unknown, one method to determine this number is to perform clustering for a range of C values, and pick the C value for which a suitable validity measure is minimized (or maximized) [14]. However this method is rather tedious, especially when the number of clusters is large. Also, in our experiments, we found that the C value obtained this way may not be optimum. This is because when C is large, the clustering algorithm sometimes converges to a local minimum of the objective function, and this may result in a bad value for the validity of the clustering, even though the value of C is correct. Moreover, when C is greater than the optimum number, the algorithm may split a single shell cluster into more than one cluster, and yet achieve a good value for the overall validity. To overcome these problems, we proposed in [8] an alternative Unsupervised C Shell Clustering algorithm which is computationally more efficient, since it does not perform the clustering for an entire range of C values.

Our proposed method progressively clusters the data starting with an overspecified number C_{max} of clusters. Initially, the FCPQS algorithm is run with $C=C_{max}$. After the algorithm converges, spurious clusters (with low validity) are eliminated; compatible clusters are merged; and points assigned to clusters with good validity are temporarily removed from the data set to reduce computations. The FCPQS algorithm is invoked again with the remaining feature points. The above procedure is repeated until no more elimination, merging, or removing occurs, or until $C=1$.

V. Examples of Possibilistic Clustering for Shape Recognition

Figures 1 and 2 show the detection of a circular "fractal edge" from a synthetically generated image. Figure 1(a) is the original composite fractal image; Figure 1(b) shows what a gray-scale edge operator finds (or doesn't find); figure 1(c) is the output of the horizontal fractal edge operator; with Figure 1(d) giving the maximum overall response of the fractal operators in four directions. Figure 2(a) depicts the (noisy) thresholded and thinned result from Figure 1(d). Figure 2(b) gives the final prototype found by the FPQCS (which, since there is only one cluster present, is the same as the crisp version). Note how the presence of noise distorts the final prototype. Figure 2(c) shows the possibilistic algorithm output, which is superimposed on the original image in Figure 2(d). The results of the PPQCS algorithm are virtually unaffected by noise. Several examples comparing crisp, fuzzy and possibilistic versions of clustering can be found in [6,8,10].

Figure 3 depicts the algorithm applied to the image of a model of the Space Shuttle. Figure 3(a) is the original image. Figure 3(b) gives the output of a typical edge operator. Note that, due to the rather poor quality of the original image, the edges found both noisy and incomplete. This data was then input into the possibilistic plano-quardic clustering algorithm. Figure 3(c) gives the eight complete prototypes which were found after running the algorithm. Finally, Figure 3(d) displays the prototype drawn only where sufficient edges points exist.

VI. Conclusions

In this paper, we demonstrated how our new possibilistic approach to objective-function-based clustering coupled with our plano - quadric shells algorithm can recognize first and second degree shapes from incomplete and noisy edge data. This approach is superior to both crisp and fuzzy clustering, as well as to traditional methods such as the Hough Transform. Extensions of this approach to other classes of shapes is currently underway.

Acknowledgment

We are grateful to our students Hichem Frigui and Olfa Nasraoui without whose suggestions and assistance the simulation experiments would not have been possible.

VII. References

1. R. N. Dave, "New measures for evaluating fuzzy partitions induced through C-shells clustering", *Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision X: Algorithms and Techniques*, Boston, Nov. 1991, pp. 406-414.

2. R. N. Dave, "Fuzzy-shell clustering and applications to circle detection in digital images", *International Journal of General Systems*, vol. 16, 1990, pp. 343-355.

3. R. N. Dave, "Adaptive C-shells clustering", *Proceedings of the North American Fuzzy Information Processing Society Workshop*, Columbia, Missouri, 1991, pp. 195-199.

4. J. C. Bezdek and R. J. Hathaway, "Accelerating convergence of the Fuzzy C-Shells clustering algorithms", *Proceedings of the International Fuzzy Systems Association Congress*, Brussels, July 1991, Volume on *Mathematics*, pp. 12-15.

5. R. Krishnapuram, H. Frigui, and O. Nasraoui, "New fuzzy shell clustering algorithms for boundary detection and pattern recognition", *Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision X: Algorithms and Techniques*, Boston, Nov. 1991, pp.458-465.

6. R. Krishnapuram, O. Nasraoui, and H. Frigui, "The fuzzy C spherical shells algorithm: A new approach", to appear in the *IEEE Transactions on Neural Networks.*

7. R. N. Dave and K. Bhaswan, "Adaptive fuzzy C-shells clustering and detection of ellipses", to appear in the *IEEE Transactions on Neural Networks*, vol. 3, no. 5, 1992.

8. R. Krishnapuram, H. Frigui, and O. Nasraoui, "A Fuzzy Clustering Algorithm to Detect Planar and Quadric shapes", submitted to Proceedings NAFIPS'92.

9. J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.

10. R. Krishnapuram and J. M. Keller, "A possibilistic approach to clustering", submitted to *IEEE Transactions on Fuzzy Systems.*

11. D. Dubois and H. Prade, *Possibility Theory: An Approach to Computerized Processing of Uncertainty*, Plenum Press, New York, 1988.

12. R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, vol. I, Addison Wesley, Reading, MA, 1992, Chapter 11.

13. G. Taubin, "Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 11, Nov. 1991, pp. 1115-1138.

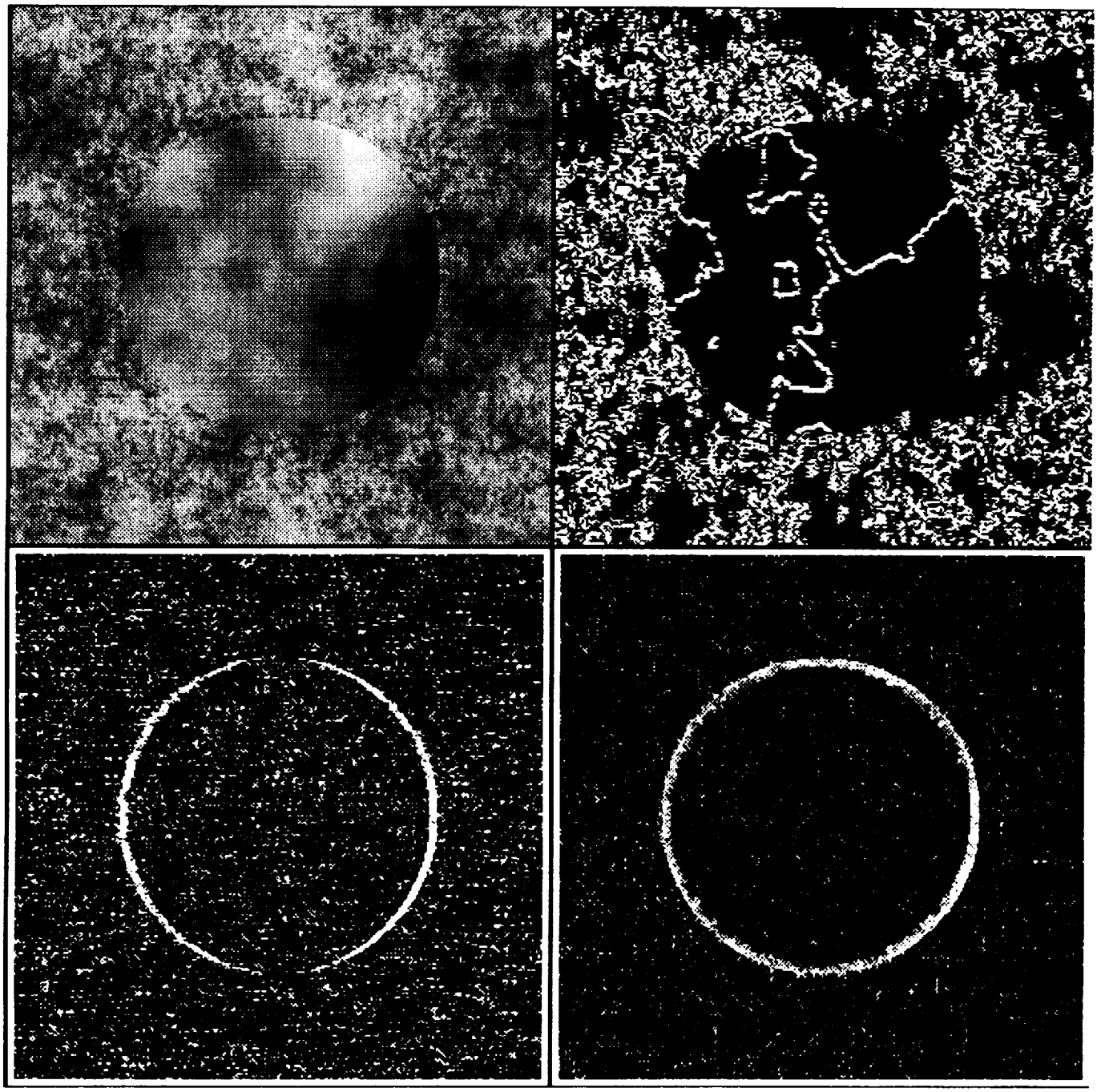


Figure 1. Detection of a fractal circular edge.
(a) Upper Left. Original fractal composite image.
(b) Upper Right. Output of gray scale edge operator.
(c) Lower Left. Output of "horizontal" fractal edge operator.
(d) Lower Right. Results of Maximum magnitude of outputs of four directions of fractal operators.

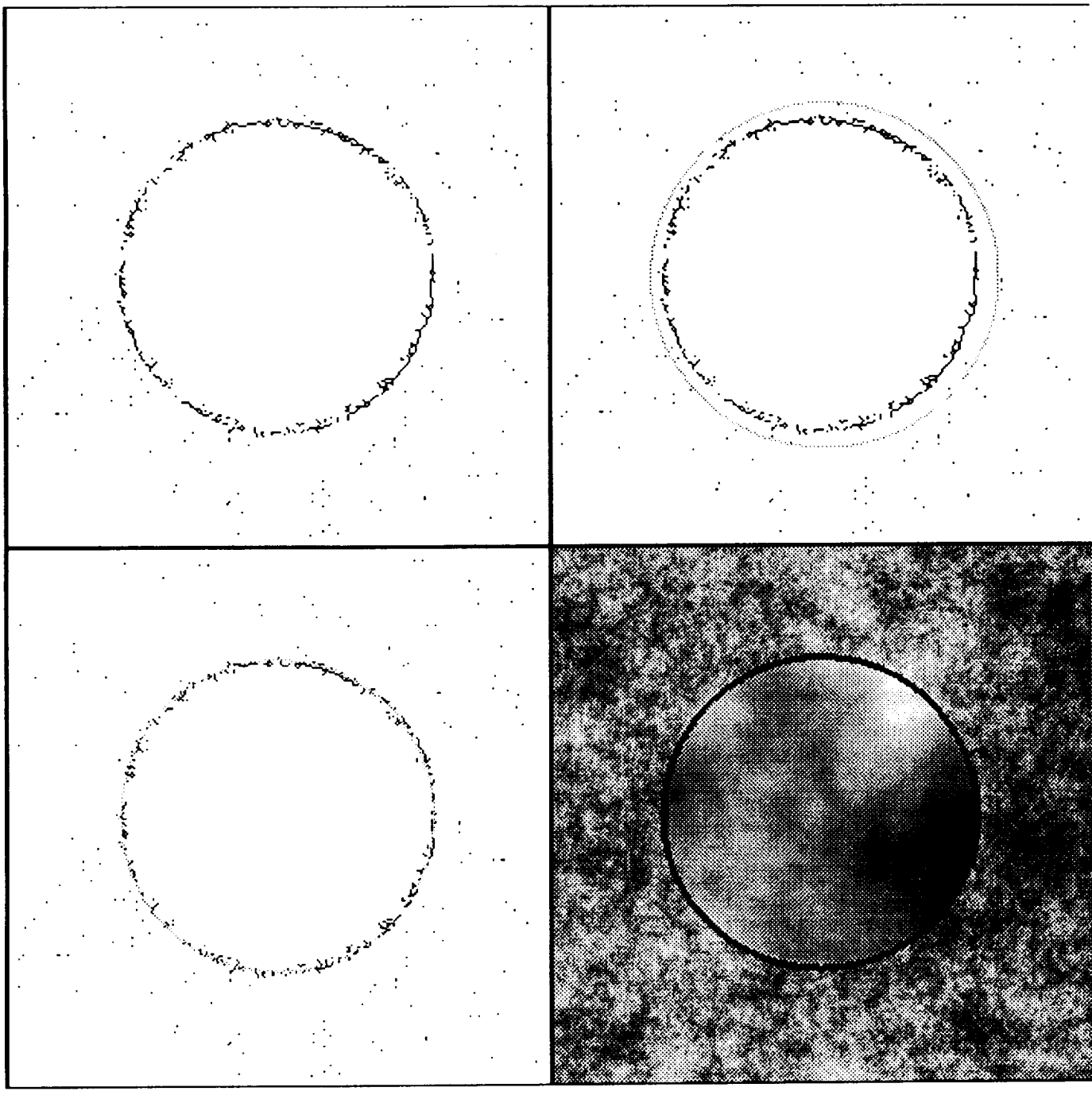


Figure 2. Recognition of circular boundary.
(a) Upper Left. Figure 1(d) thresholded and thinned.
(b) Upper Right. Circular prototype found by fuzzy (or crisp) clustering.
(c) Lower Left. Circular prototype found by possibilistic clustering.
(d) Lower Right. Possibilistic prototype superimposed on original image.

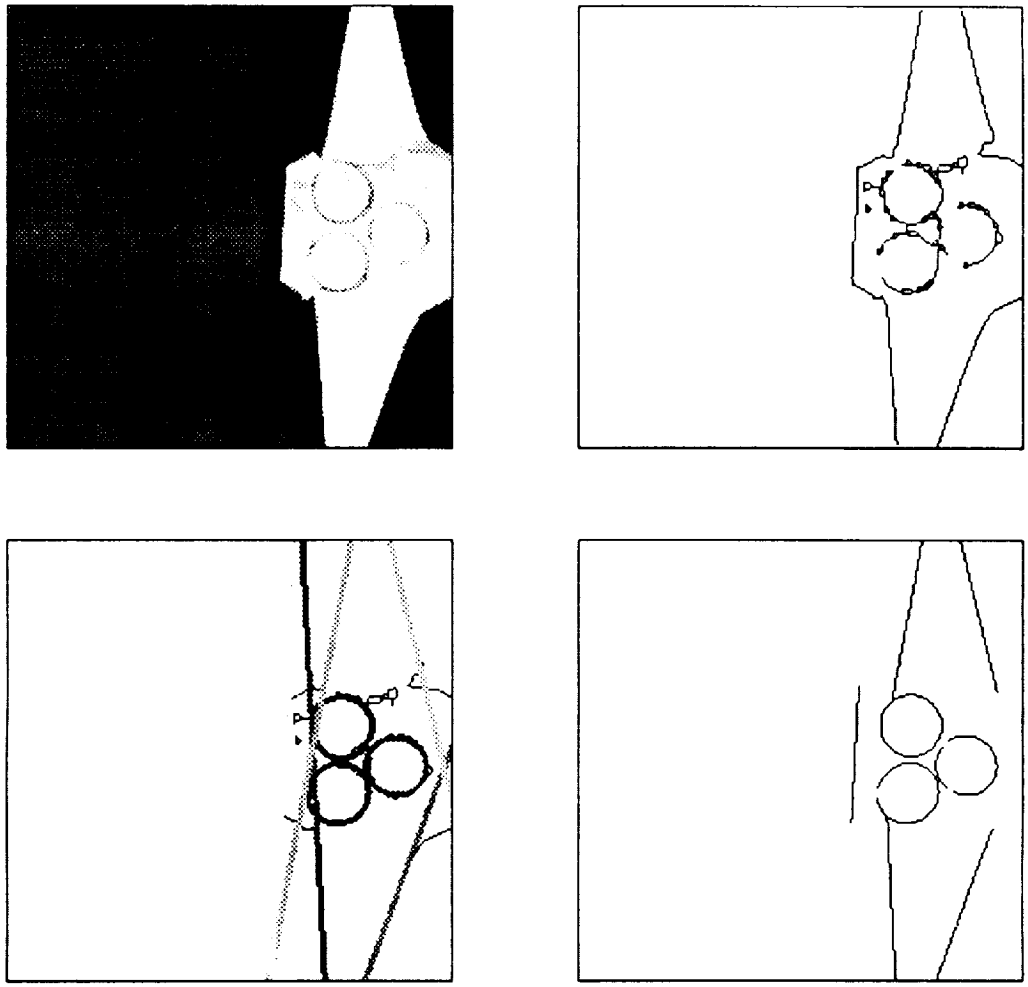


Figure 3. Recognition of Shuttle model boundaries.
(a) Upper Left. Original Shuttle image.
(b) Upper Right. Incomplete and noisy edges found by edge operator.
(c) Lower Left. Prototypes found by Possibilistic Plano-Quadric clustering.
(d) Lower Right. Possibilistic prototypes superimposed drawn where there is sufficient edge information.

Application of Genetic Algorithms to Tuning Fuzzy Control Systems

Todd Espy
Endre Vombrack
Jack Aldridge

Togai Infralogic, Inc.
17000 El Camino Real, # 204C
Houston, TX 77058

Abstract

Real number genetic algorithms (GA) have been applied for tuning fuzzy membership functions of three controller applications. The first application is our "Fuzzy Pong" demonstration, a controller that controls a very responsive system. The performance of the automatically tuned membership functions exceeded that of manually tuned membership functions both when the algorithm started with randomly generated functions and with the best manually-tuned functions. The second GA tunes input membership functions to achieve a specified control surface. The third application is a practical one, a motor controller for a printed circuit manufacturing system. The GA alters the positions and overlaps of the membership functions to accomplish the tuning. This paper discusses the applications, the real number GA approach, the fitness function and population parameters, and the performance improvements achieved. Directions for further research in tuning input and output membership functions and in tuning fuzzy rules are described.

Introduction

A significant task in building fuzzy control systems is tuning the membership functions (MBFs) to improve or optimize the performance of the controller. The tuning task has been accomplished with fuzzy systems[1], neural networks[2], and genetic algorithms[3] (GAs). In this paper, we describe the use of real number genetic algorithms[4] to successfully tune membership functions for several fuzzy control systems. A significant feature of this work is that the input MBFs are tuned whereas many previous efforts have concentrated on tuning the output MBFs. Because both input and output membership functions are required to define the control surface for the fuzzy controller, this offers an added degree of

flexibility to the tuning process. Whether such flexibility is, in fact, beneficial to fuzzy controller tuning is yet to be determined.

We first describe some aspects of real number genetic algorithms because that representation of genetic algorithms is less familiar than others. Next, we describe an application of matching a predefined control surface by tuning membership functions for the inputs. Third, we discuss the fuzzy pong application, a controller for an air flow driven by a fan and balancing a ping pong ball at a set position in a plastic tube. Fourth, we briefly discuss results for applying the technique to an AC servomotor control system. We then conclude with remarks about future directions.

Real Number Genetic Algorithms

Many genetic algorithm applications and theorems are based on bit string representations in which the parameters to be optimized are encoded in binary numbers, concatenated, and treated for GA manipulations as one continuous bit string. In tuning fuzzy membership functions, we found it more useful to keep the real number representation for the parameters of the MBFs and to manipulate the numbers using crossover and mutation techniques suitable to the real number representation[4].

Fig. 1 shows the representation of a collection of parameters as a list of real numbers. For the applications discussed below, we used five symmetric triangular membership functions with two parameters each, namely, the upper and lower ends of the support, for each universe of discourse. The fact that we need to represent pairs of ordered numbers favors the real number representation. We used twenty individuals in our populations, for convenience.

Because real number GAs are not extensively used, a standard set of operators is not yet defined. Fig. 2 illustrates our genetic algorithm operators for real number GAs: merge, crossover, mutate, and creep. Merge averages the parameters of two individuals to form the offspring. Crossover exchanges the real numbers between two fit individuals, pairwise. For the problem with two MBFs, the net effect is to replace left or right extents of the MBFs between fit individuals to concentrate the best combinations within a single individual. Presumably, the other individual would lose in the fitness evaluation during the next cycle. Mutate begins by selecting which fuzzy variable is to be selected on a random draw. For our case of two fuzzy input variables, the probability was 50-50 of selecting either one of the MBFs. Having selected the MBF, we perturb its

parameters by randomly selected magnitudes. Creep is an operation in which all parameters of an individual are randomly perturbed. Creep is a hybridizing operation well-suited for search in the local area of an individual if the random variations are limited to some maximum. Our process used 5 individuals mated with the most fit of a generation by crossover, 5 most fit individuals mutated, 5 merged individuals from a pairwise competition, and 5 new individuals selected by random draw as the basis for choosing a new generation. A variant of the creep operator was used in later generations.

The input membership functions are symmetrical and described by an upper and lower end of the support. The peak of the triangular shape is midway between these extremes and has membership value of one. The controller we used was a two-input one output generic controller that could be customized to the application. The simplest interpretation is error and error_rate for the two inputs and control for the output. This interpretation varies from application to application as in the control surface generator described in the next section. With five MBFs for each fuzzy variable, the input MBFs are characterized by 20 numbers, the size of an individual in our population. Fig. 1 illustrates the correspondence between the MBF support parameters and the GA individuals.

Matching a Control Surface

The simplest of the tuning applications we performed was the tuning of membership functions to match a prespecified control surface. Although the control surface for a controller is generally not known a priori, in those cases where it is, GA tuning may be useful. One example of such a case might be the operation of a plant by an operator in which the control commanded manually is recorded with the plant sensors. Such relations would define a partial control surface that might be encoded in a fuzzy controller.

To illustrate the capability to tune to a given control surface, we tuned the MBFs of the inputs to a two-input(x,y), one-output(z) controller to match a control surface $x^2 + y^2 = 10z$. The fitness criterion was the sum of squares of differences between the predicted output for the controller and $(x^2 + y^2)/10$. The parameters of the GAs were adjusted to minimize the mean square error between these quantities over the control surface as measured at 121 points chosen in a square pattern across the center of the x-y plane.

Fig. 3 illustrates the performance for several randomly chosen starting populations. The mean square error converges rapidly with generation number. The best fits we have observed converge to approximately 15 on the same fitness scale. This suggests that the effects of local minima are significant and that knowledge of good initial membership functions will greatly assist convergence to optimal controllers.

The Fuzzy Pong Controller

The fuzzy pong is a controlled plant consisting of a ping-pong ball suspended on a column of air provided by a small fan whose voltage is controlled by the fuzzy controller or a proportional-integral-derivative (PID) controller. (The choice is made by which code is loaded into the microcontroller memory.) The ball's location in the plastic tube is determined using an ultrasonic acoustic range sensor located at the bottom of the tube. The servocontroller function is provided by a Hitachi H8/325 microprocessor board that drives a conventional transistor amplifier that serves as the DC voltage control for the motor voltage. The set point for control is provided to the H8 by an external personal computer (PC) that also is used as a monitor and data display device. There are two set points provided by the PC: high and low set points. When the ping pong ball stabilizes its position within user defined limits about either set point for a time preset by the user, the PC commands traversal to the other set point. The fuzzy controller commands the fan voltage based on the error = (set point - ball location) and the rate of change of error = (error(t) - error(t-1)), where t is the current time in units of the sample interval. The ability of the fuzzy controller to provide more precise control than the PID had been previously established through manual tuning to achieve smallest time transitions with minimal overshoot.

The GA tuning used a fitness function that measures the number of successful transitions, up to four, that an individual can accomplish, the rise time achieved in those transitions, and the overshoot that the transitions possess. If an individual cannot achieve success in stabilizing the ball within a predetermined time, the evaluation of the fitness is terminated. The achieving of the set point within a time limit allows the evaluation of other factors and offers a chance to try again up to four attempts. The fitness is evaluated using the hardware and is thus not deterministic because of the sensitivity of the pong to ball spin, initial position, air temperature, etc. The

fitness over a sequence of populations thus may not monotonically decrease, even if the best individual from the preceding generation is kept to assure monotonicity.

Fig. 4 illustrates the fitness of the best individual in a generation as a function of generation. There is some improvement within a level established by success in finding the set point. The fitness is clearly dominated by the success in achieving the set point. The loss of a best individual also clearly limits system performance considerably. A strategy for handling this contingency such as requiring a number of generations before a best individual can be omitted might be useful. Development of an improved fitness criterion that places less emphasis on the number of sequential successes - perhaps running a fixed number of trials for each individual - would allow better discrimination of the transition characteristics. Achieving the commanded set point would need to continue to play an important role, however.

Motor Controller Tuning

We conducted experiments on tuning a fuzzy controller for an AC servomotor. The controller has been previously described[5]. It is a fuzzy PD controller capable of either control of the angular rate or the angular position. The controller exhibits "deadbeat" performance[6] - rapid response to unit step input without overshoot - that is faster than critically damped PID control.

This is an application in which tuning the input MBFs is particularly appropriate because the gains on the proportional and velocity controls are determined by MBF placement. The overall control gain achievable by tuning output MBFs alone does not provide the same ability to trade off between error and error rate that the input MBF tuning provides.

The GA tuning was able to tune a controller from a random starting population to a controller with performance equal to a laboriously tuned manual case within 5 generations. In only one case did a manually tuned controller exceed the performance of the GA tuned controllers.

Further Research

There are extensions to the techniques described here that are needed to fully evaluate the utility of this technique to tuning in general. First, the restriction of the population to twenty individuals

needs to be relaxed. Second, the operators need to be chosen randomly with parameters to determine how often the operators should occur in the random choice, similar to the practice in bit string based GAs. Third, in cases where the best individual from a previous generation may not evaluate to the same fitness value, the "fencing" of the individual to prevent loss of his data from the pool may be useful[7]. Fourth, the usefulness of using three (or more) parameters to describe a MBF should be explored. This would allow asymmetric MBFs. Such flexibility would be useful in permitting variable gain systems in which the placement of the center of adjacent MBFs determines the gain and the extent of the MBF is determined by the location of the center of the closest MBF to one of these. The effect of limiting the extent to half a support is to make the gain zero over that interval. Fifth, addition of search techniques that would allow local optimization of fitness before comparison could be useful. In a real number space, such techniques, subject to restrictions that will be applied to the resulting individuals (e.g., that the membership function's center must lie between the two ends of the support), should permit more rapid convergence of the GA search.

Summary

We have shown the applicability of real number genetic algorithms to the problem of automated tuning of membership functions for fuzzy controllers. The application tunes input membership functions which is a matching of control regions to the controller rather than the adjustment of gain of the controller. In a practical system, retention of the best individual may not assure monotonic convergence due to noise in the fitness function evaluation.

The GA search is most effective for tuning the controller in circumstances such as simulation when the failure of a system is inconsequential. For applications in which the stability of control must be maintained, such as automatic optimization of performance of an autonomous system, the applicability of a global search mechanism is questionable if the evaluation of fitness depends on controlling the device.

Acknowledgement
This work was sponsored in part by NASA Johnson Space Center under contract number NAS9-18527. We wish to thank Dr. Robert Lea for his support and encouragement of these efforts.

References
1. "Performance Evaluation of a Self-Tuning Fuzzy Controller," Walter C. Daugherity, Balaji Rathakrishnan, and John Yen, p. 389, Proceedings of the IEEE International Conference on Fuzzy Systems 1992 (FUZZ-IEEE 1992), San Diego, CA, March 8-12, 1992
2. "Neural Networks and Fuzzy Logic in Intelligent Control," Hamid Berenji, p.916, Proceedings of the 5th International Symposium on Intelligent Control 1990, Philadelphia, PA, September 5-7, 1990
3. "Fuzzy Logic and Genetic Algorithms in Time Varying Control Problems," C.L. Karr and D.A. Stanley, p. 285, 1991 Workshop Proceedings of the North American Fuzzy Information Processing Society, University of Missouri-Columbia, May 14-17, 1991
4. Handbook of Genetic Algorithms, ed. by Lawrence Davis, Ch. 5, (Van Nostrand Reinhold, New York, 1991)
5. Ron Maor and Yashvant Jani, p. 155, Working Notes of First International Workshop on Industrial Applications of Fuzzy Control and Intelligent Systems,Texas A&M University, College Station, TX, November 21-22, 1991
6. Julius Tou, Modern Control Theory, p.118 (McGraw-Hill Book Company, New York, 1964)
7. "Robot Path Planning Using a Genetic Algorithm," Timothy F. Cleghorn, Paul T. Baffes, and Lui Wang, p. 383, Proceedings of the Second Annual Workshop on Space Operations Automation and Robotics (SOAR '88) Dayton, OH, July 20-23, 1988 (NASA Conference Publication 3019)

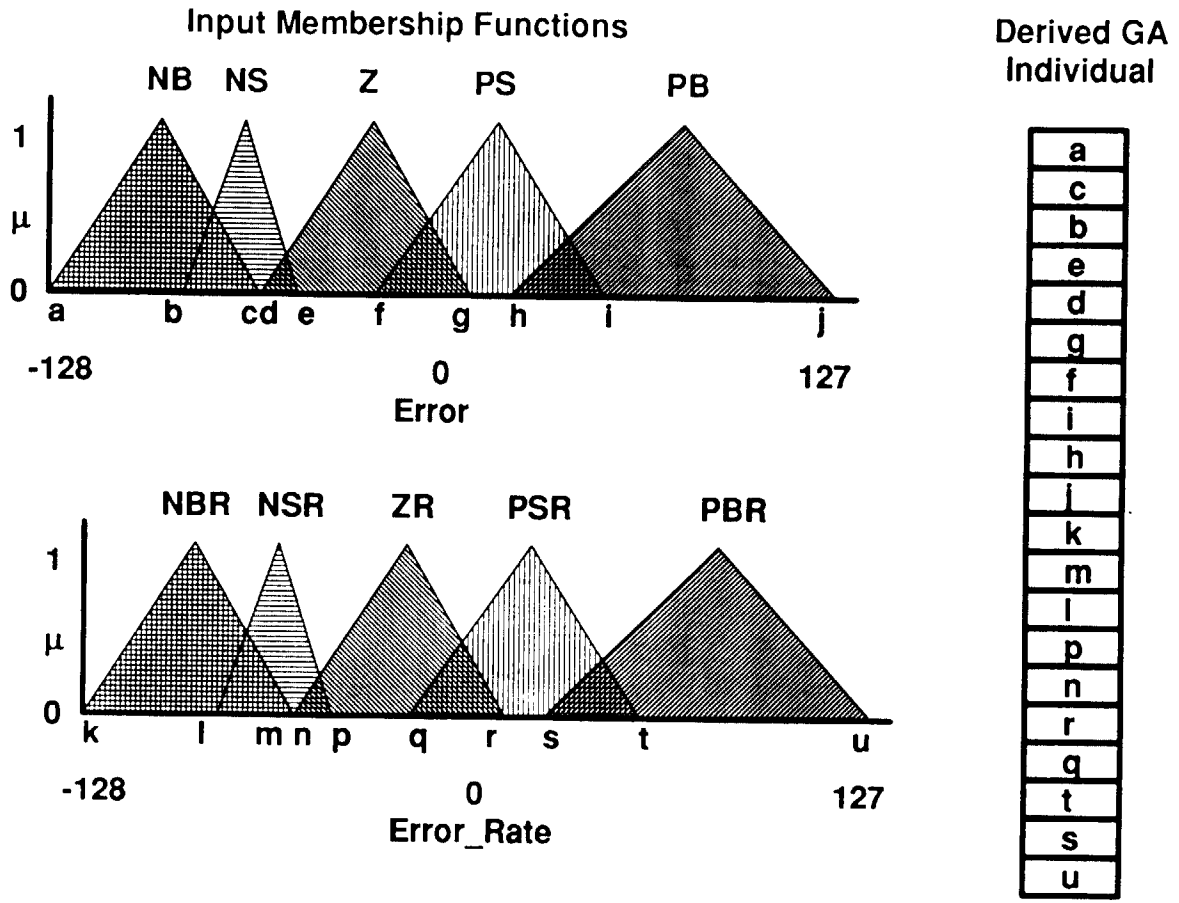
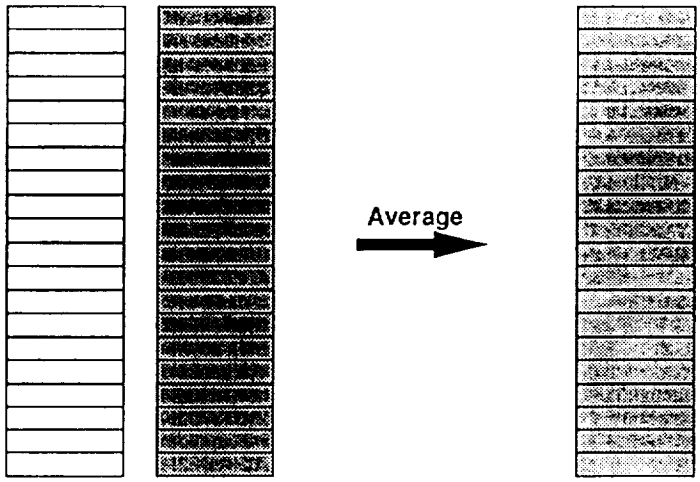


Fig. 1 Correspondence between input membership functions and the fields of an individual for the genetic algorithm tuner. The small letters represent integer values from the universe of discourse giving the ends of the support for each symmetrical membership function.

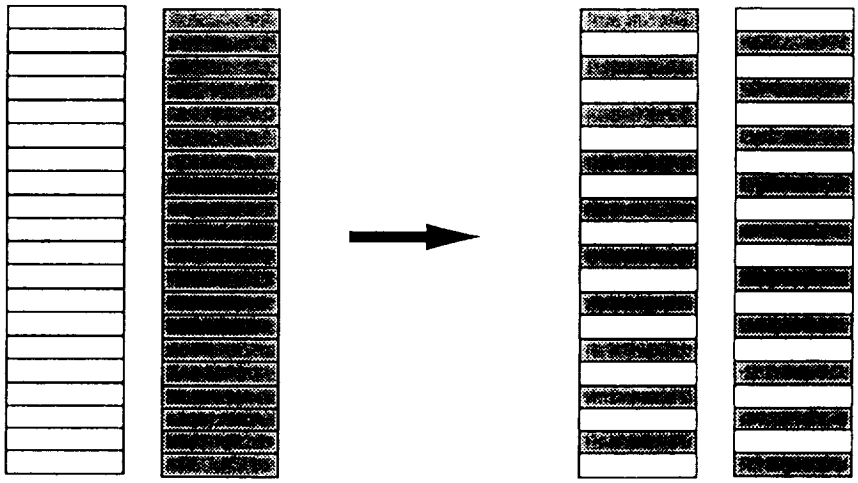
FUZZY GENETIC ALGORITHM MERGE



USE MOST FIT TO BREED BY PAIRWISE COMPETITION

(a)

REAL NUMBER GENETIC ALGORITHM CROSSOVER

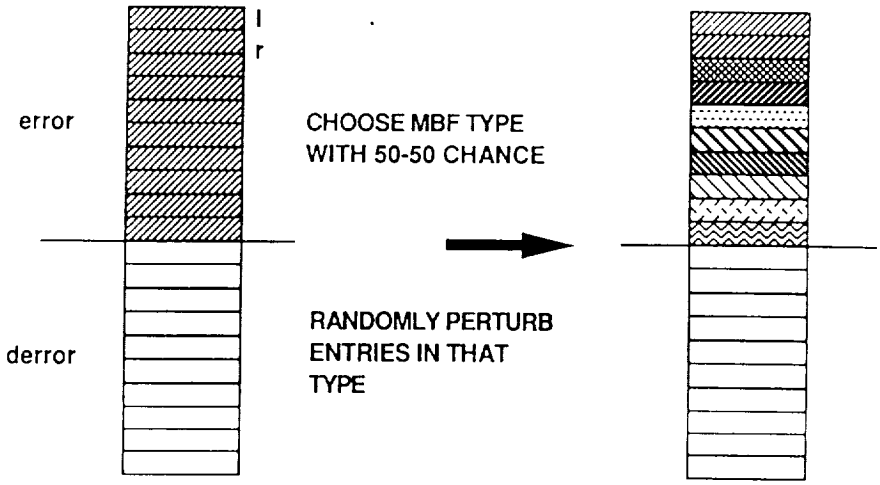


SMALL CHANGES ON MINIMUM SURFACE BRINGS CLOSER TO MAXIMUM

(b)

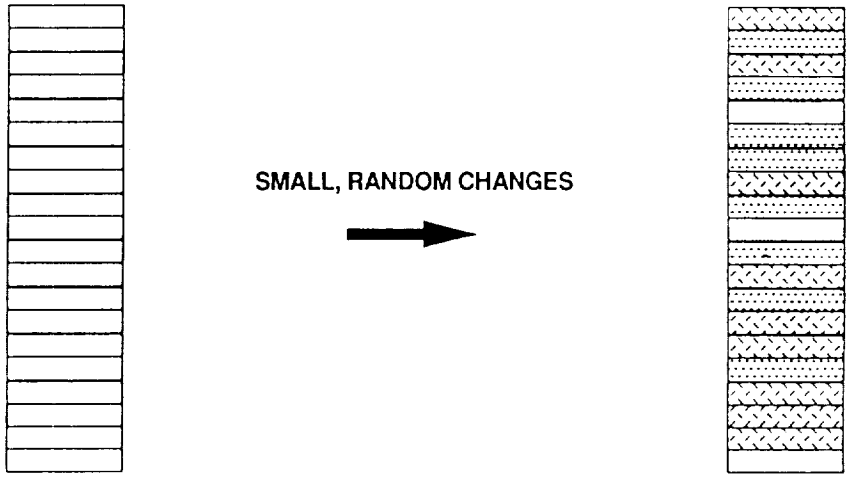
Fig. 2 Real number genetic operators defined for this tuning process

FUZZY GENETIC ALGORITHM MUTATE



(c)

REAL NUMBER GENETIC ALGORITHM CREEP



SMALL CHANGES ON MINIMUM SURFACE BRINGS CLOSER TO MAXIMUM

(d)

Fig. 2 (cont'd) Real number genetic operators

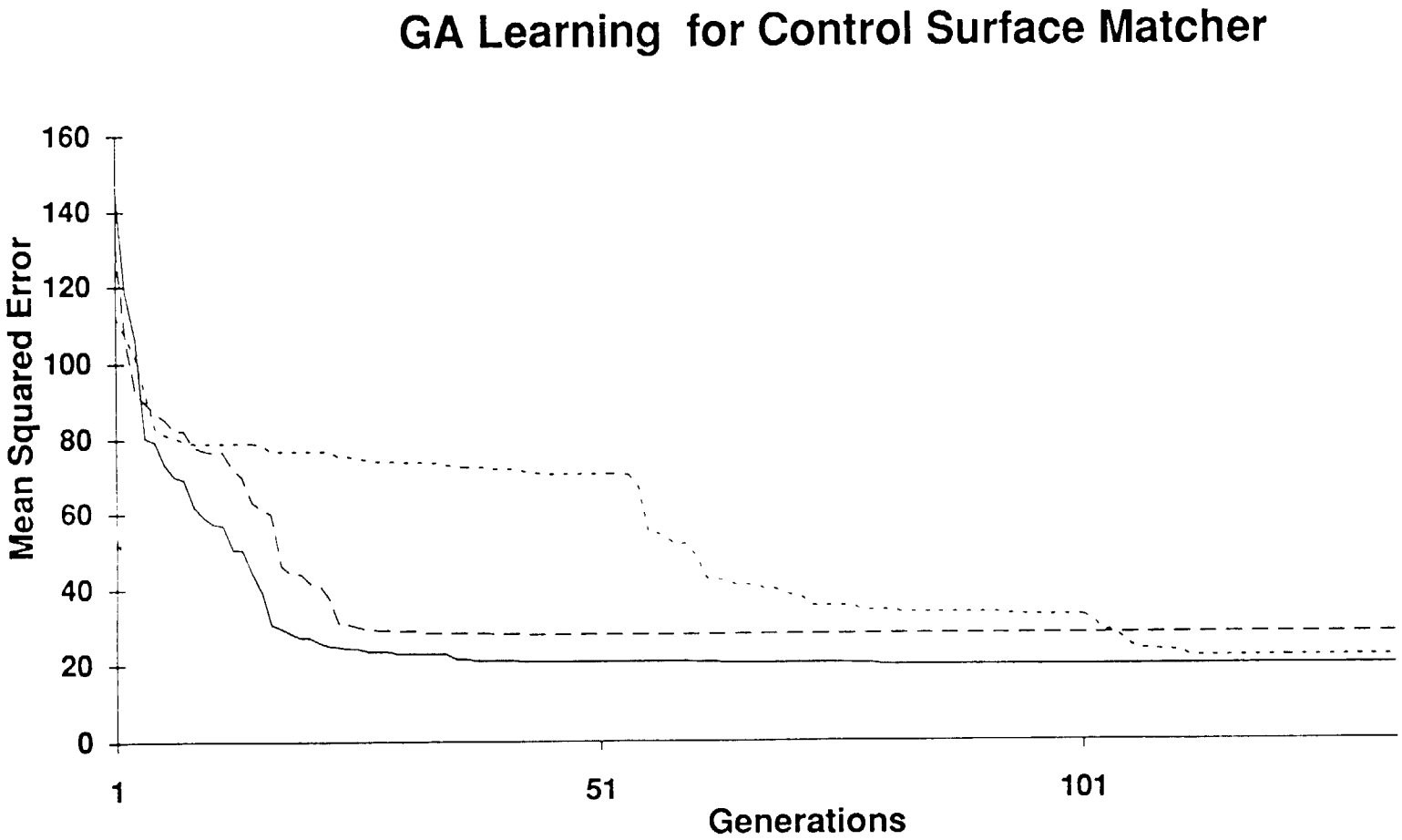


Fig.3 Results for control law learning problem

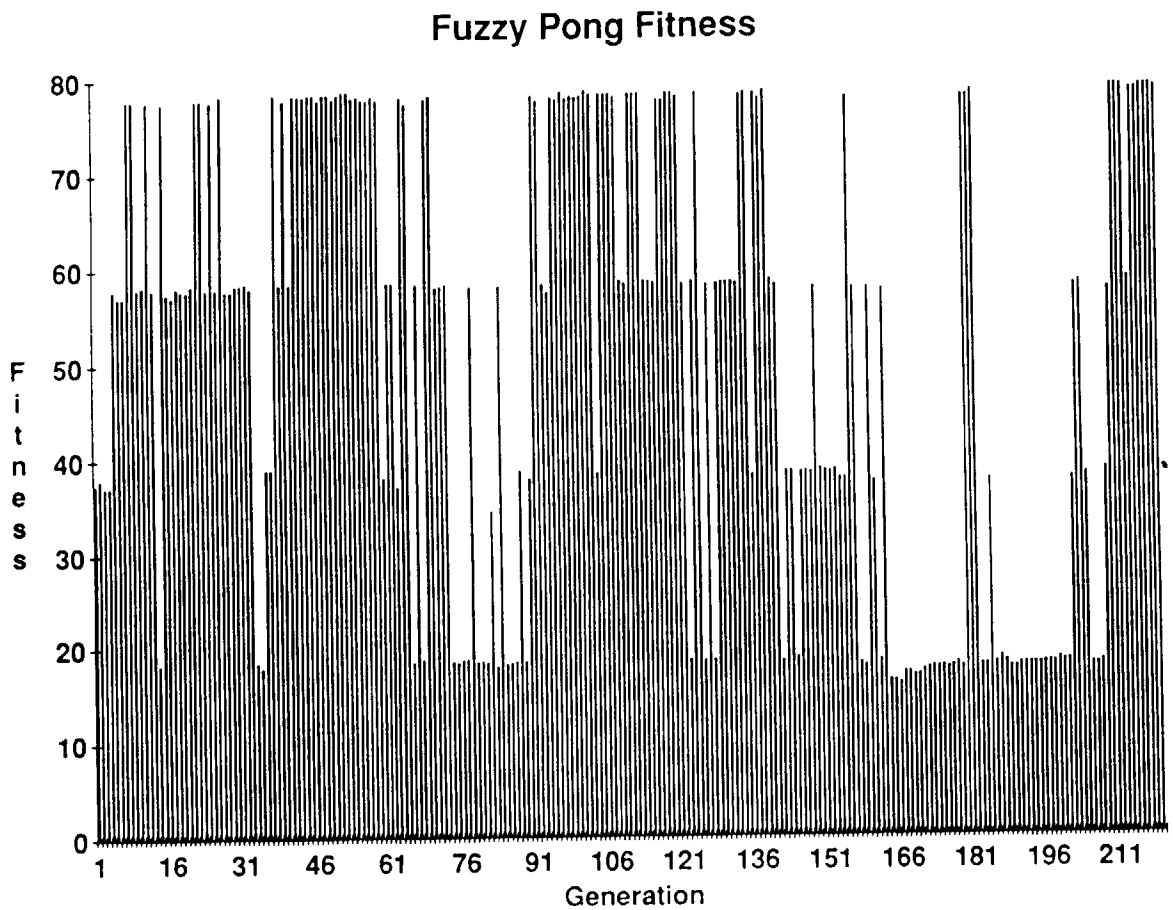


Fig.4 Results for air flow controller tuning

3D Image Recognition Based on Fuzzy Neural Network Technology

Kaoru HIROTA, Kenichi YAMAUCHI, Jun MURAKAMI, Kei Tanaka
Dept. of Instrument & Control Engineering, College of Engineering
Hosei University
3-7-2 Kajino-cho, Koganei-city, Tokyo 184, Japan

ABSTRACT

3-D stereoscopic image recognition system based on fuzzy-neuralnetwork technology has been developed. The system consists of 3 parts; preprocessing part, feature extraction part, and matching part. Two CCD color camera images are fed to the preprocessing part, where several operations including RGB-HSV transformation are done. A multi-layer perceptron is used for the line detection in the feature extraction part. Then fuzzy matching technique is introduced in the matching part. The system is realized on SUN spark station and special image input hardware system. An experimental result on bottle images is also presented.

keywords: 3D image recognition, fuzzy matching, neural network

1.Introduction

The recent development of image processing and pattern recognition technology is remarkable. Many are put into the practical use in fields of industrial testing system, remote sensing and so on. It is difficult, however, to make a flexible vision system based on human experiences and human skilled knowledge. On the other hand, fuzzy logic and neural network technology are applied over a lot of fields including the control and the image recognition, where a human like processing is introduced. By combining the both techniques, 2-D image recognition system has been realized and reported [1].

In this paper a newly developed image recognition system using the technique of the binocular vision with fuzzy neuralnetwork methodology is presented. The system is realized on SUN spark station and special image input hardware system with 2 CCD color cameras. It consists of the following 3 parts; the preprocessing part where RGB-HSV transformation and other operations are done, the feature extraction part where a multi-layer perceptron is used for the line detection, and the matching part where a fuzzy matching algorithm is introduced. Finally several experimental results on bottle images are presented in order to confirm the availability of the proposed system.

2.Image recognition process

Fig.1 shows the outline of the presented image recognition process. It is roughly divided into 3 parts.

In the preprocessing part binocular images of each 512*512 pixels are taken by using 2 CCD color cameras. Several ordinary image processing operations and the concept of color fuzzy set are introduced in order to satisfy the quality requested in the feature extraction part. Then the contour features are extracted in the feature extraction part by using the multi layer type perceptron and the factorization technique based on fuzzy logic.

Finally a fuzzy matching algorithm is introduced in the matching part, where the result is presented in terms of fuzzy set.

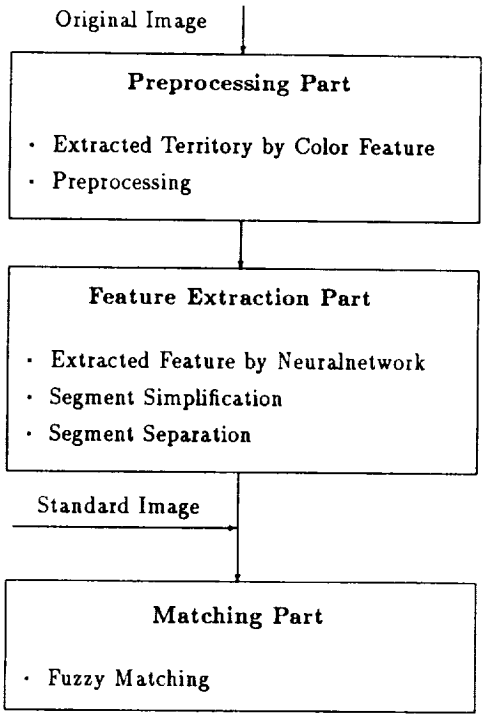


Fig.1 The outline of image recognition process

3.Colored region extraction based on color information

The input image from the camera is expressed by combining the RGB (Red, Green, and Blue) density. In the case of humanbeings color information is transmitted and is qualitatively recognized from eyes to a large brain (perception center). And a lot of models are proposed to explain the process. Here HSV (Hue, Saturation, and Value) hexagon cone color model [2] is used by introducing the RGB-HSV conversion. So the RGB color information is converted into three attributes of the hue, the saturation, and the value, where three attributes are defined by the membership functions which are shown in Fig.2.

When humanbeings extract the color features, the color distribution/tendency of the entire image is considered. For instance, when the image observed is composed of rather similar colors, then the color range to be recognized is set to be narrowed. Such characteristics are expressed by fuzzy rules. An example of fuzzy rules is shown below. By introducing the fuzzy matching technique, the feature colors are extracted.

[one example of fuzzy rules of feature color extraction]

IF the hue of the object is closely distributed
THEN the membership function of the hue should be narrowed.

4.Line segmentation using multi-layer perceptron

An image based on color information is converted to an line drawing image by using ordinary image processing technique. A multi-layer perceptron is applied to scanning the line drawing

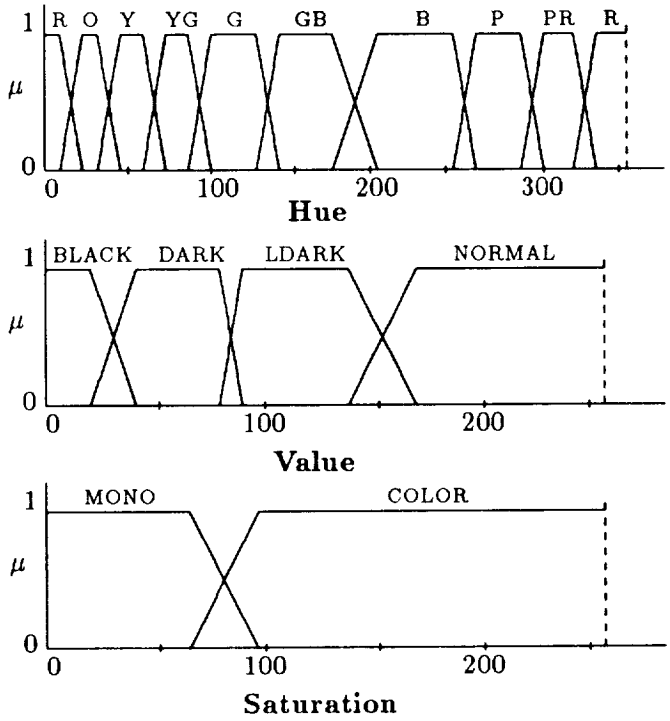


Fig.2 Membership functions of three color attributes

image and extract the directionality of lines. Here it should be noted that the non-learning data is generalized by the learning data in the Back Propagation Method of the perceptron model [3][4].

The input layer in the multi-layer perceptron corresponds to a part of the image. The teaching pattern in the learning process is a set of the typical lines of the input pattern, and the output pattern is the direction representing code called a chain code.

The outline of the multi-layer perceptron is summarized as follows: The output of the i-th neuron in the first layer is

$$u_i^l = f(s_i^l), \tag{1}$$

where

$$f(s) = \frac{1}{1+\exp(-s)}, \tag{2}$$

$$s_i^l = \sum_{j=1}^{N_{l-1}} w_{ij}^{ll-1} \cdot u_j^{l-1} - \theta_i^l . \tag{3}$$

The input of the (i,j) coordinate in the input frame is

$$u_i^1 = x(i\%W, i/W), \tag{4}$$

where % and / stand for the remainder and the quotient of division, respectively. The evaluation is given by

$$E = \frac{1}{2}\sum_{i=1}^{N_L}[u_i^L - y_i]^2, \tag{5}$$

where y_i stands for the value of the i-th neuron in the teaching pattern.

The input frame has a variable ratio which is determined by the ratio of the dark pixels to the bright pixels in the input frame. The position of the input frame is slightly changed in order to adjust the position of the center of gravity of the dark pixels to the middle of the input frame. The coordinate (G_x, G_y) of the center of the gravity of the dark pixels in the input

frame, and the ratio S of the dark pixels to the bright pixels is calculated as follows;

$$G_x = \frac{\sum_{j=1}^{H}\sum_{i=1}^{W} x(i,j)\cdot i}{H\cdot\sum_{i=1}^{W} x(i,j)} \quad (6)$$

$$G_y = \frac{\sum_{i=1}^{W}\sum_{j=1}^{H} x(i,j)\cdot j}{W\cdot\sum_{j=1}^{H} x(i,j)} \quad (7)$$

$$S = \frac{\sum_{j=1}^{H}\sum_{i=1}^{W} x(i,j)}{H\cdot W} \quad (8)$$

Based on these input values the output of the multi-layer perceptron is calculated. By moving the position of the input frame taking the output value into the consideration the line segment is traced. In the branching point of the line segment the input frame of the multi-layer perceptron also makes a branch and the line segment is traced in parallel.

5.The simplification of the extracted line segment data series

The output data series obtained by the multi-layer perceptron represents the directionality of the line segment. It is classified into a group of straight line, curved line, and corner by using the membership function shown in Fig.3.

The simplified data represent the geometrical feature of the input image. They can be understood in many ways with membership value.

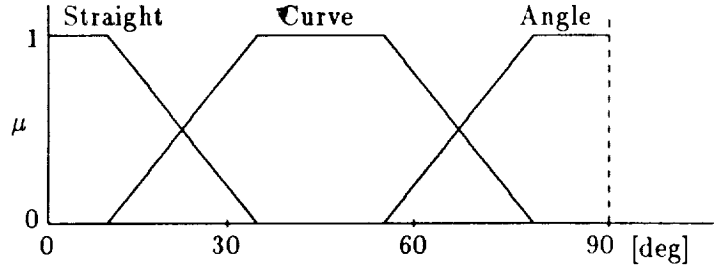


Fig.3 Membership function of geometrical features

6.Binocular stereoscopic vision

6.1 Correspondence between left and right images

Three dimensional binocular stereoscopic vision is realized by using in principle the difference between left and right images. But it is not so easy to make a correspondence of characteristic points between two images.

In this study the correspondence is made by using the simplified data series of both images mentioned in section 5. The line segment correspondence can be made based on the distance between line segments[5]. The both images are divided into several line segment blocks and the similarity between the blocks are

calculated which generates sub blocks of line segments. Such a procedure continues and finally the correspondence of line segments are obtained, where fuzzy logic is introduced especially in the representation of the shape of line segment (c.f. Fig. 3) and correspondence operation.

6.2 Separation of objects

The distance between the camera and the object is calculated based on the information of line segment correspondence obtained in 6.1, where the method in projective geometry [6] are introduced. The position [a] in the 3D space and its corresponding position [b] in the image is connected by a translation matrix M

The matrix M can be calculated from the data of 6 points. Then the both images are transformed into 3D space by applying this translation matrix M. By doing a clustering procedure in 3D space the contour line of each object is extracted.

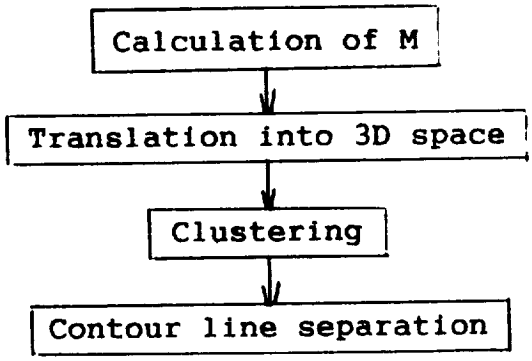


Fig.4 Contour line separation in 3D space

7. Fuzzy matching

Humanbeings can recognize the target object to some extent even if some part of it is hidden. Such functions are realized here by introducing fuzzy matching technique. The recognition result here is the similarity between the extracted information mentioned in the section 6 and the standard pattern information.

The standard pattern information consists of the type of line, the coordinates of starting point and end point, the length in the case of straight line, the curvature and the angle in the case of curved line and corner, and so on.

Firstly the segment with the minimum y coordinate is found and is checked if it is the top part of the object by observing the left and the right segments. Then the data series are divided into two parts. The similarity of the segment data against all standard pattern information is calculated. Then the relation between the segment data and the standard data with the maximum similarity is checked if there exist contradictions by considering other relations. By repeating this kind of procedure the final result is obtained. Table 1 shows the list of comparative features in the similarity calculation. Fig.5 shows their membership functions.

Table 1 A list of comparative features in the similarity calculation

Attribute	Comparative Feature	
Straight	· Start-End Coordinates	Δp
	· Length	Δl
	· Inclination	$\Delta\theta$
Curve	· Start-End Coordinates	Δp
	· Chord Length	Δl
	· Angle at the Circumference	$\Delta\theta$
Angle	· Angle	$\Delta\theta$

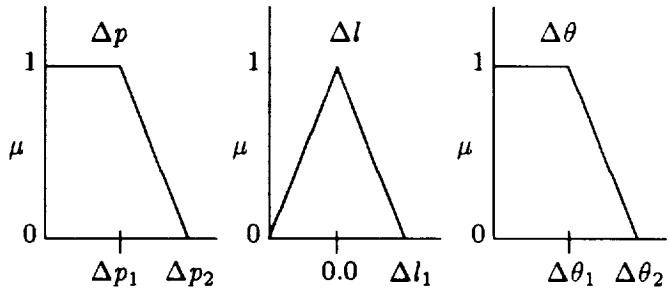


Fig.5 Membership functions of comparative features

8 Experimental result

In order to confirm the availability of this method several experiments have been done, among which one result using bottles is shown. Observed image of bottles consist of straight lines, curved lines with various curvature, and corners. There exist so many similarly looking different bottles. So they are good for testing the presented method.

Fig 6 shows several examples of original image observed by the CCD camera. Experiments were done for the single bottle, a pair of bottles, and three bottles. (The aim of latter two cases is to check the effect for occlusion.) The result is summarized in Table 2, which shows the validity of the proposed method.

9.Conclusion

Fundamental ideas and algorithms of 3D image recognition are proposed based on fuzzy neural network technique. A result of experiment on bottle images is also presented. The construction of real time 3D image recognition system for the purpose of robot vision is a part of future studies.

This study was performed through Special Coordination Funds of the Science and Technology Agency of the Japanese Government.

[References]

[1] K.Hirota, M.Katagai,K.Ikoma: 2D image pattern recognition based on fuzzy neuralnetwork technology, Proc. of 7-th Fuzzy Systems Symposium (SOFT), pp387/390 (1991) (in Japanese

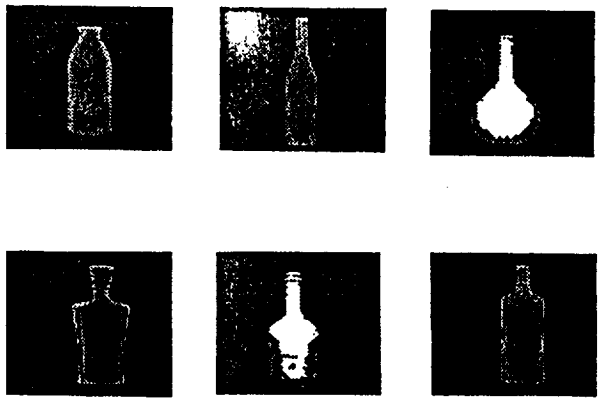


Fig.6 Sample patterns

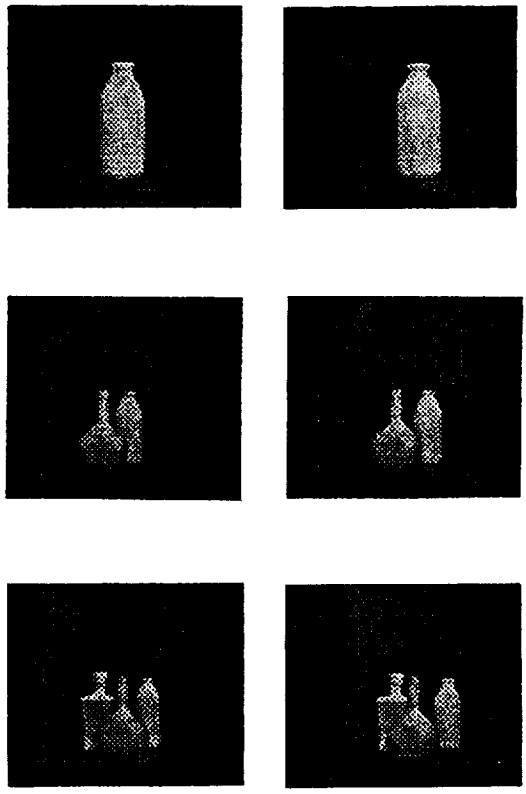
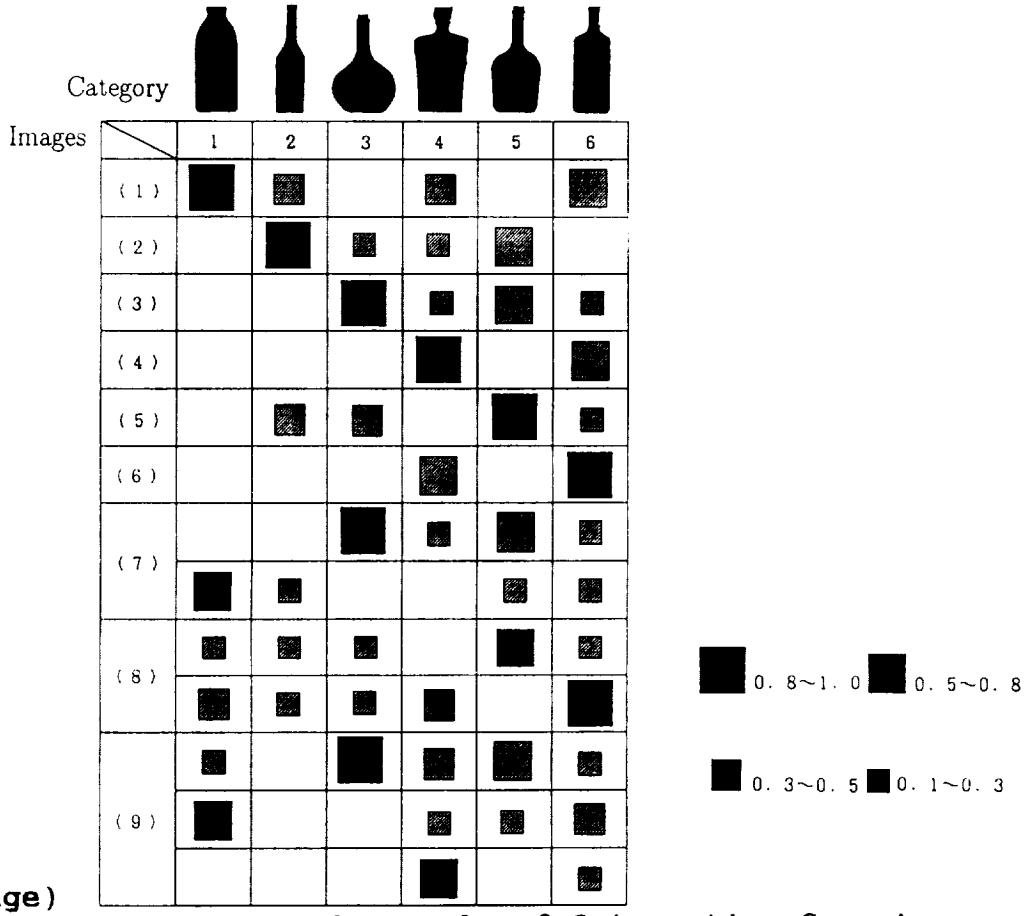


Fig.7 Examples of experimental data

Table 2 Recognition results



language)

[2] J.D.Foley,A.Vandam: Fundamentals of Interactive Computer Graphics, pp.613/620 (1983)

[3] F.Rosenblatt: The Perceptron, A Probabilistic model for information storage and organization in the brain, Physic. Rev. 65(6), pp.386/408 (1958)

[4] D.E.Rumelhalt: Learning representations by back-propagating errors, Nature 329-9, pp.533/536 (1986)

[5] H.Fukunaga,T.Kasai:Partition of line segment images and its application to stereo image processing, Tran.IECE (Japan)(D) J72-D2, pp866/872 (1989) (in Japanese language)

[6] K.Deguchi:Computer Vision, Projective Geometry for Graphics [IV], J.of SICE (Japan),30-3, pp.241/246 (1991) (in Japanese language)

A Proposal of Fuzzy Connective with Learning Function and its Application to Fuzzy Retrieval System

Isao Hayashi, Eiichi Naito, Jun Ozawa and Noboru Wakami
Central Research Laboratories,
Matsushita Electric Industrial Co., Ltd.
3-15, Yakumo-Nakamachi, Moriguchi, Osaka 570, JAPAN
TEL:+81-6-906-4849, FAX:+81-6-904-7252, E-mail:ihaya@g6.crl.mei.co.jp

Abstract

A new fuzzy connective and a structure of network constructed by fuzzy connectives are proposed here to overcome a drawback of conventional fuzzy retrieval systems. This network represents a retrieval query and the fuzzy connectives in networks have a learning function to adjust its parameters by data from a database and outputs of a user. The fuzzy retrieval systems employing this network is also constructed. Wherein users can retrieve results even with a query whose attributes do not exist in a database schema and can get satisfactory results for variety of thinkings by learning function.

1. Introduction

Recently, various fuzzy retrieval system[1)2)] had been developed. In fuzzy retrieval systems, users can retrieve data by using queries with fuzzy propositions[3)] such as a query "Search for a hotel of which rate is low AND is near to the business location" in order to "Search for a hotel which is convenient to the business trip". Fuzzy retrieval sytem is a very convenient mechanism for users since they can write the natural language by fuzzy sets in queries, i.e., "Reasonable", "Long" and "Low" and so on. However, it is nearly impossible to obtain results which satisfy us since the meanings of given operators of AND and OR using for obtaining results in queries are quite different for every user, and the number of usable operators[4)5)] are limited within several, i.e., min operator, algebraic product etc.

On the other hand, in the field of decision making problems, a method to optimize the parameters of fuzzy connectives of AND and OR according to the given input and output data was proposed by Dubois and Prade[6)] ,and Maeda et al[7)]. Fuzzy connective proposed by Maeda is based on γ - operator by Zimmermann[8)]. Parameters of the fuzzy connective are optimized for minimizing the square of errors between the observed data and the estimated value of the fuzzy connective. However, the fuzzy connective can not represent the smaller operators more than the algebraic product or the larger operators more than the algeraic sum since this fuzzy connective is constructed by the geometric mean of between the algebraic product and the algebraic sum.

In this paper, first, a new fuzzy connective[9)10)] capable to express whole operators from the drastic product to the drastic sum is formulated and a new learning method to adjust parameters of fuzzy connective is proposed. The

proposed fuzzy connective is called fuzzy connective with learning function here. The fuzzy connective with learning function is based on Maeda's operator. The t-norm and t-conorm operators[4),5)] with parameters are linearly combined by using a weighting function, and parameters are adjusted for minimizing the square of error by a steepest descent method.

Second, a new structure of network for representing a query is proposed here. Since the new network represents a query, this network is called the query network here. Query networks put the meaning of the abstractive query into shape by attributes of a database. A query network is constructed by nodes and links which join between nodes. Whole nodes except for in the input layer are constructed by the fuzzy connective with learing function. The retrieval system with query networks can give results which users desire since fuzzy connectives in querry networks have the learning function. The similar fuzzy retrieval system is proposed by Ogawa et al[1)]. However, this method can not derive the importance of attributes in a database since the membership functions are adjusted in the learning stage. The retrieval system that we proposed can not only obtain the importance of attributes in a database also acquire the meanings of AND and OR in users' queries from values of parameters of the fuzzy connective.

First, the fuzzy connective with learning function is formulated. Next, the query network is proposed. Finally, the fuzzy retrieval system with this fuzzy connective and the query network is explained here.

2. Conventional Fuzzy Connective

The operators for representing AND and OR are named generically t-norm and t-conorm, respectively. The t-norm T is a function expressing an operator of $T(x_1,x_2)$:$[0,1] \times [0,1] \rightarrow [0,1]$, satisfying the four conditions, i.e., 1)boundary conditions, 2)monotonity, 3)commutativity and 4)associativity. A typical t-norm includes the following operators.

1)Logical product: $x_1 \wedge x_2 = \min\{x_1,x_2\}$ (1)

2)Algebraic product: $x_1 \cdot x_2 = x_1 x_2$ (2)

3)Bounded product: $x_1 \odot x_2 = 0 \vee (x_1+x_2-1)$ (3)

4)Drastic product: $x_1 \triangle x_2 = \begin{cases} x_1 & (x_2=1) \\ x_2 & (x_1=1) \\ 0 & (x_1,x_2<1) \end{cases}$ (4)

The t-conorm S is to express an operation of $S(x_1,x_2) = 1-T(1-x_1,1-x_2)$ and also satisfying four conditions in the case of t-norm. In the same way, t-conorm includes the logical sum, algebraic sum, bounded sum and drastic sum, etc.

On the other hand, the following t-norm and t-conorm operators had been proposed by Schweizer[4)], etc.

$$T = 1-((1-x_1)^p+(1-x_2)^p-(1-x_1)^p(1-x_2)^p)^{1/p} \quad (5)$$

$$S = (x_1^p+x_2^p-x_1^p x_2^p)^{1/p}, \qquad p>0 \quad (6)$$

where, p is a parameter.

By value of parameter p, t-norm of Eq.5 can express logical product, algebraic product, bounded product, drastic product and so on. In the same way, t-conorm can express various operators.

The averaging operators[4] includes arithmetic mean (AM), geometrical mean (GM), conjugated geometrical means (CGM) and so on.

The order of the magnitudes of these operators are expressed by a following relationship.

$$\mathbb{A} \leq \textcircled{\cdot} \leq \cdot \leq \wedge \leq GM \leq AM \leq CGM \leq \vee \leq \dot{+} \leq \textcircled{+} \leq \mathbb{V} \quad (7)$$

Whole operators which inclues t-norm, t-conorm, and averaging operators are called fuzzy connectives here.

3. Fuzzy Connective with Learning Function

In various fuzzy retrieval systems, fuzzy connectives play the important role in queries since the different results of the retrieval system are obtained by kinds of fuzzy connectives. Let us consider a query Q with fuzzy propositions $q_1, q_2, \cdots, q_t$. For instant, a query Q is expresed as follows:

$$Q = (q_1 \sqcap q_2) \sqcup (q_3 \sqcup q_4) \sqcap \cdots \sqcap (q_{t-1} \sqcup q_t) \quad (8)$$

where, $\sqcap$ is intersection and $\sqcup$ is union.

Given the data $x_1, x_2, \cdots, x_t$ for $q_1, q_2, \cdots, q_t$ respectively, the following membership value μ_Q is considered.

- In the case of logical product and logical sum,
$$\mu_Q = (\mu q_1 \wedge \mu q_2) \vee (\mu q_3 \vee \mu q_4) \wedge \cdots \wedge (\mu q_{t-1} \vee \mu q_t). \quad (9)$$
- In the case of algebraic product and algebraic sum,
$$\mu_Q = (\mu q_1 \cdot \mu q_2) \dot{+} (\mu q_3 \dot{+} \mu q_4) \cdot \cdots \cdot (\mu q_{t-1} \dot{+} \mu q_t). \quad (10)$$

In general,

$$\mu_Q = (\mu q_1 \textcircled{T} \mu q_2) \textcircled{\perp} (\mu q_3 \textcircled{\perp} \mu q_4) \textcircled{T} \cdots \textcircled{T} (\mu q_{t-1} \textcircled{\perp} \mu q_t). \quad (11)$$

where, $\textcircled{T}$ shows t-norm and $\textcircled{\perp}$ shows t-conorm.

When we use the conventional retrieval systems, we can not determine the optimum operator to obtain the results we desire since there are so many kinds of fuzzy connectives. Moreover, since there is no operator which is capable of representing from drastic product $\mathbb{A}$ through drastic sum $\mathbb{V}$ in Eq.7, and has the learning function for adjusting parameters of itself to the meanings of AND and OR for every user, it is difficult to employ the fuzzy connective as AND or OR operator.

In ordert to solve this problem, we propose a following new fuzzy connective which can represent a whole operator in Eq.7.

$$\hat{Q} = m\ S + (1-m)\ T \quad (12)$$

where,

$$m = p_1-(p_1-p_2)x_1-(p_1-p_3)x_2$$
$$p_1 \leqq p_2,p_3, \quad 0 \leqq p_1,p_2,p_3 \leqq 1, \quad 0 \leqq -p_1+p_2+p_3 \leqq 1 \tag{13}$$

and p_1, p_2, p_3 are parameters.

T and S in Eq.12 represents t-norm and t-conorm proposed by Schweizer, Yager, and Dombi etc, respectively. For instance, when t-norm and t-conorm proposed by Schweizer are used, T and S are expressed by the following equations using parameters p_4 and p_5.

$$T = 1-((1-x_1)^{p_4}+(1-x_2)^{p_4}-(1-x_1)^{p_4}(1-x_2)^{p_4})^{1/p_4} \tag{14}$$
$$S = (x_1^{p_5}+x_2^{p_5}-x_1^{p_5}x_2^{p_5})^{1/p_5}, \qquad p_4, p_5>0 \tag{15}$$

In the fuzzy connective of Eq.12, t-norm T and t-conorm S are linearly combined by using a value of m which can be derived from the values of x_1 and x_2 by Eq.13. Therefore, the weighted operator between t-norm and t-conorm is derived according to values of x_1 and x_2.

An example of the relationship between input and output of the proposed fuzzy connective is shown in Fig.1 wherein the operator is set to emphasize t-norm when the values of x_1 and x_2 are small while the operator emphasizes t-conorm when the values of x_1 and x_2 are large, and it emphasizes t-conorm further for a larger input value of x_1.

Now, let's explain the learning function of the proposed fuzzy connective. When an output y to the input x_1 and x_2 are given, the proposed fuzzy connective is capable to adjust its parameters by a steepest descent method for minimizing the square E of error between the output y and the output $\hat{y}$ of the fuzzy connective.

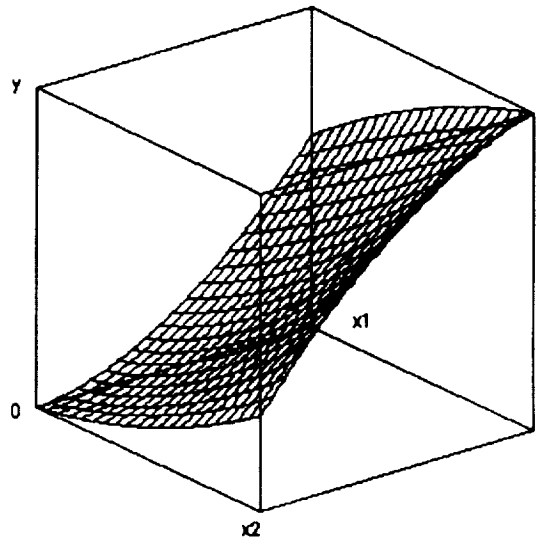


Fig.1 An Example of Relationship Between Input and Output of Fuzzy Connective with Learning Function

$$E = (\hat{y}-y)^2/2 \tag{16}$$

By using a steepest descent, the amounts of corrections of parameters p_j, j=1,2, ···,5 in Eq.12 to 15 are revised by the following equation.

$$\begin{aligned} p_j^{t+1} &= p_j^t + \Delta p_j \\ &= p_j^t - \alpha \left(\partial E / \partial p_j \right) \end{aligned} \tag{17}$$

where, p_j^t is the t-th revised parameter p_j, and α is a learning coefficient.

$\partial E / \partial p_j$ which is an effect of minute change of parameter p_j to the error E, can be expressed by the following equation.

$$\frac{\partial E}{\partial p_j} = \frac{\partial E}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial p_j} = (\hat{y}-y) \times \frac{\partial \hat{y}}{\partial p_j} \tag{18}$$

$\partial \hat{y} / \partial p_j$ can be derived from Eqs.12 to 15 by the following equation.

$$\frac{\partial \hat{y}}{\partial p_1} = (1-x_1-x_2) \times (S-T) \tag{19}$$

$$\frac{\partial \hat{y}}{\partial p_2} = x_1 \times (S-T) \tag{20}$$

$$\frac{\partial \hat{y}}{\partial p_3} = x_2 \times (S-T) \tag{21}$$

$$\frac{\partial \hat{y}}{\partial p_4} = (1-m) \times \frac{\partial T}{\partial p_4} \tag{22}$$

$$\frac{\partial \hat{y}}{\partial p_5} = m \times \frac{\partial S}{\partial p_5} \tag{23}$$

When t-norm T and t-conorm S are defined by Schweizer's ones, Eq.22 and Eq.23 are revised as the following equations.

$$\begin{aligned} \frac{\partial \hat{y}}{\partial p_4} = (1-m)(1-T)\Big(& \frac{1}{p_4^2} \log((1-x_1)^{p_4}+(1-x_2)^{p_4}-(1-x_1)^{p_4}(1-x_2)^{p_4}) \\ & - \frac{1}{p_4((1-x_1)^{p_4}+(1-x_2)^{p_4}-(1-x_1)^{p_4}(1-x_2)^{p_4})} \left((1-x_1)^{p_4}\log(1-x_1)\right. \\ & \left. +(1-x_2)^{p_4}\log(1-x_2)-(1-x_1)^{p_4}(1-x_2)^{p_4}\log(1-x_1)(1-x_2)\right)\Big) \end{aligned} \tag{24}$$

$$\begin{aligned} \frac{\partial \hat{y}}{\partial p_5} = mS\Big(& -\frac{1}{p_5^2} \log(x_1^{p_5}+x_2^{p_5}-x_1^{p_5}x_2^{p_5}) \\ & + \frac{1}{p_5(x_1^{p_5}+x_2^{p_5}-x_1^{p_5}x_2^{p_5})} \left(x_1^{p_5}\log(x_1)+x_2^{p_5}\log(x_2)\right. \\ & \left. -x_1^{p_5}x_2^{p_5}\log(x_1x_2)\right)\Big) \end{aligned} \tag{25}$$

Employing a steepest descent method, the value of E is minimized by repeating Eq.17. Since the proposed fuzzy connective is capable of learning parameters, this fuzzy connective is called the fuzzy connective with learning function here.

Next, let's consider the conditions for constituting AND and OR operators of queries. The commutativity and associativity within four conditions for t-norm and t-conorm are not always satisfied since there are so many kinds of operators constructing AND and OR. Moreover, it is not need that the boundary conditions are satisfied in this case since there are cases that the averaging operators are considered in the queries. However, since no reliability of results would be gained unless a monotonity between the given input data and retrieved output can be established, the satisfaction of monotonity is a must in this case.

Since there are many kinds of fuzzy connectives with learning function in the query network, for instance, the query Q is represented as follows:

$$Q = (q_1 \otimes_1 q_2) \otimes_2 (q_3 \otimes_3 q_4) \otimes_4 \cdots \otimes_{t-2} (q_{t-1} \otimes_{t-1} q_t) \tag{26}$$

where, $\otimes_k$, k=1,2, ···,t-1 shows the k-th of fuzzy connectives with learning function in the query network.

Since there are cases that we treat fuzzy connectives with n inputs in the queries, let us extend the fuzzy connective with learning function to one which is capable of representing n inputs $x_1, x_2, \cdots, x_n$ as follows.

$$\hat{y} = m \cdot S + (1-m) \cdot T \tag{27}$$

where,

$$m = p_1 - \sum_{j=1}^{n} (p_1 - p_{j+1}) x_j, \qquad 0 \leqq p_1, p_2, \cdots, p_{n+1} \leqq 1, \quad 0 \leqq -(n-1)p_1 + \sum_{j=1}^{n} p_j \leqq 1 \tag{28}$$

When t-norm T and t-conorm S are defined by Schweizer's ones,

$$T = 1 - (1 - \prod_{j=1}^{n} (1 - (1-x_j)^{p_{n+2}}))^{1/p_{n+2}} \tag{29}$$

$$S = (1 - \prod_{j=1}^{n} (1 - x_j^{p_{n+3}}))^{1/p_{n+3}}, \qquad p_{n+2},\ p_{n+3} > 0 \tag{30}$$

where $p_1, p_2, \cdots, p_{n+3}$ are parameters.

Next, let's explain the learning method of the fuzzy connective with learning function as same as in the case of two input variables. When the output y to the input $x_1, x_2, \cdots, x_n$ are given, the amounts of corrections of parameters p_j are revised as same as in Eq.17.

$$\begin{aligned} p_j^{t+1} &= p_j^t + \Delta p_j \\ &= p_j^t - \alpha \left(\partial E / \partial p_j \right), \qquad j=1,2, \cdots, n+3 \end{aligned} \tag{31}$$

$\partial E / \partial p_j$ which is an effect of minute change of parameter p_j to the error E, can be expressed by the following equation.

$$\frac{\partial E}{\partial p_j} = \frac{\partial E}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial p_j} = (\hat{y} - y) \times \frac{\partial \hat{y}}{\partial p_j} \tag{32}$$

$$\frac{\partial \hat{y}}{\partial p_1} = (1 - \sum_{i=1}^{n} x_i) \times (S-T) \tag{33}$$

$$\frac{\partial \hat{y}}{\partial p_j} = x_{j-1} \times (S-T) \tag{34}$$

$$\frac{\partial \hat{y}}{\partial p_{n+2}} = (1-m) \times \frac{\partial T}{\partial p_{n+2}} \tag{35}$$

$$\frac{\partial \hat{y}}{\partial p_{n+3}} = m \times \frac{\partial S}{\partial p_{n+3}} \tag{36}$$

Employing a steepest descent method, the value of E is minimized by repeating Eq.31.

A new structure of network for representing a query is proposed here. Since the new network represents a query, this network is called the query network here.

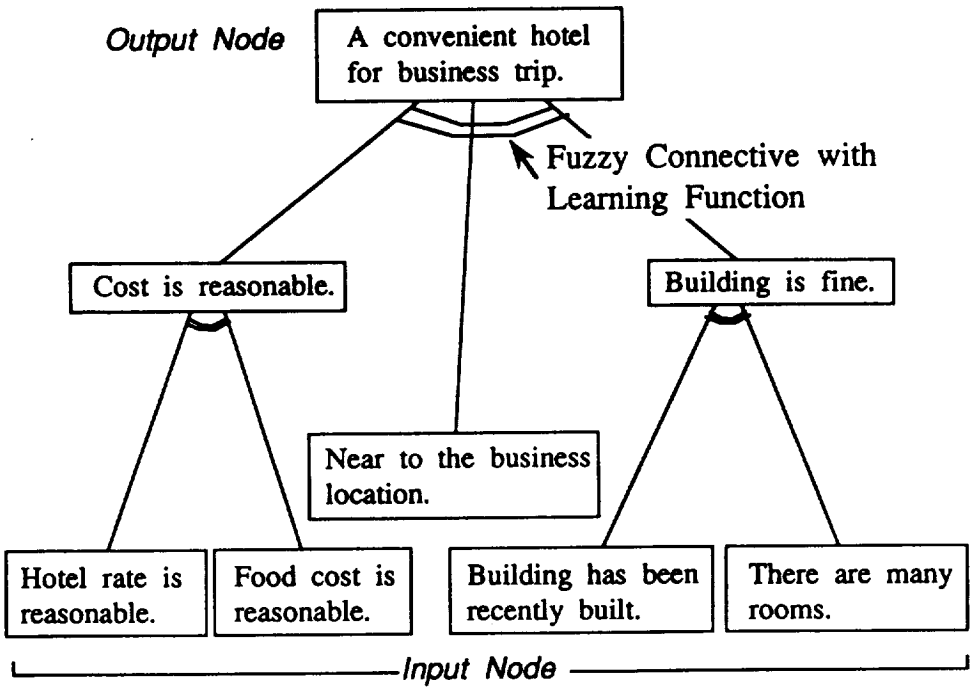


Fig.2 A Example of the Query Network

Let us define the query network as follows :

1) A query network is constructed by nodes N_m, m=1,2, ···,M which are joints of network and links L_l, l=1,2, ···,M-1 which join a node to other nodes. Nodes in each layer except for in the input and output layer have to join itself to a node in the upper layers and some nodes in the lower layers.
2) There are no links which joint between nodes in the same layer.
3) Every node is constructed by the fuzzy connective with learning function.
4) Every node means a fuzzy proposition.

where, the node in the most upper which is the output layer is called an output-node and nodes in the most lower layer which is the input layer are called input-nodes.

4. Proposed Query Networks

A example of a query network is shown in Fig.2. Now, let us assume that n a five kinds of attributes for searching hotels, i.e., hotel rate, food cost, access time, yaers and rooms are stored in a database. This query network puts the meaning of the output-node which is "Search for a hotel for business trip" into shape by five kinds of attributes through three kinds of nodes which are "Cost is reasonable", "Near to the business location" and "Building is fine" in the middle layer. By using the query network, it is easy to find some hotel by the meanings which is "Search for a hotel for business trip".

Next, let us explain how to learn parameters of fuzzy connective with learning function in query networks when the input x and output y are given. Now, let us represent the output of the i-th fuzzy connective with learning function ordered from output-node as y_i with parameters p_{ij}, j=1,2, ···,u. The learning algorithm is based on a backpropagation method for minimizing the square E of error between the output y and the output y_1 of output-node in the query network.

$$E = (y_1-y)^2/2 \tag{37}$$

In order to obtain the optimum parameters of the i-th fuzzy connective with learning function for minimizing E, an effect of minute change of parameter to the error E is calculated by the following equation.

$$\frac{\partial E}{\partial p^i_j} = \frac{\partial E}{\partial y_1} \times \frac{\partial y_1}{\partial p^i_j}, \quad i=1,2,\cdots,W \tag{38}$$

$\partial E/\partial y_1$ can be derived from Eq.37 by the following equation.

$$\frac{\partial E}{\partial y_1} = y_1-y \tag{39}$$

$\partial y_1/\partial p^i_j$ can be obtained as follows.

$$\frac{\partial y_1}{\partial p^i_j} = \delta_i \times \frac{\partial y_i}{\partial p^i_j} \tag{40}$$

where, δ_i is

$$\delta_i = \delta_{i-1} \times \frac{\partial y_{i-1}}{\partial p^{i-1}_{\kappa}}, \qquad i \geqq 2 \tag{41}$$

y_{i-1} is the output of the (i-1) th fuzzy connective with learning function whose input is equal to the output of i-th fuzzy connective with learning function.

We can calculate Eq.40 in the case that the i-th fuzzy connective with learning function is not the output-node. The learning method in the output-node has been explained in the third chapter.

Since δ_i is obtained by repeating Eq.41 in the upper layer more than the i-th fuzzy connective with learning function, $\partial E / \partial p^i_j$ in Eq.38 can be calculated. Therefore, the amounts of corrections of parameters p^i_j in Eq.38 to 41 are revised by the following equation.

$$\begin{aligned} p^i_j{}^{t+1} &= p^i_j{}^t + \Delta p^i_j \\ &= p^i_j{}^t - \beta \left(\partial E / \partial p^i_j \right) \end{aligned} \tag{42}$$

where, $p^i_j{}^t$ is the t-th revised parameter p^i_j, and β is a learning coefficient. The value of E is minimized by repeating Eq.42.

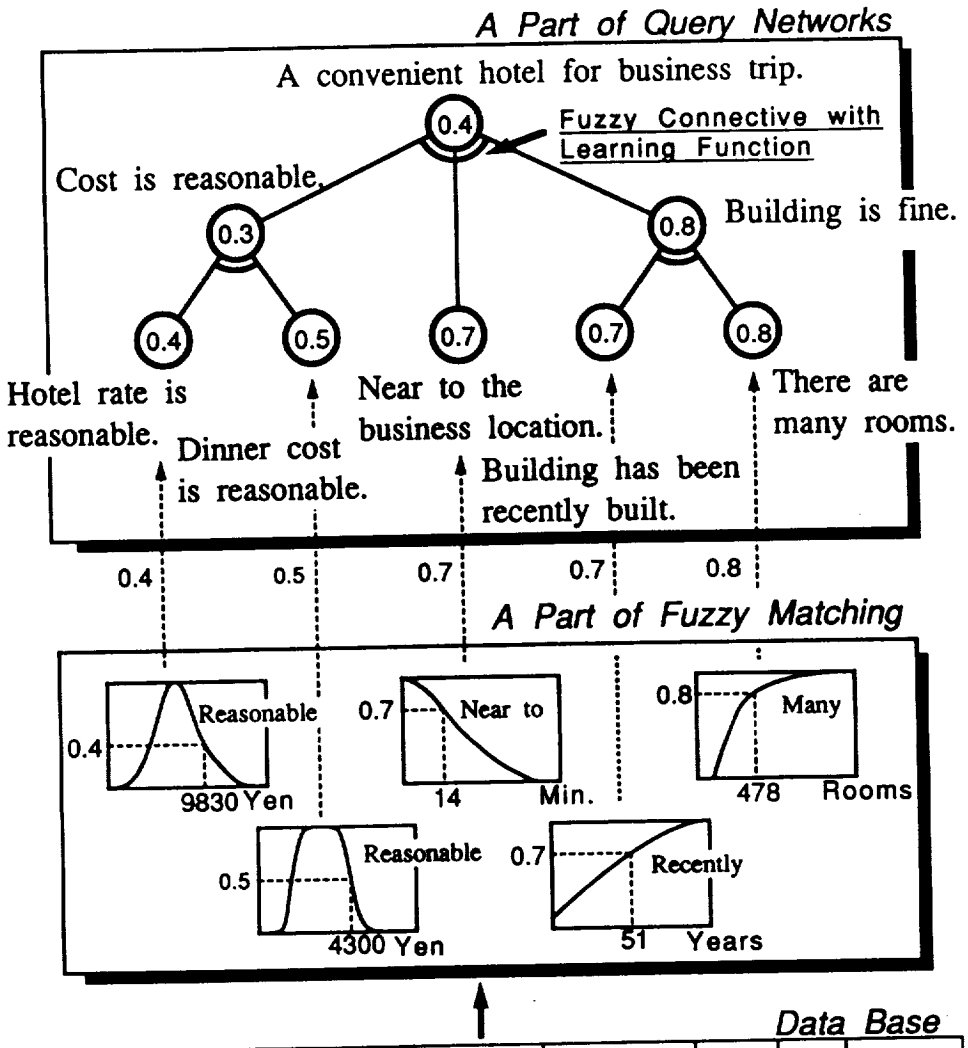
5. Fuzzy Retrieval System

In order to show the usefulness of the fuzzy connective with learning function and the query network, these mechanism are applied to the fuzzy retrieval system.

A conceptual drawing of developed retrieval system is shown in Fig.3. Data in a database are converted into membership values by using membership functions in the fuzzy matching part. These membership values are input to input-nodes of the query network. The results of the retrival system from the output-node after adjusted fuzzy connectives are obtained.

Now, let us consider here a user who search for a convenient hotel for business trip from a database stored 100 hotels near Osaka shown in Table 1. In the proposed fuzzy retrieval system, the following query network shown in Fig.2 is already constructed.

Search for a convenient hotel for business trip.
= Search for a hotel of which cost is reasonable
and(or) is near to the business location
and(or) whose building is fine.
= Search for a hotel of which rate is reasonable
and(or) of which food cost is reasonable
and(or) is near to the business location
and(or) whose building has been recently built
and(or) has so many rooms.



No.	Hotel Name	Hotel rate	Dinner Cost	Time	Year	Rooms
2	Osaka Hilton International	17000	3700	14	61	514
1	Osaka Dai-ichi Hotel	9830	4300	18	51	478
:	:	:	:	:	:	:
100	Hotel Sun Garden	5700	1500	58	45	120

Fig.3 Conceptional Drawing of Developed Fuzzy Retrieval System

The steps for retrieving are represented as follows.

1)The system displays 10 hotels as sample data which represent some kinds of sets constructed by five attribute. A user gives estimations of sample data in [0,100] according to the query which is "Search for a convenient hotel for business trip" to the system.

2)Parameters of whole fuzzy connectives with learning function in the query network are adjusted by learning algorithms in the third and forth chapter.

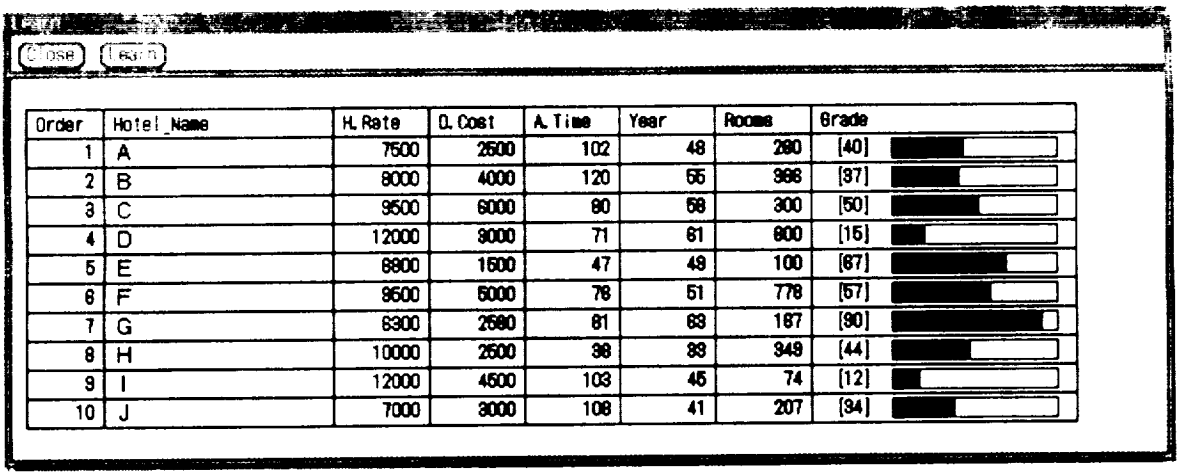
Table 1 A Database of Hotel near Osaka

No.	Hotel Name	Hotel Rate	Dinner Cost	Access Time	Year	Rooms
1	Osaka Hilton International	17000	3700	14	61	514
2	Osaka Dai-ichi Hotel	9830	4300	18	51	478
3	Hotel Hanshin	7800	3000	34	57	209
4	Osaka Terminal Hotel	8500	3800	38	58	664
5	Osaka ANA Hotel Sheraton	12500	5000	54	59	500
6	Dojima Hotel	10000	5000	26	59	134
7	Osaka Grand Hotel	9300	1500	30	33	349
8	Royal Hotel	12500	10000	6	40	1246
9	Hotel NCB	5500	1000	42	50	174
10	Umeda OS Hotel	6500	3000	48	49	283
11	Osaka Tokyu Inn	7800	1800	20	53	402
12	Hotel Kitahachi	5500	1000	56	21	38
13	Maruichi Hotel	4800	1000	12	44	44
14	Hokke Club Osaka	6100	2000	25	41	307
15	Hotel Kansai	4800	1000	37	45	711
16	Hotel Osaka World	5500	1000	48	57	202
17	Osaka ShampiaChampagne Hotel	6100	2000	40	51	300
18	Hotel Kurebe Umeda	5500	3000	14	60	282
19	East Hotel	5200	2700	20	58	144
20	Toko Hotel	5900	2500	58	54	300
21	Hotel Plaza Osaka	5500	2000	47	56	113
22	Osaka Tokyu Hotel	9000	4500	38	54	340
23	Shin-Hankyu Hotel	7800	3000	31	39	993
24	Kishu Railway Hotel	5500	1500	15	55	66
25	Hotel Sunroute Umeda	6000	1500	42	58	218
26	Mitsui Aurbum Hotel Osaka	6500	3500	55	53	405
27	Toyo Hotel	8800	3500	60	40	528
:	:	:	:	:	:	:
100	Hotel Sun Garden	5700	1500	58	45	120

3)The membership values calculated in the fuzzy matching part are input into the input layer of the query network. After the fuzzy connectives with learning function are fixed in the learning stage, the system can retrieve some hotels which users desire.

Fig.4 shows a input display for the 10 sample hotel data estimated by the user. In Fig.4, the degrees of convenience to the business trip that the user provided for the learning are shown.

Fig.5 shows the results after the learning stage. In order to shows the robustness of this learing algorithm, the result of errors between the checking data which a user estimated except for the learning data and the output of the system is also shown. Since the errors between the user's data and the output are small not only for the learning data but also for the checking data, we can obtain the optimum results by this retrieval system.



Order	Hotel_Name	H. Rate	D. Cost	A. Time	Year	Rooms	Grade
1	A	7500	2500	102	48	280	[40]
2	B	8000	4000	120	55	366	[37]
3	C	9500	6000	80	58	300	[50]
4	D	12000	9000	71	61	800	[15]
5	E	6800	1600	47	49	100	[67]
6	F	9600	5000	76	51	778	[57]
7	G	6300	2580	61	63	167	[90]
8	H	10000	2500	98	88	349	[44]
9	I	12000	4500	103	45	74	[12]
10	J	7000	3000	108	41	207	[34]

Fig.4 Input Display and Degrees of Hotel List Proved User for the Learning

Fig.6 shows the results of weights of links in the query network. Since both links between the output-node and the middle node which represents "cost is reasonable" and links between this middle node and the input-node which represents "hotel rate is reasonable" are written by bold lines, it means that the user considers the hotel rate is more important than the access convenience of hotel and so on. Fig.7 shows the results of hotels near Osaka. Fig.8 shows a photograph of the eighth hotel. Fig.9 shows the other results of hotel near Yokohama which are retrieved from the different database by the adjusted fuzzy connective with learning function. From these results shown in Fig.7 and Fig.9, users can determine the hotel that they want to stay at.

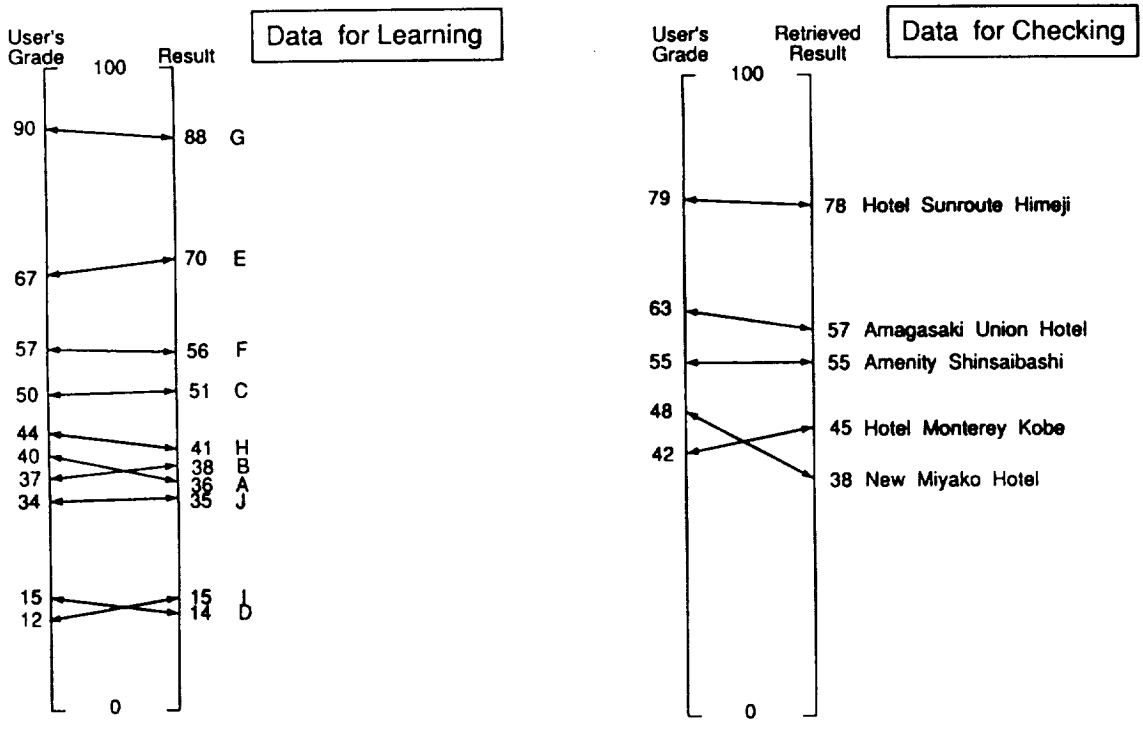


Fig.5 Results of Training Data and Checking Data

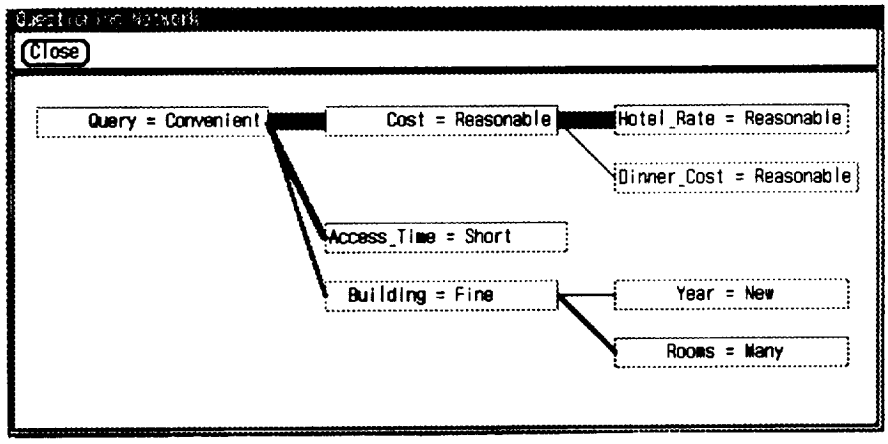


Fig.6 Results of Weights of Links in the Query Network

Close

Order	Hotel_Name	H.Rate	D.Cost	A.Time	Year	Rooms	Grade
1	Hotel_Sunroute_Himeji	6700	1800	102	52	89	[78]
2	Sanjo_Karasuma_Hotel_Kyoto	6800	2500	90	59	154	[74]
3	Kobe_Union_Hotel	6300	2580	81	63	167	[73]
4	Shin-Osaka_Sunplaza_Hotel	6800	1500	47	49	100	[70]
5	Amagasaki_Union_Hotel	6000	2000	53	47	186	[57]
6	Asahi_Plaza_Hotel_Shinsaibashi	6200	1000	45	45	88	[56]
7	Umeda_OS_hotel	7000	3000	40	58	283	[56]
8	Amenity_Shinsaibashi	6100	3000	45	61	127	[56]
9	Rihga_Royal_Hotel_Yotsubashi	7500	1500	43	60	143	[56]
10	Himeji_Castle_Hotel	7000	3000	108	41	207	[55]
11	Hotel_Sungarden_Himeji	7500	2500	102	48	260	[51]
12	Mitui_Urban_Hotel_Wakayama	5800	1500	105	48	110	[49]
13	Hotel_Monterey_Kobe	7500	3000	66	52	164	[45]
14	Himeji_Green_Hotel	5700	2700	110	62	105	[41]
15	New_Miyako_Hotel	8000	6000	70	63	714	[38]
16	Hotel_Keihan_Kyoto	6900	6000	70	49	308	[36]
17	Wakayama_Tokyu_Inn	7500	3500	109	41	165	[36]
18	Himeji_Washington_Hotel	6500	4000	106	47	145	[35]
19	Kyoto_Tower_Hotel	6500	4000	70	35	145	[35]
20	Osaka_Terminal_Hotel	12000	4500	35	58	665	[35]

Fig.7 Results of Hotel Near Osaka

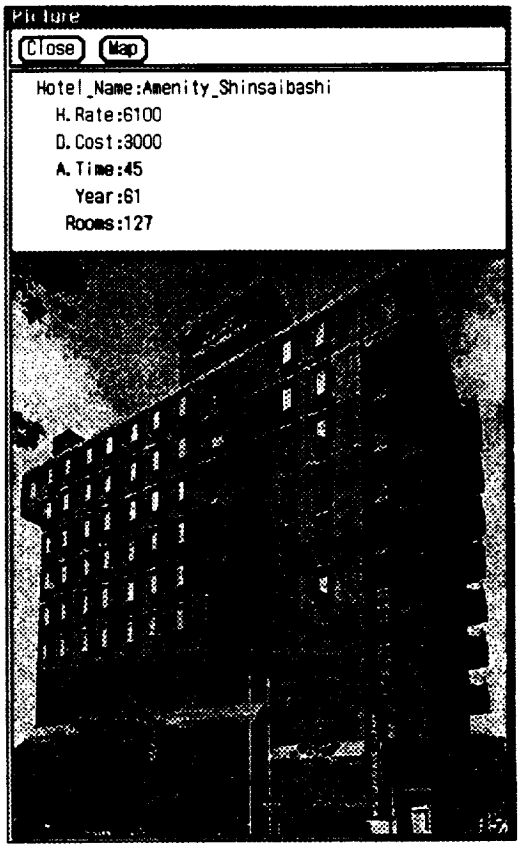


Fig.8 A Potograph of the Eighth Hotel in Results

Results

Close

Order	Hotel_Name	H.Rate	D.Cost	A.Time	Year	Rooms	Grade
1	Pendulum_Inn_Yokohama	6500	1800	18	52	89	[68]
2	Yokohama_Plala_Hotel	6600	3000	37	50	36	[64]
3	Tsurumu_Park_hotel	5800	2200	25	57	315	[53]
4	Heiiwa_Plaza_Hotel	8000	3000	15	61	127	[48]
5	Yokohama_San-Kai_Hotel	6000	5000	15	52	98	[46]
6	Meihin_Hotel	6300	5000	39	61	574	[44]
7	Ilsezakicho_Washington_Hotel	9350	6000	19	60	70	[42]
8	Hotel_Umpire	9100	4000	3	55	366	[41]
9	Hotel_Dream_Round	6500	4000	44	53	140	[40]
10	Hotel_Ritchie_Yokohama	9900	1500	16	63	70	[39]
11	Shin-Yokohama_Kokunai_Hotel	8000	4000	37	44	33	[36]
12	Tokyo_Stationery_Hotel	8200	5000	57	52	489	[36]
13	Tanaka-Ya	9000	2500	35	48	260	[36]
14	Suimai	7000	4000	50	51	120	[35]
15	Taimaru_Hotel	4000	4000	25	56	54	[35]
16	Hotel_New_Ground	10600	4000	28	63	145	[35]
17	Ground_Inter-Continental_Hotel	18500	5000	6	62	115	[34]
18	Kakanawa_Tobu_Hotel	9400	2200	43	62	208	[33]
19	Yokohama_Kokusai_Hotel	8700	3000	48	58	100	[32]
20	Weekday_Inn_Tokyo	14700	3600	58	47	653	[21]

Fig.9 Results of Hotel Near Yokohama

5. Conclusion

A fuzzy connective with learning function used a steepest descent method and a query network used a backpropagation method are proposed here. Moreover, a fuzzy retrieval system used by these mechanism is described. In near future, its practical effectiveness has to be proved through more practical applications of this system.

This research is partly performed through Special Coordination Funds of the Science and Technology Agency of the Japanese government.

References

1)S.Miyamoto : Fuzzy Sets in Information Retrieval and Cluster Analysis, Theory and Decision Library, Series D, Kluwer Academic Publishers (1990)

2)M.Zemankova-Leech and A.Kandel : Fuzzy Relational Data Bases - A Key to Expert System, TUV Rheinland (1984)

3)L.A.Zadeh : PRUF - A Meaning Representation Language for Natural Language, Int.J.Man-Machine Studies, Vol.10, 395/460 (1978)

4)M.Mizumoto : Pictorial Representation of Fuzzy Connectives, Part I. Cases of t-norms, t-conorms and Averaging Operators, Vol.31, No.2, 217/242 (1989)

5)M.Mizumoto : Pictorial Representation of Fuzzy Connectives, Part II. Cases of Compensatory Operators and Self-Dual Operators, Vol.32, No.1, 45/79 (1989)

6)D.Dubois and H.Prade : Possibility Theory, An Approach to Computerized Processing of Uncertainty, Plenum Press, New York (1988)

7)H.Maeda and S.Murakami : A Fuzzy Decision-Making Method and Its Application to a Company Choice Problem, Information Sciences, Vol.45, 331/346 (1988)

8)H.J.Zimmermann and P.Zysno : Latent Connectives in Humann-Decision Making, Fuzzy Sets and Systems, Vol.4, 37/51 (1980)

9)I.Hayashi, E.Naito and N.Wakami : A Proposal of Fuzzy Connective with Learning Function and Its Application to Fuzzy Information Retrieval, IFES'91, Yokohama, 446/455 (1991)

10)I.Hayashi, E.Naito, J.Ozawa and N.Wakami : A Proposal of Fuzzy Connective with Learning Function and its Application to Fuzzy Retrieval Systems, Third International Workshop on Neural Networks and Fuzzy Logic'92, Houston, (1992)

11)Y.Ogawa, T.Morita and K.Kobayashi : A Fuzzy Document Retrieval System using the Keyword Connection Matrix and a Learning Method, Fuzzy Sets and Systems, Vol.39, No.2, 163/179 (1991)

APPLICATION OF FUZZY SET AND DEMPSTER-SHAFER THEORY TO ORGANIC GEOCHEMISTRY INTERPRETATION

C. S KIM and G. H. Isaksen
Exxon Production Research Co.
Houston, Tx. 77001
(713) 965-7645

ABSTRACT

This paper presents an application of fuzzy sets and Dempster Shafer theory (DST) in modeling the interpretational process of organic geochemistry data for predicting the level of maturities of oil and source rock samples. This has been accomplished by (i) representing linguistic imprecision and imprecision associated with experience by a fuzzy set theory, (ii) capturing the probabilistic nature of imperfect evidences by a DST, and (iii) combining multiple evidences by utilizing John Yen's[1] generalized Dempster-Shafer Theory(GDST), which allows DST to deal with fuzzy information. The current prototype provides collective beliefs on the predicted levels of maturity by combining multiple evidences through GDST's rule of combination.

I. INTRODUCTION

Modeling the interpretation process of an expert requires representation and management of uncertain knowledge. This is because nearly every interesting domain contains knowledge that is inherently inexact, incomplete, or unmeasurable.

In this paper we explicitly treat two forms of uncertainties. One form of uncertainty is fuzziness related to linguistic imprecision. Based on fuzzy set theory, Zadeh[2] developed possibility theory to express this type of imprecision. The other form of uncertainty is the probability with which a certain evidence correctly predicts a subset of hypotheses. Dempster-Shafer Theory[3,4] (DST) deals with this type of uncertainty and provides a mechanism for combining multiple evidences for an overall belief in a subset of hypotheses. Unlike classical probability theory, DST enables the degree of

ignorance to be expressed explicitly and does not fix hypothesis negation probability once occurrence probability is known.

In the past, several attempts[5,6] have been made to generalize DST to deal with fuzzy information. While these attempts fall short of fully justifying their approaches, John Yen[1] proposed a generalized Dempster-Shafer Theory (GDST), in which the important principle of DST is preserved: That the belief and the plausibility functions are treated as lower and upper probability bounds.

In this paper, we demonstrate representation and management of two types of uncertainties by GDST as applied to the interpretation of organic geochemistry data. In the following sections, we review the basics of GDST, and the development of a knowledge-based system for geochemistry interpretation

II. BASICS OF A GENERALIZED DEMPSTER-SHAFER THEORY

This review is not intended to describe detailed theory and developments of DST and GDST. Rather, we plan to describe their representation of imprecise information and the rule of combination in a qualitative way. More interested readers should refer to the references [1,3,4] cited.

In the DST, hypotheses in a frame of discernment must be mutually exclusive and exhaustive, meaning that they must cover all the possibilities and the individual hypothesis cannot overlap with others. An important advantage of DST over classical probability theory is its ability to express degree of ignorance associated with an evidence. Also, unlike classical probability theory, a commitment of belief to a hypothesis does not force the remaining belief to be assigned to its compliment. Therefore, the amount of belief not committed to any of the subsets of hypotheses represents the degree of ignorance. In DST, a basic probability assignment(bpa) m(A), as a generalization of a probability, indicates belief in a subset of hypotheses A. This quantity m(A) serves as a measure of belief committed to the subset A.

DST also provides a formal process for combining bpa's induced by independent evidential sources, which is called the rule of combination. This process is a tool for accumulating evidences to

narrow the hypothesis set. If m_1, and m_2 are two bpa's from two evidential sources, a combined bpa is computed according to the rule of combination:

$$m_1 \oplus m_2(C) = \sum_{A_i \cap B_j = C} m_1(A_i) m_2(B_j)/k \qquad (1)$$

where k is a normalization factor,

$$k = 1 - \sum_{A_i \cap B_j = \phi} m_1(A_i) m_2(B_j), \qquad (1a)$$

$m_1 \oplus m_2(C)$ is a combined bpa for a hypothesis C,
ϕ is a null set, and
A_i, B_j are hypotheses sets induced by the two evidential sources.

In the GDST proposed by Yen[6], a basic probability m(A) is assigned to a fuzzy subset of hypotheses. In this framework, each fuzzy subset of hypotheses has bpa m(A), and fuzzy membership function $\mu_A(x_i)$, where x_i's are elemental hypotheses in the frame of discernment. The rule of combination in GDST consists of two operations: a cross-product operation and a normalization process. Basic probabilities are first combined by performing a generalized cross-product including fuzzy set operations:

$$m'_{12}(C) = m_1 \otimes m_2(C) = \sum_{A_i \cap B_j = C} m_1(A_i)\ m_2(B_j) \qquad (2)$$

where $m'_{12}(C)$ is an unnormalized bpa induced by two evidences, and $\cap$ denotes a fuzzy intersection operator.

Then, a normalization is performed on fuzzy subsets of hypotheses whose maximum membership values are less than one. A detailed procedure and justification of this normalization process can be found in the reference [1]. Yen[1] also showed that this normalization can be postponed until the last evidence without affecting the computational results and the commutativity of the rule of combination.

In case of combining only two fuzzy bpa's, a combined bpa using GDST's rules of combination is:

$$m_1 \oplus m_2(C) = \sum_{(\overline{A \cap B}) \in C} \underset{x_i}{Max}\, \mu_{A \cap B}(x_i)\, m_1(A) m_2(B)/k \qquad (3)$$

where

$$k = 1 - \sum_{A,B} (1 - \underset{x_i}{Max}\, \mu_{A \cap B}(x_i))\, m_1(A) m_2(B), \text{ and} \qquad (3a)$$

$\overline{A \cap B}$ is a normalized $A \cap B$.

As can be noticed in the equations above, GDST allows partially conflicting evidences, while DST only allows either conflicting or confirming evidences.

III. BIOMARKER INTERPRETATION SYSTEM

In exploration for oil and gas, it is important to be able to assess the maximum temperatures to which sediments or oils have been exposed in the subsurface. This is referred to as the level of thermal maturity. Organic chemical compounds known as biomarkers enable the geochemist to assess the level of maturity (LOM) of oils and sedimentary organic matter. In this paper, we focus our attention on modeling the process of interpreting biomarker data to predict LOM. The LOM scale ranges from 1 to 20, with LOM=1 being least mature and LOM=20 most mature. There exist more than 10 biomarkers whose intensities have definite links to the maturity with varying degrees of resolution and prediction power.

In our approach, these varying degrees of resolution among biomarker evidences are represented by fuzzy subsets of maturity intervals, and the probability with which an evidence correctly predicts a fuzzy maturity interval is represented by a basic probability in GDST. Therefore, evidential knowledge is represented in fuzzy rules, and the confidence for a specific rule is represented by a bpa. Moreover, GDST's rule of combination provide collective belief in the predicted level of maturity. In the following, detailed representation methods are presented along with actual application results.

(A) Representing Two Types of Imprecision

Interpretation of geochemical data is based on experience as well as theory. This interpretational knowledge is descriptive in nature, and

best represented by fuzzy logic and possibility theory. For example, one may have an experience based correlation study between level of maturity (LOM) and $\%C_{29}20S$, which is a ratio of the intensities of several organic compounds. Then, the correlation curve in Figure 1 may be used by an interpreter as follow:

IF $\%C_{29}20S$ is 40 %,
THEN expected LOM is **about** 8.

In the rule above, the concluding part is descriptive in that LOM = 8 is most possible, but LOM values of 6,7,9, and 10 are also possible with lesser degree as shown in Figure 2. Another example is the case where both premise and conclusion are best represented by fuzzy membership functions. Based on theory and experience, Heptane value can only predict maturity levels in four qualitative categories, such as immature, early mature, mature, and over mature. Examples of Heptane rules are:

IF Heptane value is medium,
THEN maturity is early mature

IF Heptane value is high,
THEN maturity is mature

IF Heptane value is very high,
THEN maturity is over mature

In the rules above, both the premise and the conclusions are descriptive and best represented by membership functions for Heptane value and maturity as depicted in Figure 3a and Figure 3b. From the fuzzy rules above and the membership functions in Figures 3a and 3b, observation of a Heptane value of 19 will result in the possibility values of 0.5, 1.0, 1.0, and 0.5 for LOM = 6, 7, 8, and 9 respectively:

$$\Pi_{LOM} = \{0.5/6,\ 1/7,\ 1/8,\ .5/9\} \qquad (4)$$

In the current system, LOM is predicted from 10 evidences each of which predicts LOM with different degree of resolution as shown by the two examples above.

In addition to the imprecision in the knowledge represented by possibility theory above, there exists another type of uncertainty associated with evidences. For example, rules associated with $\%C_{29}20S$ have higher probability of being true than the Heptane

rules. In our approach, the probability with which a proposition " If A is a1 Then B is b1" is true is represented by bpa assigned to the fuzzy subset of hypotheses induced by the proposition. The compliment of this probability is assigned to the degree of ignorance associated with the proposition, since our system generates only one fuzzy subset of hypotheses for each evidence.

(B) Test Result

In order to validate the system, thirty interpretations were tested to see if the system's interpretations conformed to those of the expert. With reference to the test results listed in Table 1, one can notice that the system interpreted maturities are biased towards higher LOM. However, these errors are all higher than they should be and consistent by itself, and can be traced to the membership function definitions. We are currently fine tuning these membership functions to correct the problem and plan to test the system with additional field data..

V. CONCLUSIONS

We presented a knowledge-based system in which linguistic imprecisions and uncertainties associated with fuzzy rules are modeled in the frame work of a generalized Dempster-Shafer Theory. This development is significant in that many application problems in oil exploration requires a mechanism of combining fuzzy information from various sources.

Even though the current biomarker interpretation system has been tested on only 30 data sets, the system will be further tested with additional field data and expanded to handle interpretations for other characteristics such as source facies, depositional environments, and the degree of biodegradation.

REFERENCES

1. Yen, J., Generalizing the Dempster-Shafer theory to fuzzy sets, IEEE Trans., Sys., Man, & Cyb., Vol. 20, No, 3, May/June 1990
2. Zadeh, L.A., Fuzzy sets as a basis for a theory of possibility, Fuzzy Sets & Systems 1(1978) 3-28, North-Holland Publishing Co., 1978
3. Dempster, A.P., Upper and lower probabilities induced by a multivalued mapping, Annals Math. Statistics, Vol. 38, No. 2, 1967, pp.325-339
4 Shafer, G., A mathematical theory of evidence, Princeton Univ. Press, Princeton, N.J., 1976
5. Ishizuka, M., K.S. Fu, and J.T.P. Yao, Inference procedures and uncertainty for the problem-reduction method, Inform. Sci., Vol. 28, 1982, pp. 179-206
6. Yager, R., Generalized probabilities of fuzzy events from fuzzy belief structure, Inform. Sci., Vol. 28, 1982, pp. 45- 62

Table 1. Comparison of interpretations

Data Set Number	Interpreted LOM	System Generated LOM
1	8-9	9-10
2	9	10
3	9	10
4	9	10-11
5	9	10
6	9	10-11
7	8.5-9	9-10
8	>10	11
9	9	9-10
10	9	9-10
11	9	9-10
12	7.5-8	7
13	>10	11-11.5
14	>10	11-11.5
15	10-11	11
16	11	11
17	9	10
18	7.5-8	8-9
19	8	9-10
20	10	11-11.5
21	10	11-11.5
22	10	11
23	10	11-11.5
24	9	9
25	9	9.5-10
26	10	11-11.5
27	9-10	11
28	9	9.5-10
29	10-11	11
30	10-11	10-11

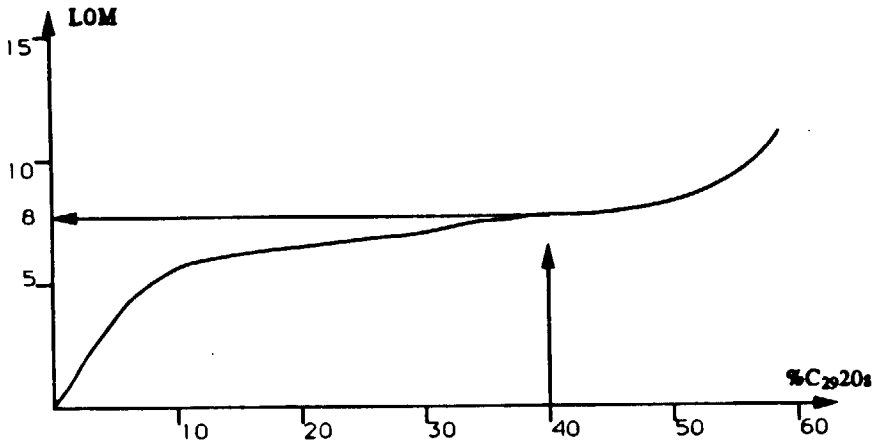


Figure 1: Experience based correlation curve between %C29 20s and LOM

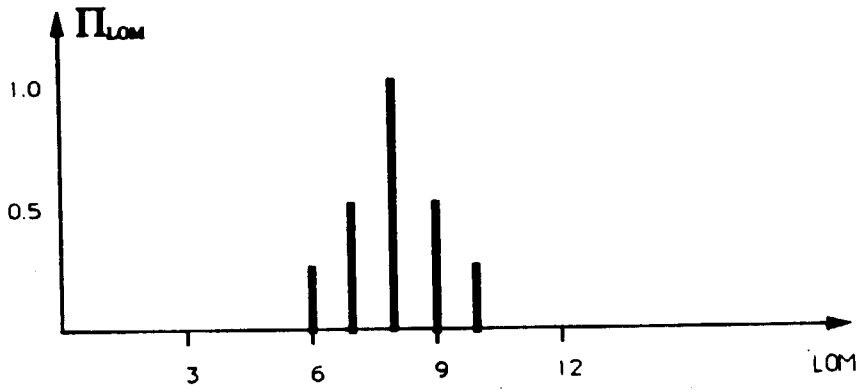


Figure 2: Possibility of LOM's induced from observing %C29 20s = 40

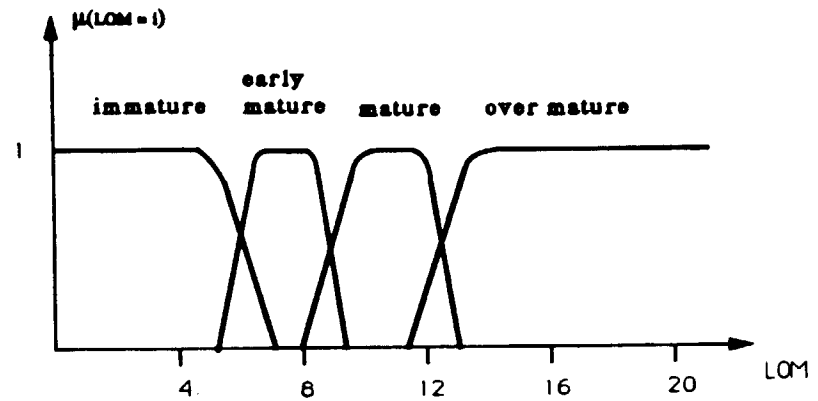


Figure 3a: Membership functions for maturity

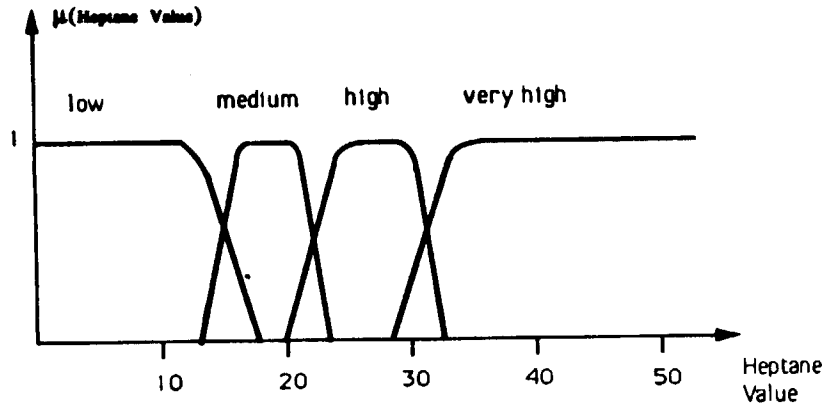


Figure 3b: Membership functions for heptane values

Towards Autonomous Fuzzy Control

Sujeet Shenoi

Mathematical and Computer Sciences
Keplinger Hall, University of Tulsa
Tulsa, OK 74104

Arthur Ramer
School of Computer Science and Engineering
University of New South Wales
P.O. Box 1, Kensington, NSW 2033, Australia

This research is partially supported by NSF-RIA Grant No. IRI9110709, OCAST Grant No. AR9-010, and by a grant from Sun Microsystems, Inc.

The efficient implementation of on-line adaptation in real time is an important research problem in fuzzy control. The goal is to develop autonomous self-organizing controllers [5] employing system-independent control meta-knowledge which enables them to adjust their control policies depending on the systems they control and the environments in which they operate. An autonomous fuzzy controller would continuously observe system behavior while implementing its control actions and would use the outcomes of these actions to refine its control policy. It could be designed to lie dormant when its control actions give rise to adequate performance characteristics but could rapidly and autonomously initiate real-time adaptation whenever its performance degrades. Such an autonomous fuzzy controller would have immense practical value. It could accommodate individual variations in system characteristics and also compensate for degradations in system characteristics caused by wear and tear. It could also potentially deal with black-box systems and novel control scenarios.

In this paper we report on our on-going research in autonomous fuzzy control. The ultimate research objective is to develop robust and relatively inexpensive autonomous fuzzy control hardware suitable for use in real time environments. This would represent an advancement over most existing fuzzy control systems. Due to the computational effort involved in implementing on-line adaptation fuzzy controllers are usually restricted to off-line adaptive configurations. They typically undergo extensive off-line training; once programmed, their control policies are set and cannot be changed in real time. We specifically focus on implementing autonomous behavior in look-up-table-based fuzzy logic controllers [1,6]. Such a controller simplifies the standard fuzzy control

algorithm by employing a look-up table generated off-line from an initial set of common sense fuzzy rules. The table acts as the control surface and represents "compiled" control knowledge. The look-up table for a two-term controller is a discrete function mapping error and error-change inputs to corresponding controller outputs; it gives rise to a 3-dimensional control surface.

The main challenge when implementing on-line adaptation in look-up table controllers is to effectively deal with the computational effort involved in recomputing the look-up table after each change to a membership function or fuzzy rule [2]. Adaptation typically corresponds to producing new "object-code " (look-up table) by repeatedly recompiling "source-code" (rules and membership functions). However, our approach bypasses the recompilation step required during controller adaptation by appropriately modifying the look-up table itself . Adaptation thus involves "hammering" the control surface itself. Controlled changes to the control surface have the overall effect of fine-tuning the control policy by quantitatively strengthening or weakening certain rules. Simulation experiments indicate that this approach is highly effective and robust. Moreover, it is possible to ensure that the qualitative characteristics of the original common-sense rules are retained during controller adaptation [2,3]. In this paper we describe our efforts at implementing autonomy in look-up-table-based fuzzy controllers. We start with a basic on-line adaptive algorithm combining gain coefficient tuning with direct look-up table modification [2,3]. We show how this algorithm can be further refined using control meta-knowledge to systematically guide and accelerate controller adaptation [4]. Finally, we describe our attempts at endowing the controller with common-sense knowledge which allows it to monitor its own performance and to autonomously trigger its own adaptation. The control algorithm for implementing autonomy in look-up table controllers is fast and relatively robust, but is still simple enough for hardware implementation. Simulation experiments indicate that it can effectively deal with a variety of systems. Moreover, its control meta-knowledge is powerful enough to effect rapid performance improvements even when the initial control policies are derived from incorrect rules or vacuous rule bases.

References

[1] M. Braae and D. A. Rutherford, Theoretical and linguistic aspects of the fuzzy logic controller, Automatica, vol. 15, 553-557, 1979.

[2] D. Mallampati and S. Shenoi, Self-organizing fuzzy logic control, in Knowledge Based Systems and Neural Networks: Techniques and Applications, R. Sharda, J. Y. Cheung, and W. J. Cochran (Eds.), Elsevier Science, New York, N.Y, pp. 271-282, 1991.

[3] D. Mallampati and S. Shenoi, Adaptive fuzzy logic controllers, Proceedings of the Fourth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Kauai, Hawaii, pp. 62-71, 1991.

[4] D. Mallampati and S. Shenoi, On-line adaptive fuzzy logic controllers, Proceedings of the 1992 International Fuzzy Systems and Intelligent Control Conference, Louisville, Kentucky, pp. 68-80, 1992.

[5] T. J. Procyk and E. H. Mamdani, A self-organizing fuzzy logic controller, Automatica, vol. 15, 15-30, 1979.

[6] D. A. Rutherford and J. C. Bloore, The implementation of fuzzy algorithms for control, Proceedings of the IEEE, vol. 64, 572-573, 1976.

Determining the Number of Hidden Units in Multi-Layer Perceptrons using F-Ratios

Ben H. Jansen and Pratish R. Desai

Department of Electrical Engineering and Bioengineering Research Center,
University of Houston, Houston, TX 77204-4793

Abstract

The hidden units in multi-layer perceptrons are believed to act as feature extractors. In other words, the outputs of the hidden units represent the features in a more traditional statistical classification paradigm. This viewpoint offers a statistical, objective approach to determining the optimal number of hidden units required. This approach is based on a F-ratio test, and proceeds in an iterative fashion. The method, and its application to simulated time-series data are presented.

1 Introduction

Artificial neural nets are increasingly being used for a variety of pattern recognition problems [1, 7, 8, 9]. Recently, Gallinari *et al.* [4] proved the formal equivalence between the linear multi-layer perceptron (MLP) and Discriminant Analysis (DA). Specifically, they noted that in a linear MLP, the first layer of weights realizes a DA of the input data, that is, projects the inputs onto a subspace so as to form well-aggregated clusters for each class. Experiments on problems with an increasing degree on nonlinearity demonstrated that DA on the hidden states gave similar performance as that of MLP. This suggests that hidden units activations can be interpreted as features. Consequently, feature selection techniques such as commonly used in statistical pattern recognition may be used to determine which hidden units are most significant, and which hidden units may be eliminated. One such method is presented here, and we show its usefulness in a problem involving the detection of specific waveforms in a time-series.

The results presented here are part of a larger study (see [2]), which investigated the use of recurrent and feed-forward neural networks for the detection of K-complexes in recordings of the electrical activity of the brain during sleep (electroencephalograms or EEGs). K-complexes are relatively large waves with a duration of between 500 and 1500 msec often seen during Sleep Stage 2. Automated detection of K-complex activity in the EEG is an important component of sleep stage EEG monitoring. Neural nets have been applied before to EEG waves with some success [3, 6].

2 Methods

The experiments described here involve the use of the multi-layer perceptron to detect bi-phasic triangular waveforms of various shapes in model-generated time-series. Both the triangular waveform and the time-series were made to resemble actual sleep EEG and K-complexes. The magnitude was extracted from segments of these time-series using the Fourier transform, and used as input to the neural nets. Once training was complete, a step-wise procedure was applied to determine the optimal number of hidden units required. The reduced net was then trained again, and tested using other data sets. The details of the data generation, net architecture and input, and net optimization procedure are provided next.

2.1 Data Generation

EEG data were obtained from six subjects. Five EEG channels (Fp1, F3, F4, T3, and T4) with observable K-complexes were used. An artificial data set was generated by producing a time series resembling actual EEG, to which a pattern representing a K-complex was added. EEG-like activity was produced through an 8th-order autoregressive (AR) model. The model coefficients were computed from actual EEG segments in the neighborhood (within 5 sec) of K-complexes (as identified by an electroencephalographer) to be used in generating "positive" examples, and from EEG taken far away from K-complexes to generate "negative" examples. Triangular patterns, resembling a K-complex, were placed in the artificial, "positive" EEG segments at various locations. No such pattern was added to the "negative" artificial EEG segments. Each positive or negative example consisted of 1000 sam-

ple points, representing 10 sec of data. The shape of the pattern differed between each of the positive examples. Specifically, the peak-to-peak amplitude of the pattern was varied in such a way that the ratio of the peak-to-peak amplitude of the pattern and the root-mean-square (rms) of the background activity would range between 0.05 and 0.15, the pattern was inserted at a random location, and the duration of the pattern varied randomly within a range similar to that of actual K-complexes. Three of such data sets were generated, referred to as the Train, Test1, and Test2 set, respectively. The Train and Test1 ("seen") data sets were generated from the same AR models, but different seed points were used to generate the EEG-like data and to control the shape and the location of the K-complex-like pattern. The Test2 data set ("unseen") was generated from the AR models obtained from EEG examples not included in the training data set.

2.2 Net Input and Architecture

Our basic approach was to compute the magnitude spectrum of 10 sec signal segments (using a FFT routine). These data were input to a multi-layer perceptron, which was trained using the backpropagation algorithm. Unless otherwise stated, the inputs to the net consisted of the magnitude at each of 64 frequency bins. A 512-point Fast Fourier Transform (FFT) was computed to obtain the magnitude, which was subsequently smoothed and reduced to 64 sample values by averaging over 8 adjacent points. These smoothed magnitude and phase values were then normalized between 0 and 1 for use as inputs to the neural network input nodes. Experiments with the hidden unit selection technique were performed on nets with 64 input units, one hidden layer with 8 units, and one or two output units.

2.3 Optimizing using Discriminant Analysis

The core of the optimization procedure derives from stepwise feature selection methods often used in statistical pattern recognition. In these approaches, the 'best' feature is selected from a pool of features using some criterion. All the pair-wise combinations of this best feature with any of the remaining features are explored to determine which is the 'best' pair, and if this additional feature has any discriminating power. If the answer to the last question is yes, triplets are formed by combining the best pair with any of the remaining

features. This process is repeated until it is found that adding a feature to the ones already selected does not lead to significant improvements in the criterion function.

In the present application, the outputs (activations) of the hidden units are treated as features. The Wilks' Λ is used as the criterion function to determine which feature should be selected. The Wilks' Λ is a multi-variate statistic that tests the equality of group means for the selected features [5]. The Λ may be converted to an approximate F-ratio. In the present method, the conditional F-ratio is used. The latter measures how much a given feature contributes to the group differences given the variables already selected. At each step the conditional F-ratios are computed for each feature. If a feature which has already been selected has a non-significant F-ratio, it is removed. If none of the features are removed, then the feature which creates the largest change in the criterion function is added to the selection. If none of the remaining features have a significant F-ratio, the procedure halts.

3 Results

In the first experiment, magnitude data were used to train a single output net with the Train data set. Upon convergence, training was halted, and the Train, Test1, and Test2 data sets were input to determine the classification performance of the net. A correct classification rate of 100% was found for Train, 92% for Test1, and 87% for Test2, respectively. Following this stage, the activations of the 8 hidden units for each example in the Train data set were recorded and subjected to the F-ratio test. The results shown in Table 1. Hidden units are listed in the order in which they were selected, together with their F-value at the time of selection.

The relatively large difference in F-value between unit 3 and 7 suggests that unit 3 is a very important feature. The scatter plot of the activations of unit 3 and 7, in response to the presentation of the training examples, is shown in Figure 1. It can be observed that the two classes are very well separated, except for a few positive examples that fall in the negative class cluster.

Mamelak, *et al.* [7] found that the overall performance of a single output net is usually worse than a 2 output net for a two-class problem. Even though each example can be assigned an unique pattern, with no indeterminate pat-

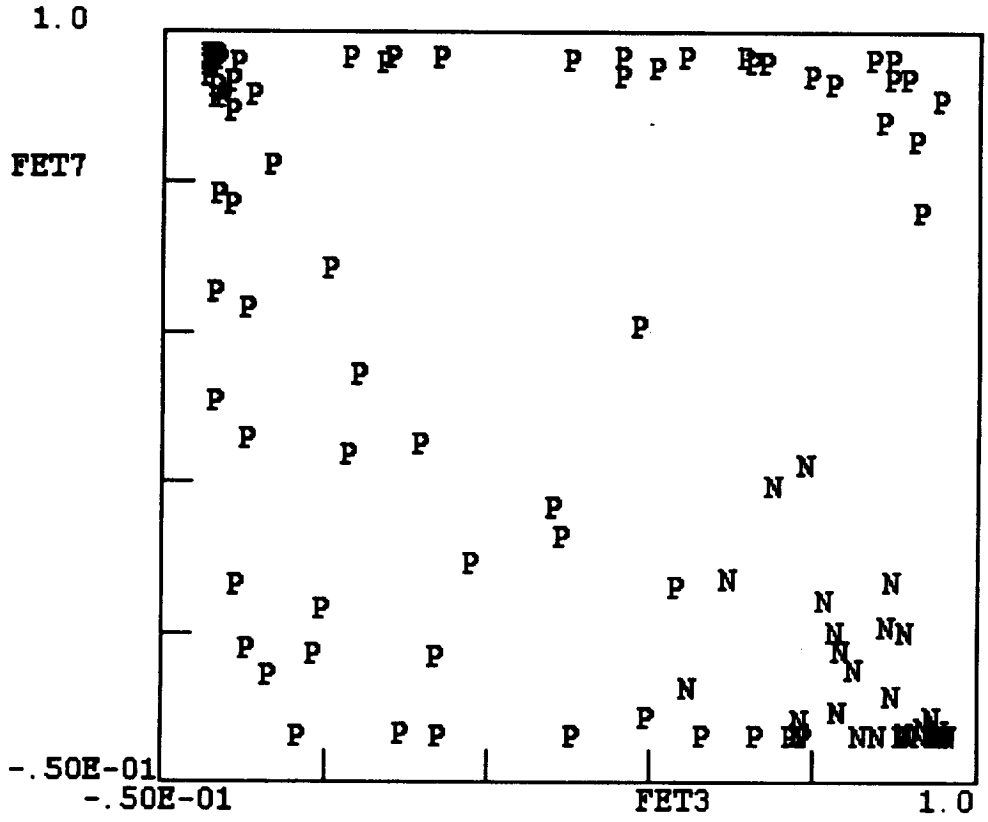


Figure 1: *Scatter plot of the activations of 2 hidden units (3rd and 7th), for the net with 8 hidden units and 1 output unit trained on the power spectra of exp.4.*

Table 1: *F-values obtained by performing an F-test on the 8 hidden unit outputs of a single-output net .*

Hidden Unit	F-value
3	155.88
7	37.77
8	68.73
2	43.43
5	43.51
6	34.28
1	4.25

terns, if a single output unit is used for a two-class problem, they found that the mapping between input and output patterns is actually too restricted, limiting the ability of the single-output net to fine-tune the threshold levels for all remaining patterns. We decided to explore this issue by applying the same training set as used above to a net with 8 hidden units and 2 output units. The net converged in 1187 cycles. The results of the F-test on the 8 hidden unit outputs are presented in Table 2.

Table 2: *F-values obtained by performing an F-test on the 8 hidden units activations of a net with 2 output units*

Hidden Unit	F-value
5	203.22
8	106.47
1	193.73
7	12.12
3	34.13
2	9.66

Observe that units 5, 8, and 1 produce large F-values, indicating their relative importance. Figure 2 shows the scatter plot for the first two selected hidden units. As shown, both classes are well clustered and are sitting well

in the corners of the square box. Compared to the results obtained with the net with one output unit (see Figure 1), the separation between the two classes is better defined. This confirms the observations made by Mamelak *et al.*.

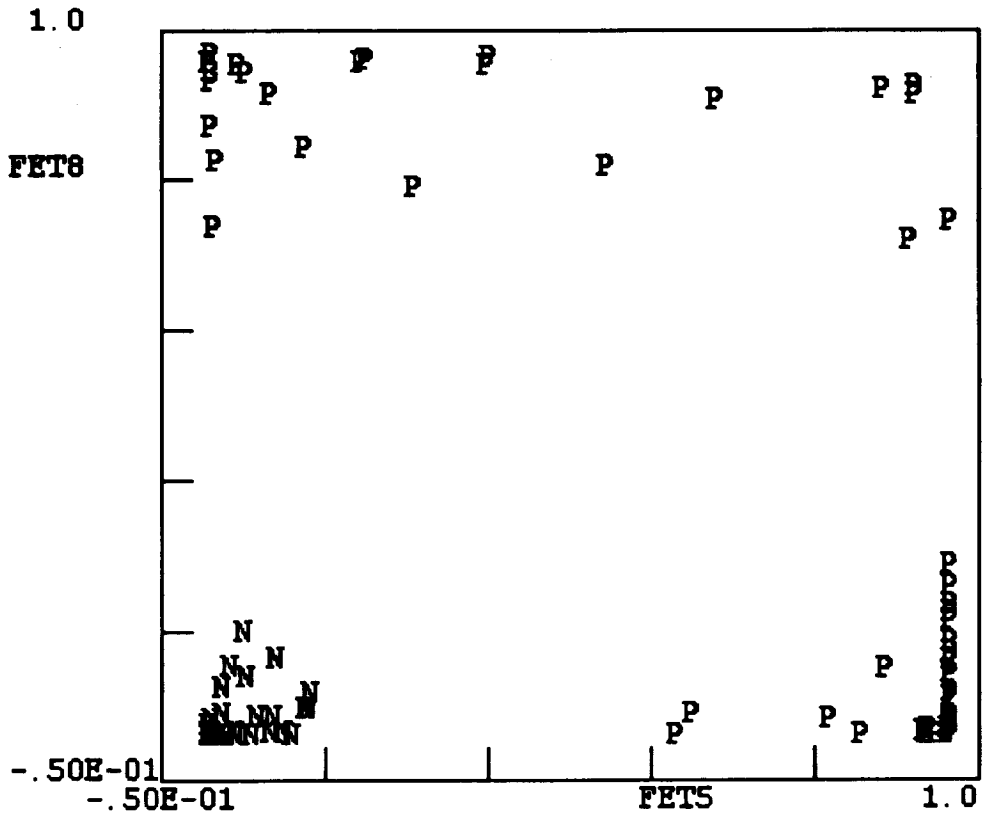


Figure 2: *Scatter plot of the activations of 2 hidden units (5th and 8th), for the net with 8 hidden units and 2 output units.*

Both of the aforementioned experiments suggest that a net with just two hidden units would perform as well as a net with 8 hidden units. This was explored in the next experiment involving a net with 2 hidden units and 2 output units. Again, training was done using the magnitude data, and it was found that the net converged in 1503 cycles. The scatter diagram of the

activations of the two hidden units is shown in Figure 3. As one can see,

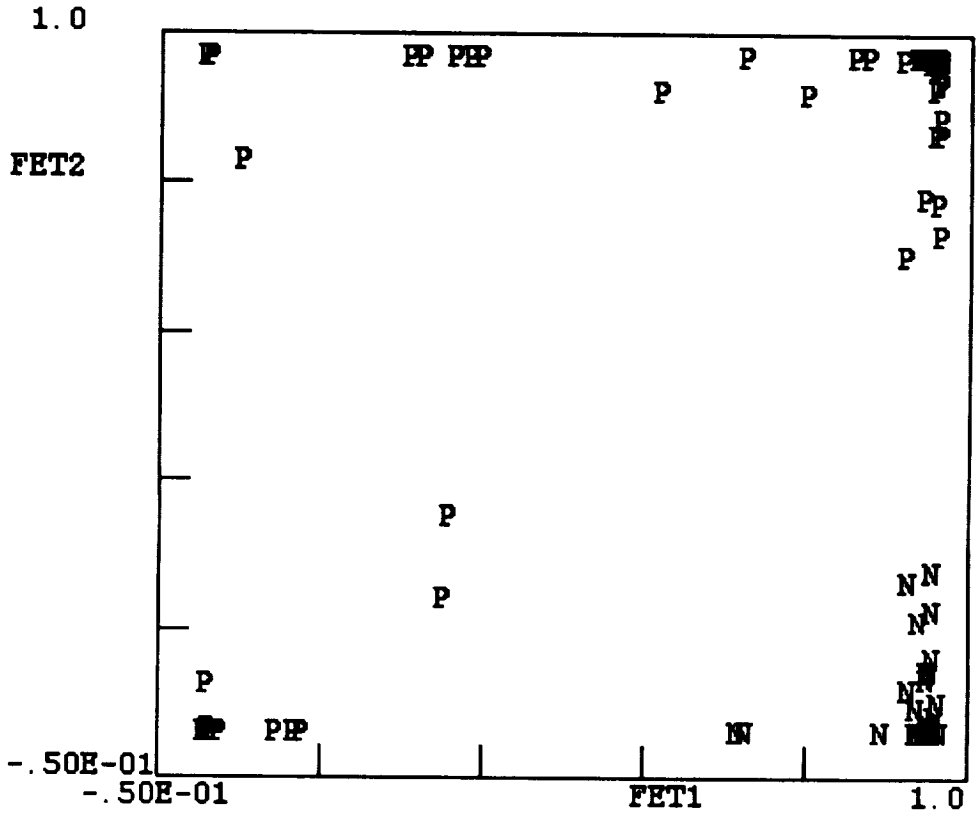


Figure 3: *Scatter plot of the activations of 2 hidden units for the net with 2 hidden units and 2 output units.*

the two classes are well-separated and occupying the corners of the feature space. The negative examples (N) are grouped into one corner, whereas the positive examples (P) are distributed over the other 3 corners. There was no specific relationship between the positive examples within one corner. This strongly suggests that a net with two hidden units should be sufficient to classify all the examples correctly. This was tested on the Train, Test1, and Test2 data sets, and although not perfect classification results were obtained for the two testing sets, the results were not significantly different from those

obtained with a net with 8 hidden units and 2 output units, and with a net with 8 hidden units and a single output.

4 Conclusions

We have presented a simple technique for the *a posteriori* determination of the hidden units required in a multi-layer perceptron. The method uses the fact that the hidden units appear to perform a discriminant analysis, essentially extracting features from the neural net input. The relative importance of each hidden unit can be assessed using an F-ratio test. In addition, the absolute value of the F-ratio provides insight in the degree of confidence one may place in the classifications produced by the net. For example, if the most significant hidden units have F-values barely above the level of significance, the classifying power of the net will be small.

The method described here is part of most widely available software packages for multi-variate data analysis, including BMDP and SPSS, making it very easy to apply this method.

References

[1] E. Barnard, R.A. Cole, M.P. Vea, and F.A. Alleva, "Pitch detection with a Neural-net classfier". *IEEE Transactions on Signal Processing*, vol. SP-39, pp. 298-307, 1991.

[2] P. R. Desai, *Waveform Detection Using Artificial Neural Networks*, M.Sc.-Thesis, Department of Electrical Engineering, University of Houston, 1991.

[3] R.C. Eberhart, R.W. Dobbins, W.R.S. and Webber, "EEG waveform analysis using casenet". *Proceedings of IEEE Engineering in Medicine and Biology Society 11th Annual International Conference*, pp. 2046, 1989.

[4] P. Gallinari, S. Thiria, F. Badran and F. Fogelman-Soulie, "On the relations between discriminant analysis and multi-layer perceptrons". *Neural Networks*, vol. 4, pp. 349-360, 1991.

[5] M. James, *Classification Algorithms,* John Wiley & Sons, New York, 1985.

[6] B.H. Jansen, "Artificial Neural Nets for K-Complex Detection". *IEEE Engineering in Medicine and Biology*, vol. 9, n. 3, pp. 50-52, 1990.

[7] A.N. Mamelak, J.J. Quattrochi, and J.A. Hobson, "Automated staging of sleep in cats using neural networks". *Electroencephalography and clinical Neurophysiology*, 79, pp. 52-61, 1991.

[8] T.J. Sejnowski, and R.P. Gorman, "Analysis of hidden units in a layered network trained to classify sonar targets". *Neural Networks*, vol. 1, pp. 75-89, 1988.

[9] A.H. Waibel, and K.J. Lang, "A time delay neural network architecture for isolated word recognition". *Neural Networks*, vol. 3, pp. 23-43, 1990.

A New Approach for Designing Self-Organizing Systems and Application to Adaptive Control

P.A. Ramamoorthy, Shi Zhang, Yueqing Lin and Song Huang
Department of Electrical and Computer Engineering
University of Cincinnati, M. L. 30
Cincinnati, Ohio 45221-0030
FAX:(513) 556-7326; TEL: (513) 556-4757
Email: pramamoo@nest.ece.uc.edu

Abstract

There is tremendous interest in the design of intelligent machines capable of autonomous learning and skillful performance under complex environments. A major task in designing such systems is to make the system plastic, and adaptive when presented with new and useful information and stable in response to irrelevant events. A great body of knowledge, based on neuro-physiological concepts, has evolved as a possible solution to this problem. Adaptive resonance theory (ART) is a classical example under this category. The system dynamics of an ART network is described by a set of differential equations with nonlinear functions.

An entirely new approach for designing self-organizing networks characterized by nonlinear differential equations is proposed in this paper. Similar to the neuro-physiological approach, the method presented here relies upon another area - that of passive nonlinear network theory. A passive nonlinear network is formed by proper interconnection of various nonlinear elements where each and every nonlinear element is constrained to be lossless or lossy. When energy storing elements are present in such a network, we can obtain a set of Input/Output relationships as nonlinear differential equations. The basic property that the network is lossy (consumes energy) ensures that the nonlinear differential equations obtained from the network would represent absolutely stable systems and this property holds as long as the individual element values are maintained in their permissible range of values. Thus, to deign complex nonlinear systems (a complex nonlinear plant plus a controller to optimize its performance, for example) and self-organizing systems, one simply has to force the system dynamics to mimic the dynamics of a properly constructed passive nonlinear network, a process akin to reverse engineering.

In our research which is in its early stages, we have developed the basis for the above approach and applied it with relative ease to a number of problems leading to encouraging results. The fruits of such an approach seem to be endless. For example, the approach can be applied to linear and nonlinear controller design (for linear and nonlinear plants), self tuning controllers, model reference adaptive controllers, self-organizing networks, adaptive IIR filter design, adaptive beam-forming, two-dimensional systems, fuzzy systems etc. In this paper, we provide some details of this approach and show results from some of these topics to show the power of this approach.

1 Introduction

There is currently tremendous interest and research activity in the areas of neural networks and fuzzy logic. The major driving force behind all these efforts is the hope that they can provide creative and novel solutions to the design of complex, autonomous and self-organizing systems. Fuzzy logic tries to mimic human approach to decision making when presented with fuzzy and often conflicting data and rules. Neural networks have originated from efforts to mimic neuro-physiological behavior.

From a functional point of view, both neural networks and fuzzy expert systems implement a mapping f: u $\rightarrow$ y, where u is an input vector, y the output vector and f is the mapping function which in general is a highly nonlinear function. In the case of fuzzy expert systems, the mapping is achieved through higher order logical relations between the inputs and the outputs where as in the case of neural networks, it is achieved through simple but repetitive linear and nonlinear operations. Fuzzy expert systems by themselves are feed-forward systems but their use in applications such as control lead to systems with feedback. Neural net architectures can either be feed-forward architectures or architectures with feedback. The system dynamics of feedback (also known as recurrent) neural networks are in general represented by a set of differential equations with nonlinear terms. Self-organizing techniques through which fuzzy rules and membership functions are learnt or improved are conceptually similar to the learning or training procedures in the neural network domain.

When we deal with systems with feedback, **the object of this paper**, stability becomes an important issue and has to take precedence over learning or self-organizing. However, it is not easy to establish stability of large-scale nonlinear systems. In fact, it is known that a first-order nonlinear equation with just one parameter can lead to stable, unstable and chaotic situations depending upon the value of that parameter. In this paper, we establish a frame work for designing such **feedback or recurrent systems that are guaranteed to be stable** with relative ease and show how it can be incorporated into fuzzy expert systems and neural networks with self-organizing capability.

2 The Basic Philosophy

As indicated before, our desire to mimic human cognition and functioning of neuro-physiological architectures has led to the two areas: Fuzzy logic and neural networks. The basic philosophy behind our new approach is to use "Passive Nonlinear Network Theory" to build new neural architectures with internal feedback. As will be shown, it leads to a new paradigm that is easier to handle (at least for engineers and computer scientists) than neuro physiology or human cognition.

A passive nonlinear network is simply an electrical network formed by proper interconnection of various nonlinear elements.The nonlinear elements in the network are constrained to be either lossless or lossy and the interconnections are such that the basic circuit laws are obeyed. As an example, the equation

$$i_R(t) = G\ tan^{-1}(v_R(t)) \qquad (1)$$

represents a two-terminal passive nonlinear resistor since

$$p(t) = i_R(t)v_R(t) \geq 0 \qquad \text{for all t}$$

indicating that the element consumes power all the time. In addition to the already known passive nonlinear resistors, we have defined a number of passive nonlinear elements. When such elements are interconnected with dynamic elements as shown in Fig.1, we can write down the dynamic equations for the network as a set of stable nonlinear equations:

$$[PI]\dot{X} = F[X, U] \qquad (2)$$

where

$$X = [i_{L_1}, i_{L_2}, ..., i_{L_L}, v_{C_1}, v_{C_2}, ...v_{C_C}]^T$$

$$P = [L_1, L_2, ..., L_L, C_1, C_2, ..., C_C]$$

$$U = [I_1, I_2, ..., I_I, V_1, V_2, ..., V_V]$$

I, an identity matrix of size $(L_L + C_C) * (L_L + C_C)$

F, a vector of nonlinear functions of X and U

and

'.' indicates differentiation.

It can be observed that the set of equations given in (2) represents a stable network or system as long as the element values are in the permissible range so as to retain the lossy or lossless property. The stability property holds good even if we incorporate complex, exotic nonlinear elements. If such a system is turned on with only initial stored energy in the dynamic elements, the state variables will all go to zero as time progresses.

Reader familiar with the ART networks [1-4] will recognize immediately the similarity in the structure of the set of equations (2) obtained from the passive network and the set of equations characterizing ART networks:

$$\epsilon\dot{x}_k = -x_k + (1 - Ax_k)J_k^+ - (B + Cx_k)J_k^- \qquad k = 1\ \ to\ \ M + N \qquad (3)$$

$$\dot{Z}_{ij} = k_1 f(x_j)[-E_{ij}Z_{ij} + h(x_i)] \qquad i = 1\ \ to\ \ M; \qquad (4)$$

$$\dot{Z}_{ji} = k_2 f(x_j)[-E_{ji}Z_{ji} + h(x_i)] \qquad j = i\ \ to\ \ M + 1 \qquad (5)$$

where the descriptions of the various terms can be found in the references. However, a major difference between ART dynamic equations and the set of equations derived from

the passive networks is that the former has been derived from an understanding of difficult cognition processes and slow evolution (ART-1 to ART-2 and so on). The passive network approach enables us to come up with a number of entirely different sets of equations with relative ease as will be obvious from the examples given. Another difference is that the ART equations are written in such a way that some state variables are forced to reach saturation (similar to introducing activity or nonlossy property in some of the elements in the network). The "Winner-Take-All" portion of the ART network belongs to this category.

The basic philosophy behind our design approach is to 1) define a number of nonlinear elements obeying the lossless or lossy condition, 2) form a generic network architecture that would lead to most general form of nonlinear state equations and 3) force the state equations corresponding to the system under consideration to obey the form given in equation (2). The property that the equations represent a stable network whether they are set to a fixed mode or in a self-organizing mode makes this approach unique and promising.

3 Simulation Examples

In this section, we provide a number of examples to illustrate the applicability of the approach to a number of problem domains.

3.1 Nonlinear/Adaptive Controller Design

Consider a single-degree-of-freedom manipulator represented by a 2nd - order transfer function as shown in Fig.2. The task is to design an adaptive controller which will force the manipulator to follow a desired trajectory.

The classical approach in adaptive control is to define a control input

$$T(t) = -k_1 q - k_2 \dot{q} \tag{6}$$

and adapt the coefficients K=$[k_1, k_2]^T$ using

$$[\dot{k}] = -c\frac{\partial e^2}{\partial k} \tag{7}$$

where e corresponds to the tracking error.

A network based controller using the same form for control input as in (6) is given by

$$\begin{aligned} \dot{k}_1 &= -(k_1 + \frac{4}{\pi} tan^{-1}(k_1)) + q\dot{q} + k_2 + 1 \\ \dot{k}_2 &= -(k_2 + \frac{4}{\pi} tan^{-1}(k_2)) - k_2 + 3 \end{aligned} \tag{8}$$

where the controller equations have been obtained so as to force the plant and the controller combination mimic a fourth-order passive nonlinear dynamic network [1] and assuming that the desired output of the plant as $q_d, \dot{q}_d = 0$. The constants in the equations are chosen to let k_1, k_2 to 1 as t $\rightarrow \infty$.

Another set of controller equations based on the network approach is given by

$$\dot{k}_1 = -(k_1 + \frac{4}{\pi} tan^{-1}(k_1)) + q\dot{q} + k_2 + 1$$

$$\dot{k}_2 = -(k_2 + \frac{4}{\pi} tan^{-1}(k_2)) + \dot{q}^2 - k_2 + 3 \qquad (9)$$

We provide this addition controller expression simply to illustrate how easy it is to derive alternate forms.

We have shown some simulation results in Figs. 3A-C using the controller expressions in (8) . The simulations were carried out assuming different initial values for $q, \dot{q}$ and some initial values for k_1 and k_2 and the task of the controller is to move the manipulator to location zero. Figs. 3A and 3B shows $q, \dot{q}$ as a function of time and Fig 3C shows a phase plane plot (q V$_s$ $\dot{q}$) of the manipulator. It can be noted that the adaptive controller does a good job of controlling the manipulator. Though we are not including the results, we have performed the simulations with a) error in the plant coefficient values, b) a sudden change in the values of the friction and compliance coefficients and c) unmodeled dynamics represented by another second-order transfer function. The results were really impressive and showed the robustness of the nonlinear adaptive controller obtained using the network approach. It should be noted here that nonlinear functions such as $tan^{-1}(k_1)$, initial and final values for k_1 and k_2 etc were chosen randomly with no efforts to optimize anything.

3.2 Application to Fuzzy Control

Fuzzy logic [5] has been used to design controllers for various systems and processes [ref. 6, for example]. The classical approach is to find the difference between the actual and desired outputs and the derivatives of the outputs and use a fuzzy expert system to generate the control input(s) (see Fig. 4A). Thus, the plant and the controller form a closed loop and the stability of the feedback system could become an issue. The architecture could be easily modified to mimic a passive network (as shown in Fig. 4B) and hence guarantee stability.

To illustrate this concept, we have taken a third order model example used in ref.[7], retained only the two dominant poles and used the fuzzy look-up table given in that paper with some modifications to generate the fuzzy controller output F($e, \dot{e}$). Denoting the transfer function of the plant as

[1] We are not going into complete details of deriving the equations as we are in the process of patenting some of the nonlinear elements and their applications.

$$H(s) = \frac{b}{s^2 + as + b} = \frac{Y(s)}{U(s)} \qquad (10)$$

with u as input to the plant, and y the output of the plant, the dynamics of the complete system is given by

$$\begin{aligned} \dot{y} &= y_1 \\ \dot{y}_1 &= -by - ay_1 + u \\ u &= kF(e, \dot{e}) \\ \dot{k} &= -y_1 F(e, \dot{e}) - k - \frac{4}{\pi} tan^{-1}(k) + u_1 \end{aligned} \qquad (11)$$

where u_1 is chosen to force k to a particular value as the plant output moves to the target value. The responses of the plant using the classical fuzzy control approach and the new network based approach for two values of $k(\infty)$ are shown in Fig 5. It can be noted that there is some improvement in the response [2]. However, the key point here is that the system represented by equation (11) will remain stable and robust for external disturbances.

3.3 Application to Model Reference Adaptive Control (A Simple Self-Organizing System)

Here we consider the application of the passive network approach to model reference adaptive control (MRAC) where the aim is to design a controller such that the combined system (plant + controller) mimics a given model. The problem is quite simple if the plant model and the parameters are known precisely. If that is not the case or if the parameters vary with respect to time, an adaptive controller is the preferred solution. The set-up for the classical adaptive control as well as the new network based approach are shown in Fig. 6 . The classical approach is to use a gradient based technique to update the controller parameters but is known to be prone to instability etc.

The set of equations comprising the whole adaptive system based on the network approach is given by footnoteWe used subscripts m, p, t to denote closed-loop-model, plant and time-evolving model respectively.

$$\begin{aligned} \theta_m &= \theta_t(t) + k \qquad (closed - loop - model \ \ requirement) \\ \dot{x}_p &= -\theta_p x_p - k x_p + r \qquad (plant \ \ dynamics) \\ \dot{x}_t &= -\theta_t x_t - k x_p + r = k(x_t - x_p) - \theta_m x_t + r \\ &\quad (dynamics \ of \ the \ time-evolving \ model \ of \ the \ plant) \\ \dot{k} &= x_p x_p + (x_p - x_t)x_t - F_1(x_p - x_t)(k + \frac{4}{\pi} tan^{-1}(k)) - x_m^2 \end{aligned} \qquad (12)$$

[2] It appears that the original fuzzy controller has already been optimized very well.

$$(controller\ \ dynamics)$$

where

$$F_1(x_p - x_t) = \begin{cases} 1 & when |x_p - x_t| \geq 1 \\ |x_p - x_t| & otherwise \end{cases}$$

Again, the expression for the controller dynamics was obtained by forcing the three different dynamics to mimic a highly coupled passive network. The set of equations were simulated using some initial values for x_p, x_m, x_t, k and r(t), a sinusoidal function. The time evolution of k(t) is shown in Fig.7. It can be noted that k tends to its expected value of 0.5 in nearly 1500 iterations, a nice feat for an almost randomly chosen controller function. The key point to be noted from this example is that self-organizing networks can also be designed very easily using the new approach.

It is noted above that the classical MRAC approach can lead to instability under certain conditions. This could probably be explained using network concepts by noting that there are two closed loops in the whole system, one involving the plant and the controller and the other involving the plant, adaptive control law and the controller. The two loops were formed by some mathematical considerations and do not seem to be coupled as well as a network based approach and the complete system is not constrained to be passive and lossy. Hence the possibility for instability.

4 Summary

An entire new and exciting approach for designing nonlinear systems and self-organizing networks is proposed in this paper. The approach is based on a simple yet powerful concept that of using properties of properly constructed nonlinear passive networks. We have shown examples from different areas indicating how the approach can be applied to many different areas and the possible applications seem to be endless. The preliminary results obtained so far are very encouraging. We believe that it is just the beginning of a new era for a powerful methodology which can compete with approaches mimicking human cognition.

5 Reference

1. S. Grossberg, "Adaptive pattern classification and universal recoding II; Feedback, expectation, olfaction and illusions," Biol. Cyber. 23, 187, 1976.

2. G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self–organizing neural pattern recognition machine," Comp. Vision, Graphics, and Image Proc., vol. 37,pp. 54–115, 1987.

3. G. A. Carpenter and S. Grossberg, "ART2: self–organization of stable category recognition codes for analog input patterns," Applied Optics, vol. 26, pp. 4919–4930, 1987.

4. G. A. Carpenter and S. Grossberg, "ART3: hierarchical search using chemical transmitters in self–organizing pattern recognition architectures," Neural Networks, vol. 3, pp. 129–152, 1990.

5. L. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," IEEE Trans. Sys., Man, Cybern., vol. smc-3, pp. 28–44, 1973.

6. S. Chiu, S. Chand, D. Moore and A. Chaudhary, "Fuzzy logic for control of roll and moment for a flexible wing aircraft," IEEE Control Systems Magazine, pp. 42–48, June 1992.

7. S. Tzafestas and N. P. Papanikolopoulos, " Incremental Fuzzy Expert PID Control," IEEE Trans. on Industrial Electronics, pp. 365–371, vol. 37, No. 5, Oct. 1990.

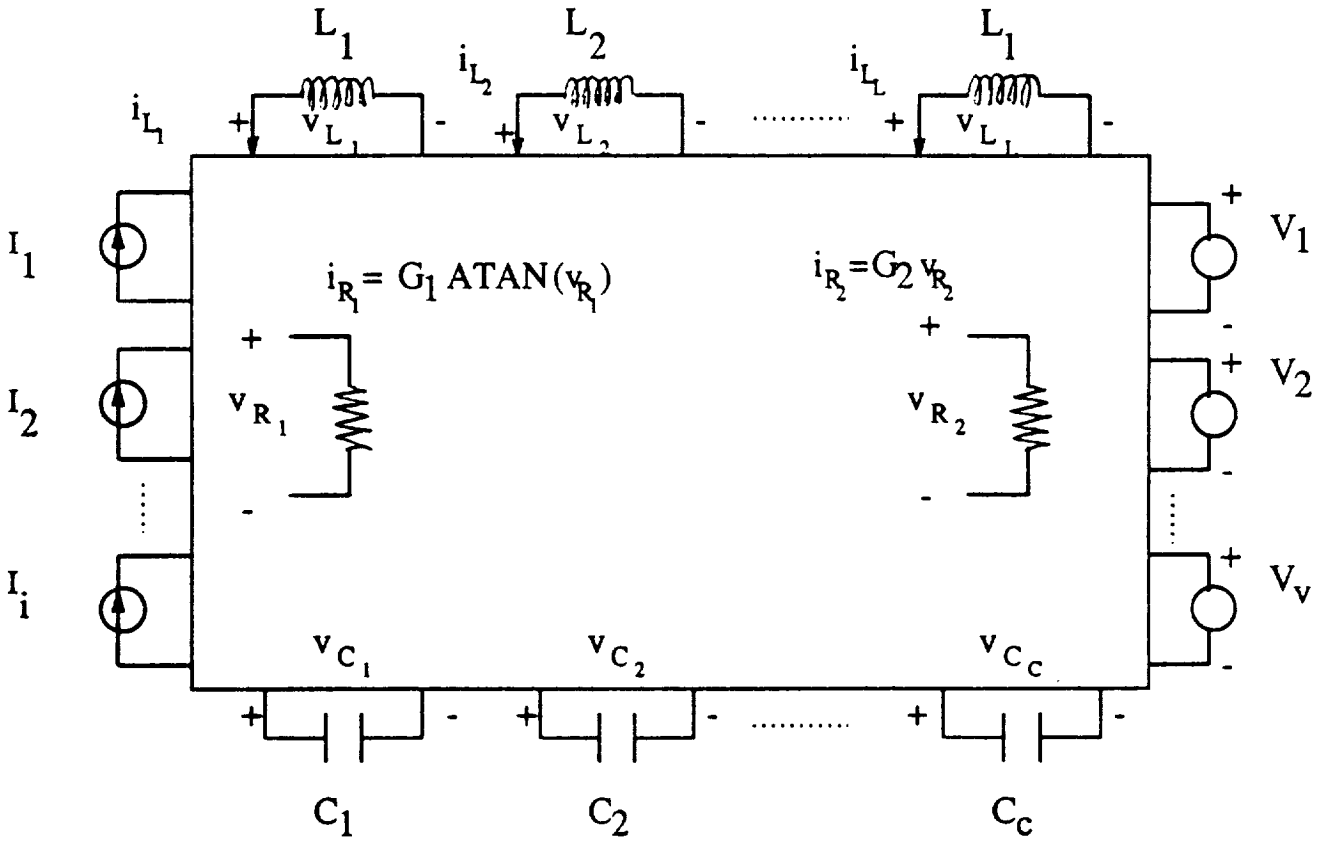
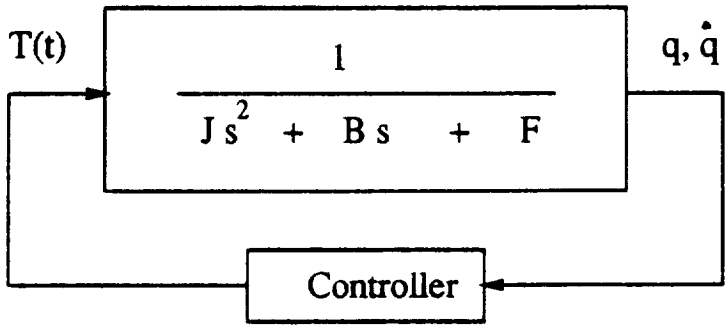


Fig. 1 A passive nonlinear dynamical network.



q: joint angle
q̇: joint velocity
T(t): applied torque

J: moment of inertia = 1
B: viscous friction = 5
F: compliance coefficient = 0.7

Fig. 2 A manipulator with a single degree of freedom.

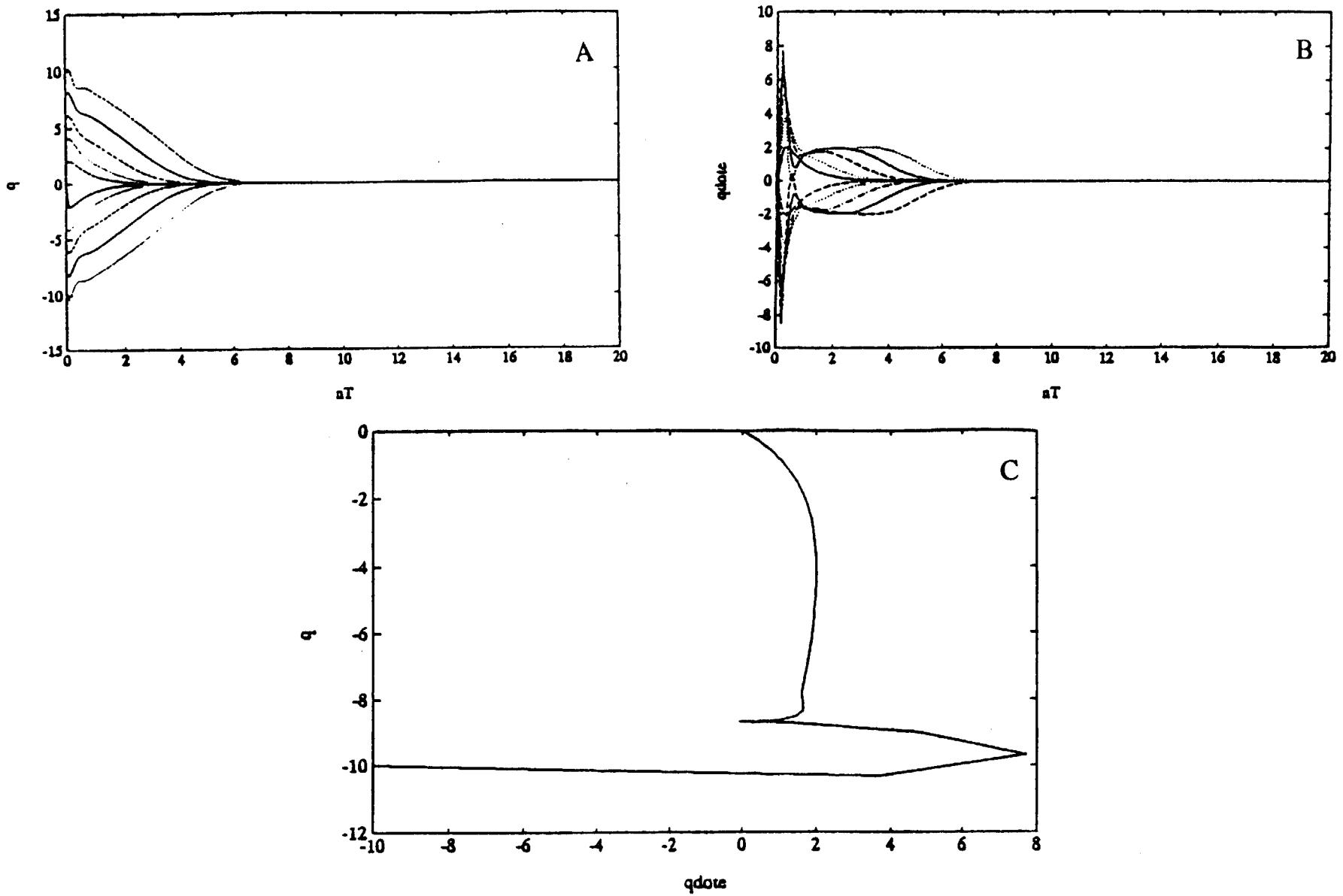
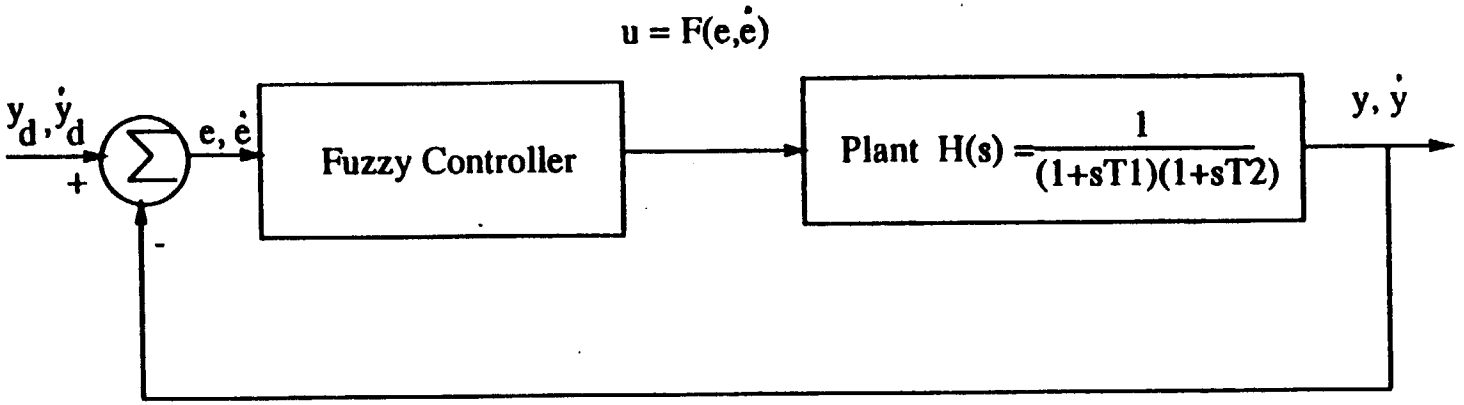
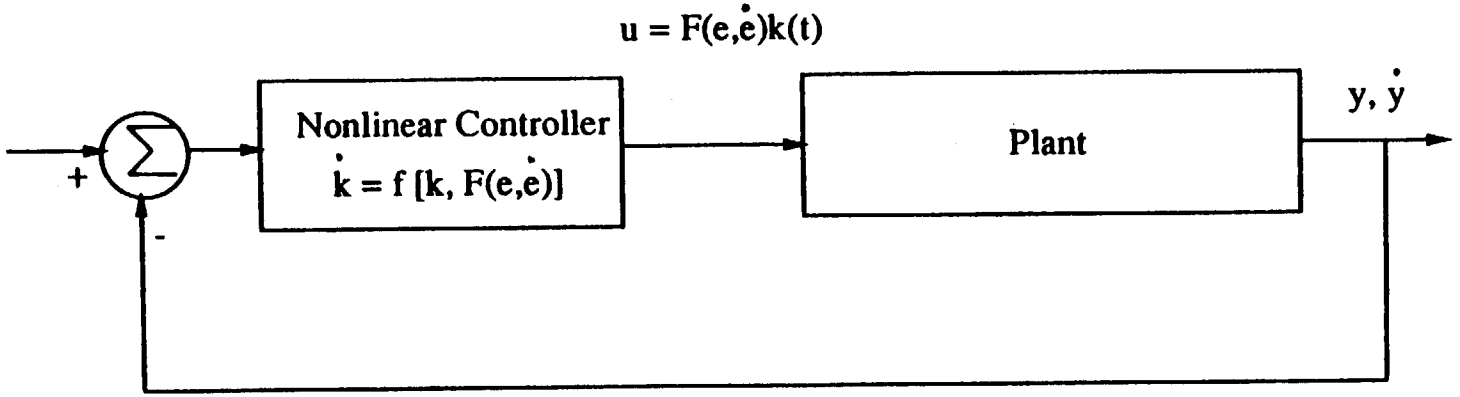


Fig. 3 A. The plant response as a function of time. B. The derivative of the plant as a function of time. C. phase-plane plot of the plant.



A. Classical Fuzzy Control Approach.



B. New Nonlinear Network Based Approach.

Fig. 4 Application to Fuzzy Control.

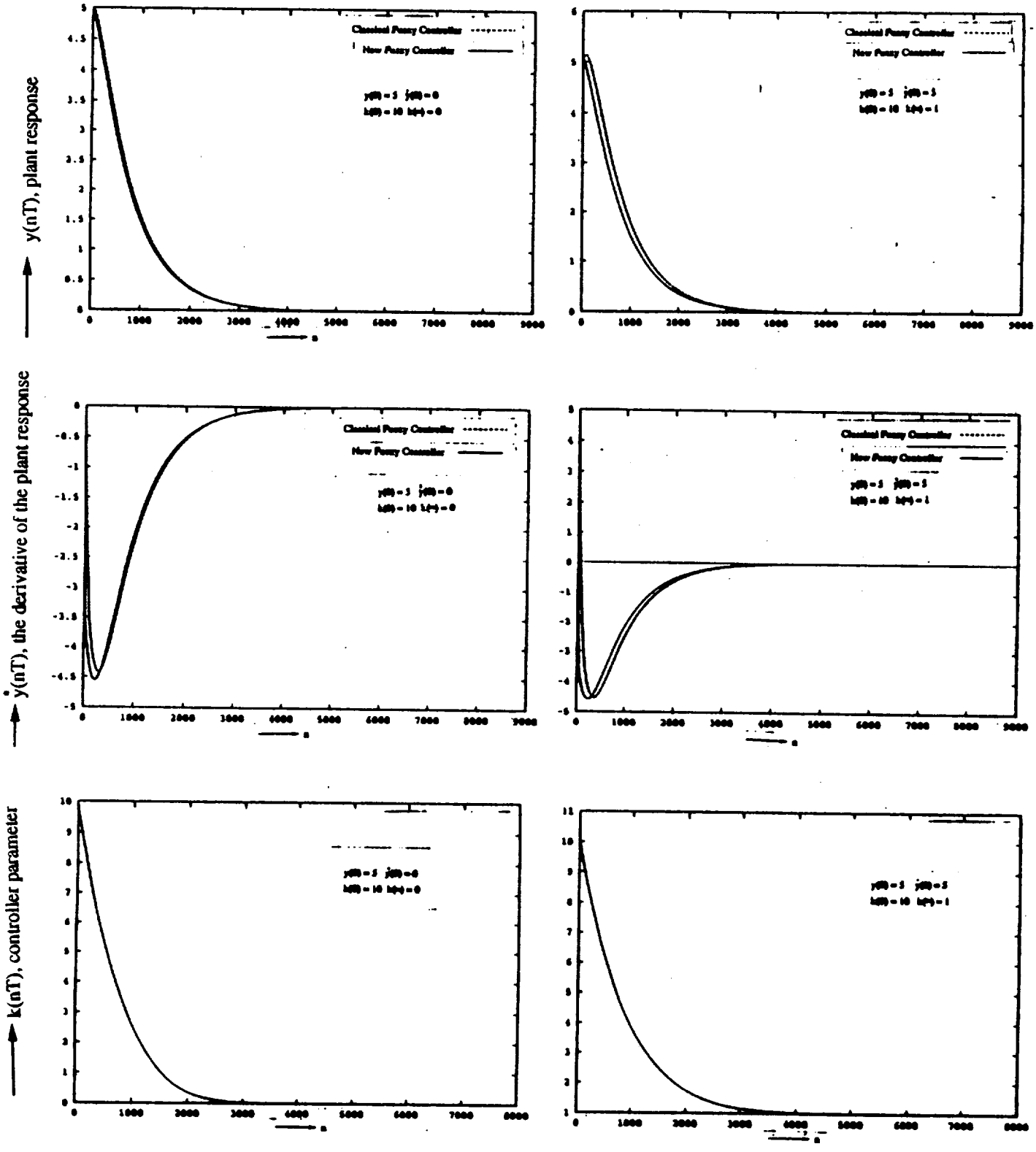


Fig. 5 Response of the plant using classical and network based fuzzy controller.

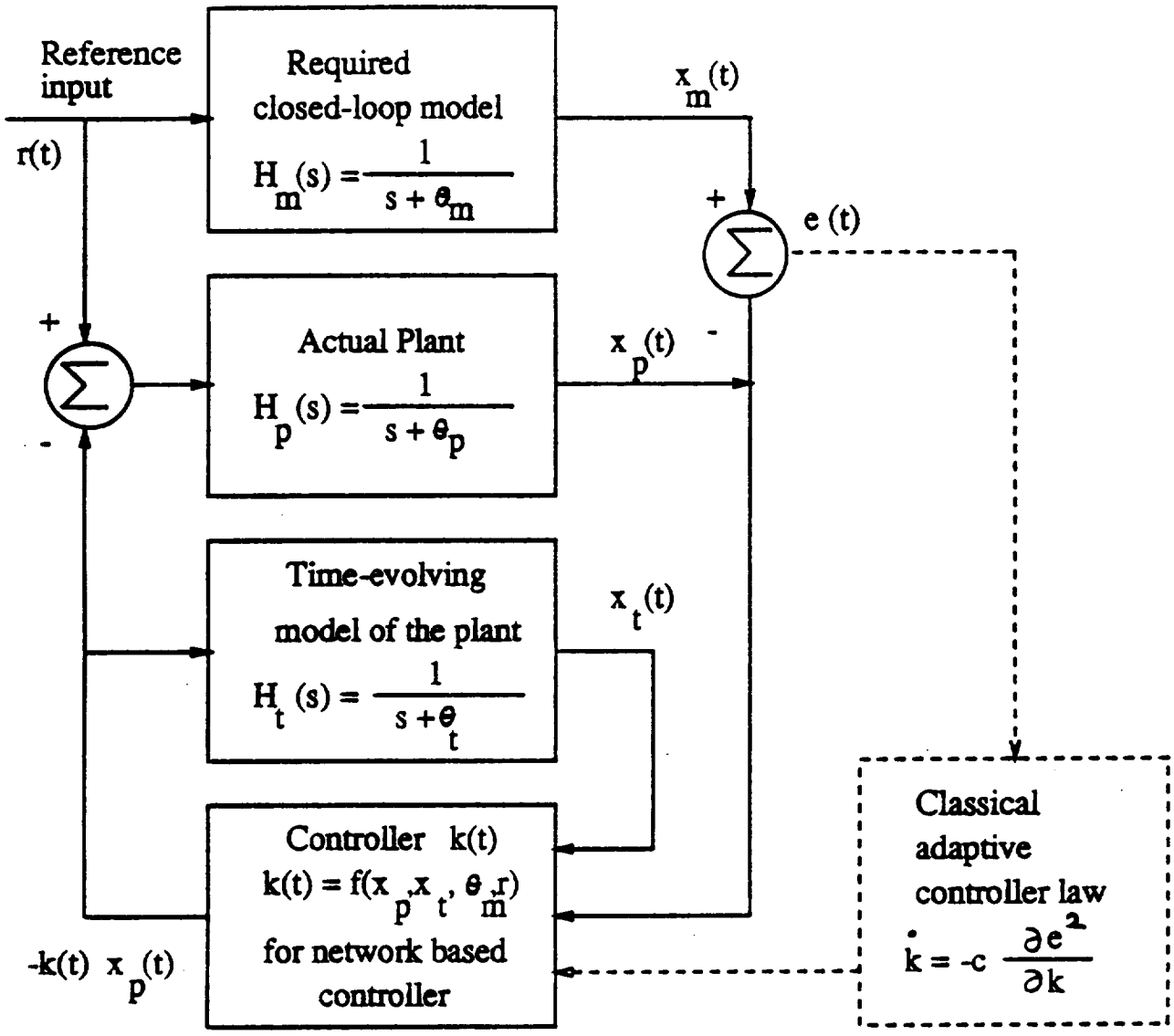


Fig. 6 Model reference adaptive control using classical adaptive control law and the new network based approach. $\theta_m = 1.$ and $\theta_p = 0.5$ used in the simulation.

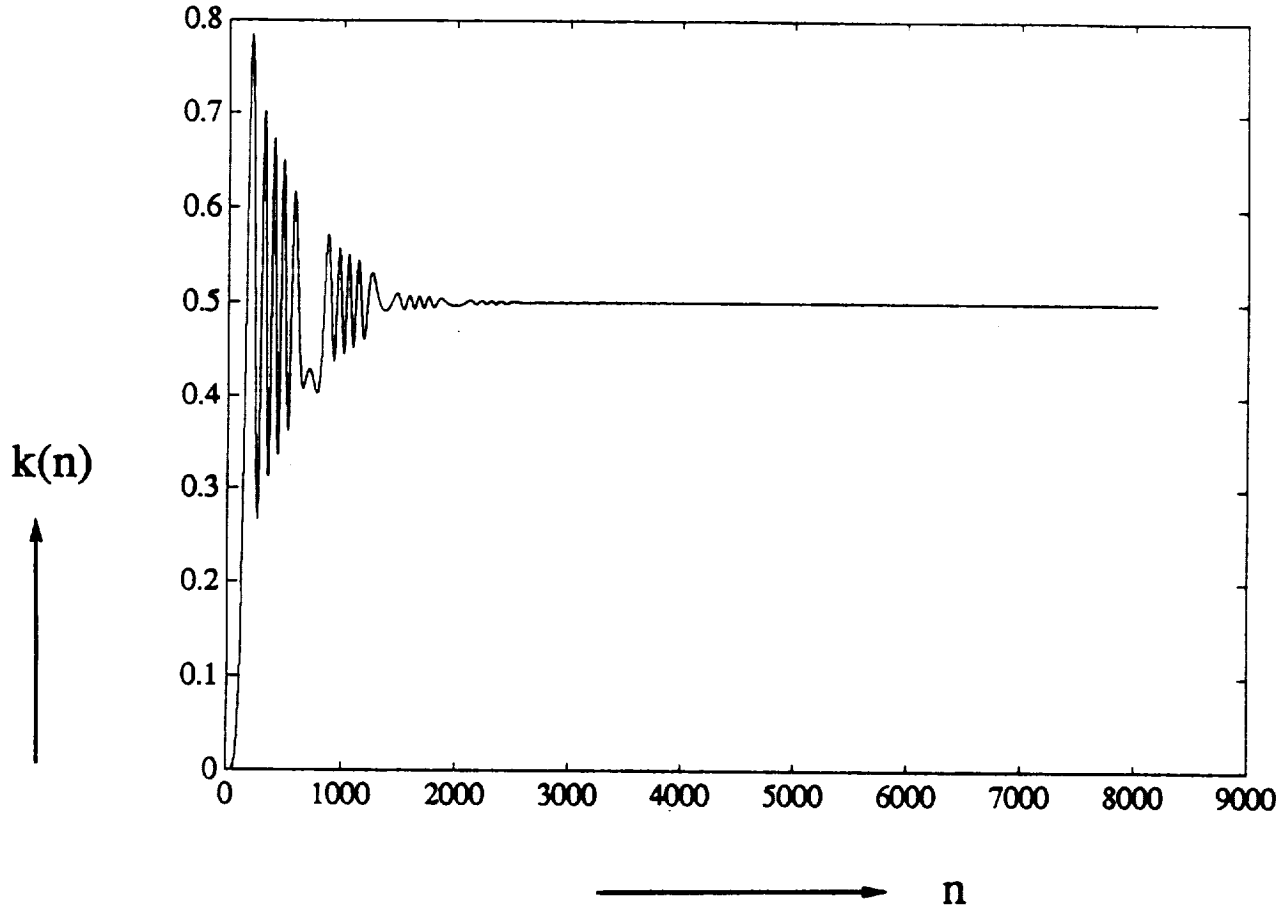


Fig. 7 Time evolution of the controller parameter k.

ADAPTIVE FUZZY SYSTEM FOR 3-D VISION

FINAL REPORT
ON
NASA/JSC CONTRACT
NAG 9-509

CONTRACT PERIOD: 4-01-91 TO 3-31-92

SUBMITTED BY

Dr. Sunanda Mitra
Principal Investigator and
Associate Professor,
Department of Electrical Engineering.
Texas Tech University
Lubbock, Texas 79409-3102

To
Dr. Robert Lea
Fuzzy Logic Technical Coordinator,
Information System Directorate,
NASA- Johnson Space Center PT4
Houston, Texas 77058

April 15,1992

PROJECT SUMMARY

A novel adaptive fuzzy system using the concept of the Adaptive Resonance Theory (ART) type neural network architecture and incorporating fuzzy c-means (FCM) system equations for reclassification of cluster centers has been developed.

The Adaptive Fuzzy Leader Clustering (AFLC) architecture is a hybrid neural-fuzzy system which learns on-line in a stable and efficient manner. The system uses a control structure similar to that found in the Adaptive Resonance Theory (ART-1) network to identify the cluster centers initially. The initial classification of an input takes place in a two stage process; a simple competitive stage and a distance metric comparison stage. The cluster prototypes are then incrementally updated by relocating the centroid positions from Fuzzy c - Means (FCM) system equations for the centroids and the membership values. The operational characteristics of AFLC and the critical parameters involved in its operation are discussed. The performance of the AFLC algorithm is presented through application of the algorithm to the Anderson Iris data, and laser-luminescent fingerprint image data. The AFLC algorithm successfully classifies features extracted from real data, discrete or continuous, indicating the potential strength of this new clustering algorithm in analyzing complex data sets.

This hybrid neuro-fuzzy AFLC algorithm will ehnance analysis of a number of difficult recognition and control problems involved with Tethered Satellite Systems and on-orbit space shuttle attitude controller.

I. INTRODUCTION

Cluster analysis has been a significant research area in pattern recognition for a number of years[1]-[4]. Since clustering techniques are applied to the unsupervised classification of pattern features, a neural network of the Adaptive Resonance Theory (ART) type[5],[6] appears to be an appropriate candidate for implementation of clustering algorithms[7]-[10]. Clustering algorithms generally operate by optimizing some measures of similarity. Classical, or crisp, clustering algorithms such as ISODATA[11] partition the data such that each sample is assigned to one and only one cluster. Often with data analysis it is desirable to allow membership of a data sample in more than one class, and also to have a degree of belief that the sample belongs to each class. The application of fuzzy set theory[12] to classical clustering algorithms has resulted in a number of algorithms[13]-[16] with improved performance since unequivocal membership assignment is avoided. However, estimating the optimum number of clusters in any real data set still remains a difficult problem[17].

It is anticipated, however, that a valid fuzzy cluster measure implemented in an unsupervised neural network architecture could provide solutions to various real data clustering problems. The present work describes an unsupervised neural network architecture[18],[19] developed from the concept of ART-1[5] while including a relocation of the cluster centers from FCM system equations for the centroid and the membership values[2]. Our AFLC system differs from other fuzzy ART-type clustering algorithms [20],[21] incorporating fuzzy min-max learning rules. The AFLC presents a new approach to unsupervised clustering, and has been shown to correctly classify a number of data sets including the Iris data. This fuzzy modification of an ART-1 type neural network, i.e. the AFLC system, allows classification of discrete or analog patterns without a priori knowledge of the number of clusters in a data set. The optimal number of clusters in many real data sets is, however, still dependent on the validity of the cluster measure, crisp or fuzzy, employed for a particular data set.

II. ADAPTIVE FUZZY LEADER CLUSTERING SYSTEM AND ALGORITHM

A. AFLC System and Algorithm Overview

AFLC is a hybrid neural-fuzzy system which can be used to learn cluster structure embedded in complex data sets, in a self-organizing, stable manner. This system has been adapted from the concepts of ART-1 structure which is limited to binary input vectors[5]. Pattern classification in ART-1 is achieved by assigning a prototype vector to each cluster that is incrementally updated[10].

Let $X_j = \{ X_{j1}, X_{j2}, \ldots X_{jp} \}$ be the j th input vector for $1 \leq j \leq N$ where N is the total number of samples in the data set and p is the dimension of the input vectors. The initialization and updating procedures in ART-1 involve similarity measures between the bottom-up weights (b_{ki} where k = 1,2,...,p) and the input vector (X_j), and a verification of X_j belonging to the i th cluster by matching of the top-down weights (t_{ik}) with X_j. For continuous-valued features, the above procedure is changed as in ART-2[6]. However if the ART-type networks are not made to represent biological networks, then a greater flexibility is allowed to the choice of similarity metric. A choice of Euclidean metric is made in developing the AFLC system while keeping a simple control structure adapted from ART-1.

Figure 1

Figures 1(a) and 1(b) represent the AFLC system and operation for initialization and comparison of cluster prototypes from input feature vectors, which may be discrete or analog. The updating procedure in the AFLC system involves relocation of the cluster prototypes by incremental updating of the centroids v_i, (the cluster prototypes), from FCM system equations[2] for v_i and μ_{ij} as given below :

$$V_i = \frac{1}{\sum_{j=1}^{N_i} (\mu_{ij})^m} \sum_{j=1}^{N_i} (\mu_{ij})^m X_j \; ; \qquad 1 \leq i \leq C \tag{1}$$

$$\mu_{ij} = \frac{\left(\frac{1}{\|X_j - v_i\|^2}\right)^{1/(m-1)}}{\sum_{k=1}^{C}\left(\frac{1}{\|X_j - v_k\|^2}\right)^{1/(m-1)}}; \quad 1 \le i \le C; \; 1 \le j \le N \tag{2}$$

where N_i is the number of samples in cluster i and C is the number of clusters. The v_i's and μ_{ij}'s are recomputed over the entire data sample N.

As described here, AFLC is primarily used as a classifier of feature vectors employing an on-line learning scheme. Figure 1(a) shows a p-dimensional discrete or analog-valued input feature vector, X to the AFLC system. The system is made up of the comparison layer, the recognition layer, and the surrounding control logic. The AFLC algorithm initially starts with the number of clusters (C) set to zero. The system is initialized with the input of the first feature vector X. Similar to leader clustering, this first input is said to be the prototype for the first cluster. The normalized input feature vector is then applied to the bottom-up weights in a simple competitive learning scheme, or dot product. The node that receives the largest input activation Y is chosen as the prototype vector as is done in the original ART-1.

$$Y_i = \max\{\sum_{k=1}^{p} X_{jk} b_{ki}\}; \qquad 1 \le j \le N \tag{3}$$

Therefore the recognition layer serves to initially classify an input. This first stage classification activates the prototype or top-down expectation (t_{ik}) for a cluster, which is forwarded to the comparison layer. The comparison layer serves both as a fan-out site for the inputs, and the location of the comparison between the top-down expectation and the input. The control logic with an input enable command allows the comparison layer to accept a new input as long as a comparison operation is not currently being processed. The control logic with compare imperative command disables the acceptance of new input and initiates comparison between the cluster prototype of Y_i i.e., the centroid v_i and the current input vector, using equation (4). The reset signal is activated when a mismatch of the first and

second input vectors occurs according to the criterion of a distance ratio threshold as expressed by equation (4)

$$R = \frac{\sqrt{d^2(X_j, v_i)}}{\frac{1}{N_i}\sum_{k=1}^{N_i}\sqrt{d^2(X_k, v_i)}} < \tau \tag{4}$$

where : k = 1....N_i the number of samples in class i and $\sqrt{d^2(X_j, v_i)}$ is the Euclidean distance as indicated in equation(5).

$$d^2(x_j - v_i) = \left\| x_j - v_i \right\|^2 \tag{5}$$

If the ratio R is less than a user-specified threshold τ, then the input is found to belong to the cluster originally activated by the simple competition. The choice of the value of τ is critical and is found by a number of initial runs. Preliminary runs with τ varying over a range of values yield a good estimate of the possible number of clusters in unlabeled data sets.

When an input is classified as belonging to an existing cluster, it is necessary to update the expectation (prototype) and the bottom-up weights associated with that cluster. First, the degree of membership of X to the winning cluster is calculated. This degree of membership, μ, gives an indication, based on the current state of the system, of how heavily X should be weighted in the recalculation of the class expectation. The cluster prototype is then recalculated as a weighted average of all the elements within the cluster. The update rules are as follows: the membership value μ_{ij} of the current input sample X_j in the winning class i, is calculated using equation (2), and then the new cluster centroid for cluster i is generated using equation (1). As with the FCM, m is a parameter which defines the fuzziness of the results and is normally set to be between 1.5 and 30. For the following applications, m was experimentally set to 2.

The AFLC algorithm can be summarized by the following steps :

1. Start with no cluster prototypes, C = 0.

2. Let X_j be the next input vector.

3. Find the first stage winner Y_i, as the cluster prototype with the maximum dot-product.

4. If Y_i does not satisfy the distance ratio criterion, then create a new cluster and make its prototype vector be equal to X_j. Output the index of the new cluster.

5. Otherwise, update the winner cluster prototype Y_i by calculating the new centroid and membership values using equations (1) and (2). Output the index of Y_i. Go to Step 2.

A flow chart of the algorithm is shown in Figure 2.

Figure 2

III. OPERATIONAL CHARACTERISTICS OF AFLC

A. Match-based Learning and the Search

In match-based learning, a new input is learned only after being classified as belonging to a particular class. This process ensures stable and consistent learning of new inputs by updating parameters only for the winning cluster and only after classification has occurred. This differs from error-based learning schemes, such as backpropagation of error, where new inputs are effectively averaged with old learning resulting in forgetting and possibly oscillatory weight changes. In [5] match-based learning is referred to as resonance, hence the name Adaptive Resonance Theory.

Because of its ART-like control structure, AFLC is capable of implementing a parallel search when the distance ratio does not satisfy the thresholding criterion. The search is arbitrated by appropriate control logic surrounding the comparison and recognition layers of Figure 1. This type of search is necessary due to the incompleteness of the classification at the first stage. For illustration, consider the two vectors (1,1) and (5,5). Both possess the same unit vector. Since the competition in the bottom-up direction consists of measuring how well the normalized input matches the weight vector for each class i, these inputs would both excite the same activation pattern in the recognition layer. In operation, the comparison layer serves to test the hypothesis returned by the competition performed at the recognition

layer. If the hypothesis is disconfirmed by the comparison layer, i.e. $R > \tau$, then the search phase continues until the correct cluster is found or another cluster is created. Normalization of the input vectors (features) is done only in the recognition layer for finding the winning node. This normalization is essential to avoid large values of the dot products of the input features and the bottom-up weights and also to avoid initial misclassification arising due to large variations in magnitudes of the cluster prototypes. The search process, however, renormalizes only the centroid and not the input vectors again.

B. Determining the Number of Output Classes

AFLC utilizes a dynamic, self-organizing structure to learn the characteristics of the input data. As a result, it is not necessary to know the number of clusters a priori; new clusters are added to the system as needed. This characteristic is necessary for autonomous behavior in practical situations in which nonlinearities and nonstationarity are found.

Clusters are formed and trained, on-line, according to the search and learning algorithms. Several factors affect the number, size, shape, and location of the clusters formed in the feature space. Although it is not necessary to know the number of clusters which actually exist in the data, the number of clusters formed will depend upon the value of τ. A low threshold value will result in the formation of more clusters because it will be more difficult for an input to meet the classification criteria. A high value of τ will result in fewer, less dense clusters. For data structures having overlapping clusters, the choice of τ is critical for correct classification whereas for nonoverlapping cluster data, the sensitivity of τ is not a significant issue. In the latter case the value of τ may vary over a certain range, yet yielding correct classification. Therefore the sensitivity of τ is highly dependent on specific data structure as shown in Figure 1(c). The relationship between τ and the optimal number of clusters in a data set is currently being studied.

C. Dynamic Cluster Sizing

As described earlier, τ is compared to a ratio of vector norms. The average distance parameter for a cluster is recalculated after the addition of a new input to that cluster; therefore, this ratio (R) represents a dynamic description of the cluster. If the inputs are dense around the cluster prototype, then the size of the cluster will decrease, resulting in a more stringent condition for membership of future inputs to that class. If the inputs are widely grouped around the cluster prototype, then this will result in less stringent conditions for membership. Therefore, the AFLC clusters have a self-scaling factor which tends to keep dense clusters dense while allowing loose clusters to exist.

D. The Fuzzy Learning Rule

In general, the AFLC architecture allows learning of even rare events. Use of the fuzzy learning rule in the form of equations (1) and (2), maintains this characteristic. In weighted rapid learning[5], the learning time is much shorter than the entire processing time and the adaptive weights are allowed to reach equilibrium on each presentation of an input, but the amount of change in the prototype is a function of the input and its fuzzy membership value (μ_{ij}). Noisy features which would normally degrade the validity of the class prototype are assigned low weights to reduce the undesired affect. In the presence of class outliers, assigning low memberships to the outliers lead to correct classification. Normalization of membership is not involved in this process. However, a new cluster of outliers only can be formed during the search process[22]. Development of such outlier/noise cluster in AFLC is currently under progress.

Weighted rapid learning also tends to reinforce the decision to append a new cluster. This is due to the fact that, by definition, the first input to be assigned to a node serves as that node's first prototype, therefore, that sample has a membership value of one. Future inputs are then weighted by how well they match the prototype. Although the prototype does change over time, as described in the algorithm, each sample retains its weight which tends to limit moves away from the current prototype. Thus the clusters possess a type of inertia which tends to stabilize the system by making it more difficult for a cluster to radically change its prototype in the feature space.

Finally, the fuzzy learning rule is stable in the sense that the adaptive weights represent a normalized version of the cluster centroid, or prototype. As such, these weights are bounded on [0,1] and are guaranteed not to approach infinity.

E. AFLC as a General Architecture

As with most other clustering algorithms, the size and shape of the resultant clusters depends on the metric used. The use of any metric will tend to influence the data toward a solution which meets the criteria for that metric and not necessarily to the best solution for the data. This statement implies that some metrics are better for some problems than are others. The use of a Euclidean metric is convenient, but displays the immediate problem that it is best suited to simple circular cluster shapes. The use of the Mahalanobis distance accounts for some variations in cluster shape, but its non-linearity serves to place constraints on the stability of its results. Also, as with other metrics, the Euclidean and Mahalanobis distance metrics lose meaning in an anisotropic space.

IV. TESTS AND RESULTS: FEATURE VECTOR CLASSIFICATION

A. Clustering of the Anderson Iris Data

The Anderson Iris data set[23], consists of 150 4-dimensional feature vectors. Each pattern corresponds to characteristics of one flower from one of the species of Iris. Three varieties of Iris are represented by 50 of the feature vectors. This data set is popular in the literature and gives results by which AFLC can be compared to similar algorithms.

We had 52 runs of the AFLC algorithm for the Iris data for 13 different values of τ, with 4 runs for each τ. Figure 1(c) shows the τ-C graph. With Euclidean distance ratio and τ ranging between 4.5 and 5.5, the sample data was classified into 3 clusters with only 7 misclassifications. The misclassified samples actually belonged to Iris versicolor, cluster #2, and were misclassified as Iris virginica, cluster

#1. From Figure 1(c) it can be observed that the optimal number of clusters can be determined from the τ -C graph as the value of C that has $\frac{dC}{d\tau} = 0;\ for\ C \neq 1$, for the maximum possible range of τ.

Figure 3, shows the input Iris data clusters using only three features for each sample data point. Figure 4a shows the computed centroids of the three clusters based on all four features. The intercluster Euclidean distances are found to be 1.75 (d_{12}), 4.93 (d_{23}), and 3.29 (d_{13}). d_{ij} is the intercluster distance between clusters i & j. The comparatively smaller intercluster distance between clusters 1 and 2 indicates the proximity of these clusters. Figure 4b shows a confusion matrix that summarizes the classification results.

Figure 3

Figure 4

B. Classification of Noisy Laser-luminescent Fingerprint Image Data

Fingerprint matching poses a challenging clustering problem. Recent developments in automated fingerprint identification systems employ primitive and computationally intensive matching techniques such as counting ridges between minutae of the fingerprints[24]. Although the technique of laser luminescent image acquisition of latent fingerprint provide often identifiable images[25], these images suffer from amplified noise, poor contrast and nonuniform intensity. Conventional enhancement techniques such as adaptive binarization and wedge filtering provide enhancement at the expense of significant loss of information necessary for matching. Recent work[26] presents a novel three stage matching algorithm for fingerprint enhancement and matching. Figure 5b shows the enhanced image of 5a subsequent to selective Fourier spectral enhancement and bandpass filtering. We used the AFLC algorithm to cluster three different classes of fingerprint images using seven invariant moment features[26],[27] computed from images that are enhanced[26]. A total of 24 data samples are used, each sample being a 7-dimensional moment feature vector. These moment invariants are a set of nonlinear functions which are invariant to translation, scale, & rotation. The three higher order moment features

are given less weights thus reducing the affect of noise and leading to proper classification. The τ-C graph for the fingerprint data in Figure 1(c) shows a range of τ from 3.0 to 4.5 for which proper classification resulted. The fingerprint data has also been correctly classified by a k-nearest neighbor clustering using only four moment features[26]. Euclidean distances of these clusters indicate that the clusters are well separated which is consistent with the comparatively larger range of τ found for proper classification. Figures 5a and 5b represent one fingerprint class before and after enhancement. Figure 6a shows the computed centroids of three fingerprint clusters. Figure 6b shows a confusion matrix that indicates correct classification results.

Figure 5, Figure 6

V. CONCLUSION

It is possible to apply many of the concepts of AFLC operation to other control structures. Other approaches to Fuzzy ART are being explored[20],[21] that could also be used as the control structure for a fuzzy learning rule. Choices also exist in the selection of class prototypes. With some modification, any of these techniques can be incorporated into a single AFLC system or a hierarchical group of systems. The characteristics of that system will depend upon the choices made.

While AFLC does not solve all the problems associated with unsupervised learning, it does possess a number of desirable characteristics. The AFLC architecture learns and adapts on-line, such that it is not necessary to have a priori knowledge of all data samples or even of the number of clusters present in the data. However the choice of τ is critical and requires some a priori knowledge of the compactness and separation of clusters in the data structure. Learning is match-based ensuring stable, consistent learning of new inputs. The output is a crisp classification and a degree of confidence for that classification. Operation is also very fast, and can be made faster through parallel implementation. A recent work[28] shows a different approach to neural-fuzzy clustering by integrating Fuzzy C - means model with Kohonen neural networks. A comparative study of these recently developed neural-fuzzy clustering algorithms is needed. Future work will involved further modification of the AFLC system and algorithm for analyzing simulation data of the TSS system[29] and for automated attitude controller design of on-orbit shuttle[30].

VI. ACKNOWLEDGMENT

This work was partially supported by a grant from NASA-JSC under contract #NAG9-509, a grant from Northrop Corporation under contract #HHH7800809K, and a Texas Instruments Graduate Student Fellowship to Scott Newton through Texas Tech University. The authors gratefully acknowledge many excellent and thoughtful suggestions from J. Bezdek for improving this paper.

VII. REFERENCES

[1] R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis, John Wiley and Sons, New York , 1973.

[2] J. C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum, New York, 1981.

[3] A. K. Jain and R. C. Dubes, Algorithms for Clustering Data, Englewood Cliffs, NJ, Prentice-Hall, 1988.

[4] K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press Inc., San Diego, 1990.

[5] G. A. Carpenter and S. Grossberg, "A Massively Parallel Architecture for a Self-organizing Neural Pattern Recognition Machine," Computer Vision, Graphics & Image Processing, 37, pp. 54-115, 1987.

[6] G.A. Carpenter and S. Grossberg, "ART2: Self organization of stable category recognition codes for analog patterns" Applied Optics 2 pp. 4919-4938, 1987

[7] R. Lippmann, "An Introduction to Computing with Neural Networks," IEEE ASSP Magazine, No 4, pp. 4-21, 1987.

[8] L. I. Burke, "Clustering Characterization of Adaptive Resonance," Neural Networks, Vol. 4, pp 485-491, 1991.

[9] Y.H. Pao, Adaptive Pattern Recognition and Neural Networks, Addison-Wesley Publishing Co., 1989.

[10] B. Moore, "ART 1 and Pattern Clustering," Proc. of the 1988 Connectionist Models Summer School, Eds. D. Touretzky, G. Hinton, and T. Sejnowski, Morgan Kaufman Publishers, San Mateo, CA, 1988.

[11] G. H. Ball and D. J. Hall, "ISODATA, an Iterative Method of Multivariate Data Analysis and Pattern Classification," Proceedings of the IEEE International Communications Conference, Philadelphia, PA, June 1966.

[12] L. A. Zadeh, "Fuzzy Sets," Information and Control, 8, pp. 338-353, 1965.

[13] J. C. Bezdek, "A Physical Interpretation of Fuzzy ISODATA," IEEE Transactions on Systems, Man and Cybernetics, vol. 6, pp. 387-389, 1976.

[14] J. C. Bezdek, "A Convergence Theorem for the Fuzzy ISODATA Clustering Algorithms," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-2, no. 1, pp. 1-8, 1980.

[15] E. E. Gustafson and W. C. Kessel, "Fuzzy Clustering with a Fuzzy Covariance Matrix," Proceedings of the IEEE CDC, San Diego, CA, pp. 761-766, 1979.

[16] I. Gath and A. B. Geva, "Unsupervised Optimal Fuzzy Clustering," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, no. 7, pp.773-781, 1989.

[17] X. L. Xie and G. Beni, "A Validity Measure for Fuzzy Clustering," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-13, No. 2, p. 841-847, 1991.

[18] S. C. Newton and S. Mitra, "Self-organizing Leader Clustering In A Neural Network Using A Fuzzy Learning Rule," SPIE Proc. on Adaptive Signal Processing, Vol. 1565, July 1991.

[19] S. C. Newton and S. Mitra, "Applications of Neural Networks in Fuzzy Pattern Recognition," presented at the Research Conference on Neural Networks For Vision and Image Processing, Boston University, May 10-12, 1991.

[20] P. Simpson, "Fuzzy Adaptive Resonance Theory," Proceedings of the Neuroengineering Workshop, H. Szu and M. Sayeh, eds. September, 1990.

[21] G. Carpenter, S. Grossberg, and D. Rosen, "Fuzzy ART: An Adaptive Resonance Algorithm for Rapid, Stable Classification of Analog Patterns," Proceedings of the 1991 International Joint Conference on Neural Networks," July, 1991.

[22] R.N. Dave, "Characterization and detection of noise in clustering" Pattern Recognition Letters 12, No. 11, November, 1991

[23] E. Anderson, "The Irises of the Gaspe Peninsula," Bull. Amer. Iris Soc., Vol. 59, pp. 2-5, 1935.

[24] "Computer Graphics in the Detective Business", IEEE Computer Graphics and Applications, Vol. 5 pp. 14-17, April 1985.

[25] C.R. Menzel, "Latent Fingerprint Development with Lasers", ASTM Standardization News, Vol. 13 pp. 34-37, March 1985.

[26] E. Kaymaz, "Fingerprint Analysis and Matching". M.S.E.E. Thesis, Texas Tech University, December 1991.

[27] M.K. Hu, "Visual Pattern Recognition by Moment Invariants" IRE Trans. Information Theory, Vol. IT-8, pp. 179-187, 1962

[28] J.C. Bezdek, E.C.K. Tao, and N.R. Pal, "Fuzzy Kohonen Clustering Networks", Proc. of IEEE Int. Conference on Fuzzy Systems, pp. 1035-1043, San Diego, March 8-12, 1992.

[29] R. N. Lea, J. Villarreal, Y,. Jani and C. Copeland, "Fuzzy Logic Based Tether Control". Proc. of NAFIPS-'91, p. 398-402, May 14-17, 1991.

[30] R. N. Lea, J. Hoblit and Y. Jani, "Performance Comparison of a Fuzzy Logic Based Attitude Controller with the Shuttle On-orbit Digital Auto-pilot", Proc. of NAFIPS-'91, p. 291-295, May 14-17, 1991.

FIGURE CAPTIONS

Figure 1. Operation characteristics of AFLC Architecture . 1(a) shows the initial stage of identifying a cluster prototype, 1(b) shows the comparison stage using the criterion of Euclidian distance ratio $R > \tau$ to reject new data samples to the cluster prototype. The reset control implies the deactivation of the original prototype and activation of a new cluster prototype and 1(c) shows the τ - c graph for choosing τ for unlabelled datasets.

Figure 2. Flow-chart of the AFLC Algorithm

Figure 3. Iris Data Represented by Three-Dimensional Features

Figure 4a. Computed Centroids of Three Iris Clusters Based on All Four Feature Vectors

Figure 4b. Iris Cluster Classification Results shown as a confusion matrix

Figure 5a. A Noisy Laser-luminescent Fingerprint Image

Figure 5b. The Enhanced Image of 5a. by Selective Fourier Spectral Filtering

Figure 6a. Computed Centroids of Three Fingerprint Clusters in Seven-Dimensional Vector Space

Figure 6b. Fingerprint Data Classification Results

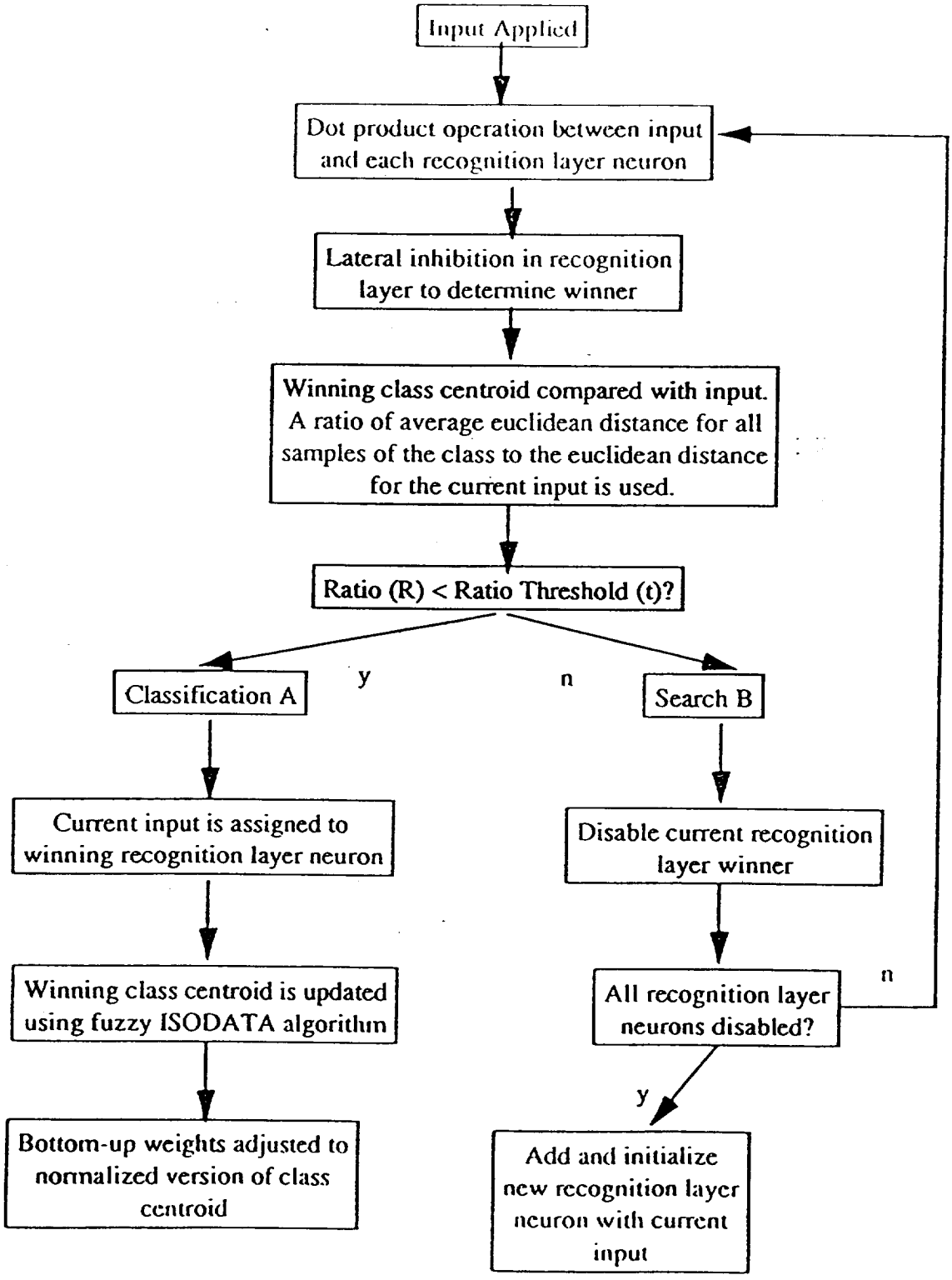

Input Applied
Dot product operation between input and each recognition layer neuron
Lateral inhibition in recognition layer to determine winner
Winning class centroid compared with input. A ratio of average euclidean distance for all samples of the class to the euclidean distance for the current input is used.
Ratio (R) < Ratio Threshold (t)?
y
n
Classification A
Search B
Current input is assigned to winning recognition layer neuron
Disable current recognition layer winner
Winning class centroid is updated using fuzzy ISODATA algorithm
All recognition layer neurons disabled?
n
y
Bottom-up weights adjusted to normalized version of class centroid
Add and initialize new recognition layer neuron with current input

Figure 2.

ANDERSON IRIS DATA

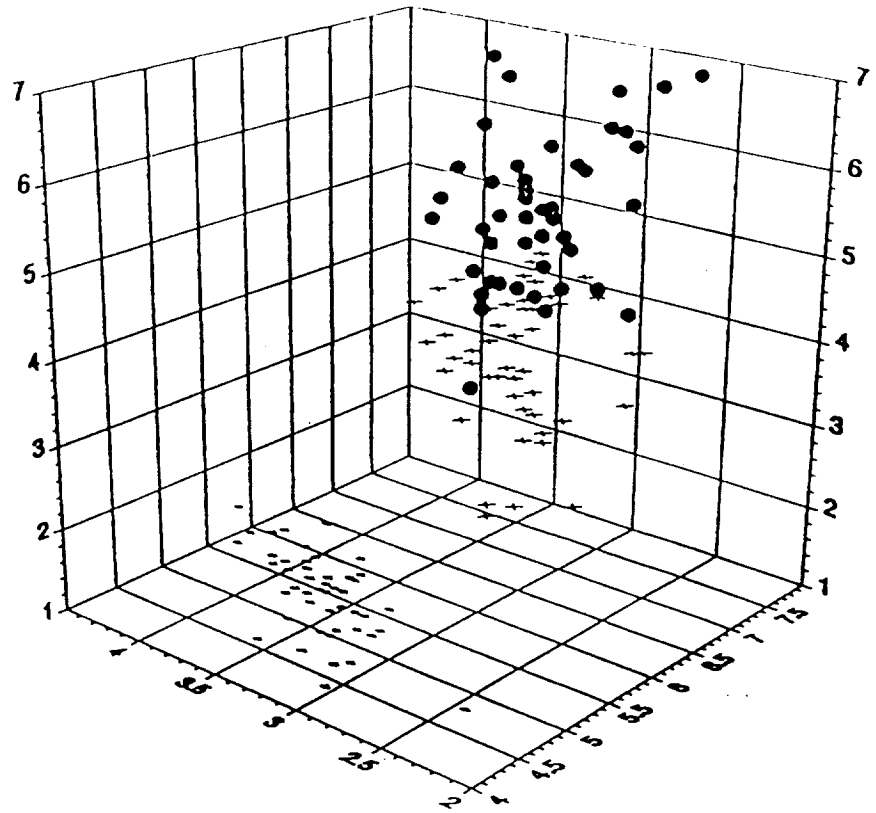


Figure 3.

CLUSTER No.	CLUSTER CENTROID VECTOR xi, i=1,2,3,4			
1	5.95	2.76	4.33	1.34
2	6.72	3.05	5.66	2.14
3	5.00	3.42	1.46	0.24

Figure 4a.

		ACTUAL 1	2	3
OUTPUT	1	50	7	
	2		43	
	3			50

Figure 4b.

Fingerprint Data

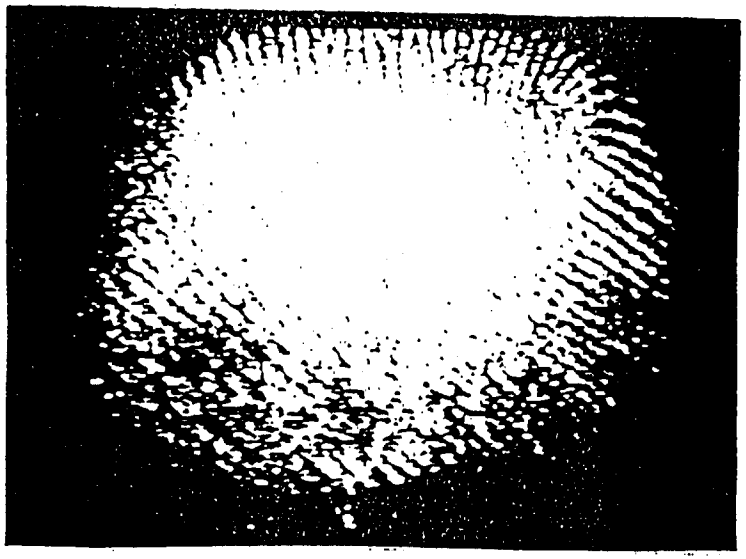


Figure 5a.

Figure 5b.

CLUSTER No.	CLUSTER CENTROID VECTOR X_i, i = 1,2 ... 7						
	1	2	3	4	5	6	7
1	199.48	15254.10	75.25	14.99	488.95	-822.35	278.50
2	262.52	15956.27	22750.70	4500.20	23888973.00	530694.06	-11096280.00
3	538.03	803.625	57.02	34.38	427.83	112.10	881.09

Figure 6a.

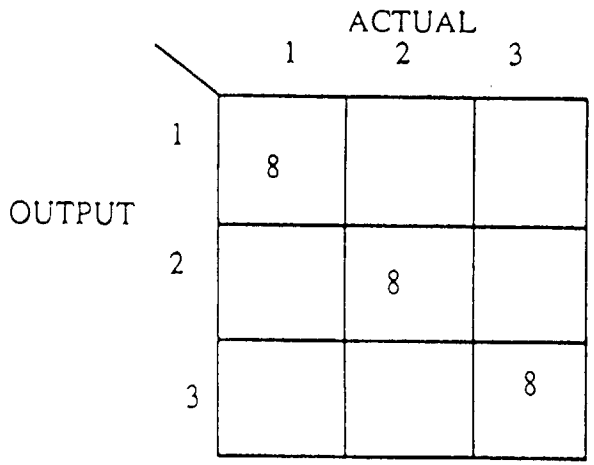


Figure 6b.

S11-63
ABS. ONLY
15041
P 3

Fuzzy Logic Path Planning System for Collision Avoidance by an Autonomous Rover Vehicle

Michael G. Murphy, Ph.D.
University of Houston-Downtown

N93-22217

The Space Exploration Initiative of the United States will make great demands upon NASA and its limited resources. One aspect of great importance will be providing for autonomous (unmanned) operation of vehicles and/or subsystems in space flight and surface exploration. An additional, complicating factor is that much of the need for autonomy of operation will take place under conditions of great uncertainty or ambiguity. This report addresses issues in developing an autonomous collision avoidance subsystem within a path planning system for application in a remote, hostile environment that does not lend itself well to remote manipulation by Earth-based telecommunications. A good focus is unmanned surface exploration of Mars. The uncertainties involved indicate that robust approaches such as fuzzy logic control are particularly appropriate. Four major issues addressed in this report are: avoidance of a fuzzy moving obstacle; backoff from a deadend in a static obstacle environment; fusion of sensor data to detect obstacles; and, options for adaptive learning in a path planning system. Previous work dealt with stationary obstacle scenarios. Examples of the need for collision avoidance by an autonomous rover vehicle on the surface of Mars with a moving obstacle would be: wind-blown debris, surface flow or anomalies due to subsurface disturbances, another vehicle, etc. The other issues of backoff, sensor fusion, and adaptive learning are important in the overall path planning system.

For true autonomy of operation, higher-level path planning is necessary to ensure integrity of the physical system, allow for conservative modification of guidance rules based on experience, and facilitate efficient backoff from deadend approaches. A consideration is to seek generalized features that encourage extension or adaptation of this path planning system to other environments (e.g., autonomous collision avoidance for space vehicles with respect to other space vehicles, space debris, etc.)

Using the simplest approach to a complicated problem, it is best not to try to project the exact path of a moving obstacle. Instead, fuzzy rules and a fuzzy inferencing mechanism are used to assess the likelihood of collision. The architecture for a fuzzy avoidance system for a moving fuzzy obstacle is addressed. In general, this will be a subsystem of a general path planning system for autonomous exploration with collision avoidance.

Sensor fusion, combining information based on more than one sensor operating simultaneously, promises to give a significant improvement in obstacle detection over the use of a single sensor source. The problem is to

have a computationally reasonable means of combining and interpreting sensor data from dissimilar sources.

There are several approaches to backoff from deadends in a static environment that is not fully mapped and where uncertainty of information is a regular element of the environment. One technique is based on reversing direction coupled with extending the critical distance for sensor processing and synthesis to avoid oscillatory travel patterns. Another approach is to store a modified world model that would map approximate information regarding the explored environment. It is likely that the first approach may have an advantage in the sense of a lesser degree of complexity. Other possibilities are storing a limited map of the explored region or blocking one or more sectors from being chosen until new data is available.

One of the most promising options for adaptive learning in control environments is the use of neural networks; e.g., to tune (adjust) the membership functions of fuzzy variables. A bigger problem is to develop an adaptive system that will operate on data being generated as the system performs and continually update parameters of the system to improve or maintain optimal (or near optimal) performance.

DRIVING A CAR WITH CUSTOM-DESIGNED FUZZY INFERENCING VLSI CHIPS AND BOARDS*

François G. Pin and Yutaka Watanabe
Center for Engineering Systems Advanced Research
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, TN 37831-6364

Invited Paper: To appear in the Proceedings of the 1992 International Joint Technology Workshop on Fuzzy Logic and Neural Networks, Houston, Texas, June 1–3, 1992.

* Research sponsored by the Engineering Research Program, Office of Basic Energy Sciences, of the U.S. Department of Energy, under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

DRIVING A CAR WITH CUSTOM-DESIGNED FUZZY INFERENCING VLSI CHIPS AND BOARDS

François G. Pin and Yutaka Watanabe
Center for Engineering Systems Advanced Research
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, TN 37831-6364

ABSTRACT

Vehicle control in a-priori unknown, unpredictable, and dynamic environments requires many calculational and reasoning schemes to operate on the basis of very imprecise, incomplete, or unreliable data. For such systems, in which all the uncertainties can not be engineered away, approximate reasoning may provide an alternative to the complexity and computational requirements of conventional uncertainty analysis and propagation techniques. Two types of computer boards including custom-designed VLSI chips have been developed to add a fuzzy inferencing capability to real-time control systems. All inferencing rules on a chip are processed in parallel, allowing execution of the entire rule base in about 30 μsec (i.e., at rates much faster than sensor data acquisition), and therefore, making control of "reflex-type" of motions envisionable. The use of these boards and the approach using superposition of elemental sensor-based behaviors for the development of qualitative reasoning schemes emulating human-like navigation in a-priori unknown environments are first discussed. We then describe how the human-like navigation scheme implemented on one of the qualitative inferencing boards was installed on a test-bed platform to investigate two control modes for driving a car in a-priori unknown environments on the basis of sparse and imprecise sensor data. In the first mode, the car navigates fully autonomously, while in the second mode, the system acts as a driver's aid providing the driver with linguistic (fuzzy) commands to turn left or right and speed up or slow down depending on the obstacles perceived by the sensors. Experiments with both modes of control are described in which the system uses only three acoustic range (sonar) sensor channels to perceive the environment. Simulation results as well as indoors and outdoors experiments are presented and discussed to illustrate the feasibility and robustness of autonomous navigation and/or safety enhancing driver's aid using the new fuzzy inferencing hardware system and some human-like reasoning schemes which may include as little as six elemental behaviors embodied in fourteen qualitative rules.

1. INTRODUCTION

One of the greatest challenges in developing motion planning and control systems for vehicles operating in a-priori unknown, unpredictable, and dynamic environments is to design the methods for handling the many imprecisions, inaccuracies, and uncertainties that are present and pervasive in the perception and reasoning modules. These imprecisions typically are caused by: (1) errors in the sensor data (current sensor systems are far from perfect) which lead to inaccuracies and uncertainties in the representation of the environment, the robot's estimated position, etc., (2) imprecisions or lack of knowledge in our understanding of the system, i.e., we are unable to generate complete and exact (crisp) mathematical and/or numerical descriptions of all the phenomena contributing to the environment's and/or the system's behavior, and (3) approximations and imprecisions in the information processing schemes (e.g., discretization, numerical truncation, convergence thresholds, etc.) that are used to build environmental models and to generate decisions or control output signals. In such systems, for which it is not currently feasible to fully engineer all the uncertainties away from the perception subsystems, approximate (or "qualitative") reasoning may provide an alternative to the complexity and prohibitive computational requirements of conventional uncertainty analysis and propagation techniques.

In cooperation with MCNC, Inc. and the University of North Carolina, two types of VME-bus-compatible computer boards including custom-designed VLSI chips have been developed to add a qualitative reasoning capability to real-time control systems [1],[2],[3],[4]. The methodologies embodied on the VLSI hardware utilize the Fuzzy Set Theoretic operations [5],[6],[7],[8] to implement a production rule type of inferencing on input and output variables that can directly be specified as qualitative variables through membership functions. All rules on a chip are processed in parallel, allowing full execution of the rule base in about 30 μsec. This extremely short time of operation makes real-time reasoning feasible at speeds much faster than typical sensor data acquisition rates, therefore, making envisionable the control of very fast processes such as sensor-based "reflex-type" motions.

The basic operation of these boards and a formalism merging the fuzzy and behaviorist theories for the development of qualitative reasoning schemes emulating human-like navigation have been discussed in [4]. The approach using superposition of elemental sensor-based fuzzy behaviors has been shown to allow easy development and testing of the inferencing rule base, while providing for progressive addition of behaviors to resolve situations of increasing complexity. This fuzzy behavior formalism has been used to demonstrate the feasibility of autonomous robot navigation in a-priori unknown environments on the basis of sparse and very imprecise sensor data [9]. For these feasibility experiments, a small omnidirectional robotic platform prototype [10] equipped with a ring of acoustic range finders (sonars) was used in a laboratory environment. In this paper, we present further developments on the feasibility of autonomous navigation in a-priori unknown environments using approximate reasoning and very inaccurate sensor data. Section 2 describes how the "human-like reasoning" navigation rule base of the small omnidirectional platform was extended to allow for the kinematic limitations of a car (non-holonomic and steering constraints) and was applied to the autonomous navigation of a car in laboratory simulations. The operation of the system in driver's aid mode is also described in this section. The entire perception and fuzzy inferencing system was then positioned on a car and Section 3 presents the operation of the system in outdoor environments. The last section discusses the results of these feasibility studies and presents the concluding remarks.

2. FUZZY BEHAVIORS FOR CAR DRIVING

In the experiments with the small omnidirectional platform, fuzzy rule bases embodying six basic navigation behaviors [9] were developed to control the turn rate (TR) and the translational speed (TS) of the platform as a function of the goal direction (GD) and obstacle proximity (OP). The single chip board [1] was used which allows inferencing on four input variables to produce two output variables. The four input variables were selected as the goal direction and obstacle proximity in sectors at the left, center, and right of the travel direction. As shown on Fig. 1, each sector encompasses five sonars. In each sector,

the distance returns from each of the five sonars are weighed by a factor proportional to their firing direction, and the smallest value is utilized to indicate obstacle proximity within the sector. Effectively, this corresponds to giving the platform the equivalent of three "very wide and blurry" eyes. The navigation goal can be specified in the current system as a goal point or as a heading to be maintained. When the goal is a point, the odometry system updates the position of the robot at each loop rate and calculates the relative direction to the goal point as input to the inferencing system. When the goal is a heading, a compass is used to directly provide the relative goal direction as the difference between the platform current heading and the goal heading. As explained in [4], membership functions representing the levels of uncertainty with which the values were obtained are applied to the four input values. Very robust navigation characteristics were obtained in the laboratory experiments using these very sparse and imprecise sensor data (purposefully selected as such to emphasize the feasibility demonstration), and as little as fourteen fuzzy rules representing the six basic behaviors controlling the platform's turning rate and speed (see [4] or [9]): GD → TR, GD → TS,OP → TS, "far" OP → TR, "near" OP → TR, "very near" OP → TR.

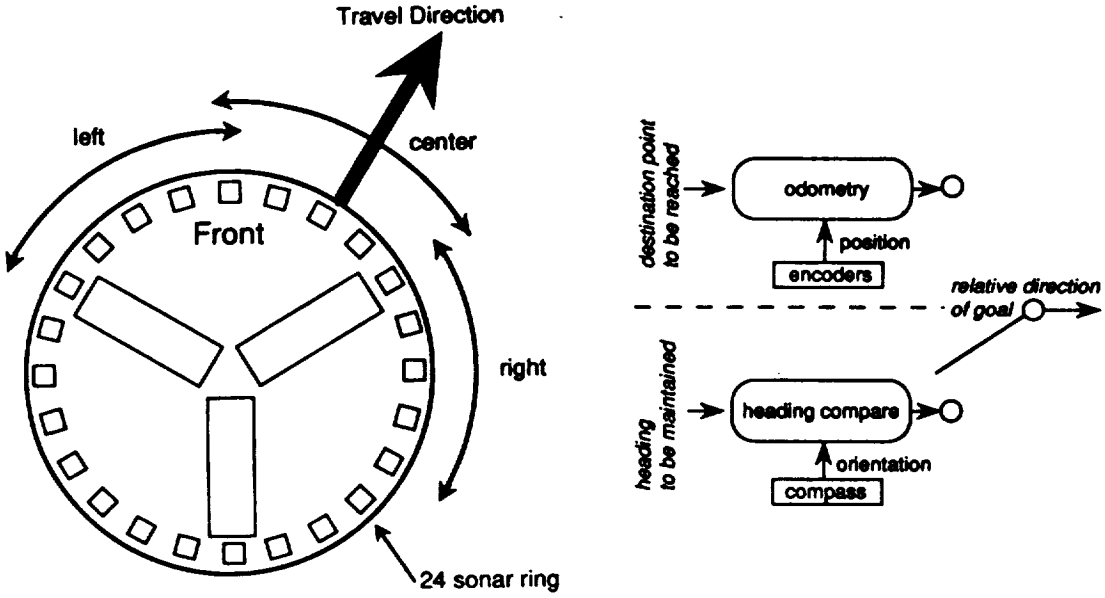


Fig. 1. Schematic of the three 5-sonar sectors providing obstacle proximity input data, and the two methods for calculating the goal direction depending on the mode of goal specification.

2.1 APPLICATION TO CAR DRIVING

One of the expected strengths of our proposed "Fuzzy-Behaviorist" approach using "human-like" behaviors is that the *linguistic logic* embodied in the behaviors should be invariant among systems of similar characteristics. In other words, for robots with similar perceptive and motion capabilities, the linguistic expression of given behaviors, and therefore their representation in the fuzzy framework, should be the same for compatible input and output. For example, a "goal tracking" behavior connecting the perceived goal direction to a rate of turn [e.g. IF (goal is to the right) THEN (apply increment of turn to the right)] should be invariant for any robot which has a means to perceive the goal direction and to perform the required turn. Using this property (and realizing that the rate of turn of a car is proportional to the steering angle of the wheels), all navigation behaviors developed for the laboratory omnidirectional platform appear directly applicable to the driving of a car of similar size, except for those behaviors which require a rate of turn too large for the car to perform because of its limited steering angle. The "very near" OP $\rightarrow$ TC behavior, which requires the platform to perform high rates of turn (using its omnidirectional capability) when obstacles are detected at dangerously close ("very near") distances, is the only behavior which therefore could not be considered invariant from the platform to the car.

As a demonstration of the transportability of invariant behaviors from one system to another, the same behaviors (except for the "very near" OP $\rightarrow$ TC behavior) and the very same fuzzy rules that were utilized for the omnidirectional platform were used to implement the autonomous control of a car on the basis of the same "three wide blurry eyes" and goal direction input. Figure 2 shows a simulation example of such a navigation in which the car has to reach a goal (in the upper right section) and then return to its start position (in the lower left section). Note that the out and return paths are different. Also note that a large maximum steering angle has been selected for the car in this simulation to allow very small radii of turn (e.g. see the sharp turn in the upper right section) and therefore prevent situations with "very near" obstacles.

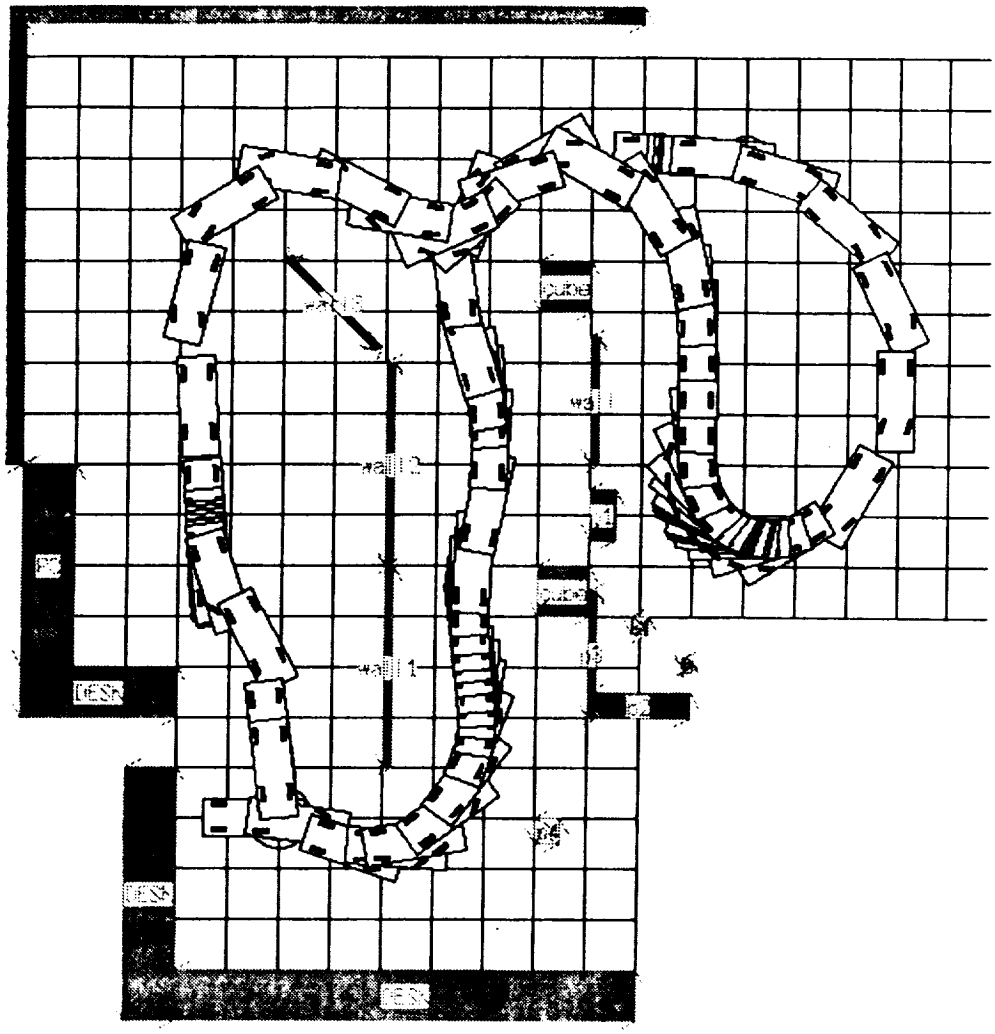


Fig. 2. Simulation example of the autonomous navigation of a car using three "wide" sonars and the same invariant navigation behaviors than for the omnidirectional platform.

2.2 ADDITION OF A MANEUVERING BEHAVIOR

To complete the navigation rule base for the driving of the car, a behavior has to be included to handle the situations where "very near" obstacles are detected. Another strength of our proposed "Fuzzy-Behaviorist" approach is its capability for superposition of elemental behaviors along a "subsumption-type" of architecture (e.g. see [11]), allowing for progressive addition of behaviors to the system to resolve situations of increasing complexity. Since the five other basic behaviors assure collision-free navigation amidst "far" and "near" frontal obstacles, the situations involving "very near" obstacles would occur when the car does not have enough space to complete a turn away from obstacles because of its limited steering angle and radius of turn, and thus would require

some maneuvers using reverse gear. By observing human reactions to such stimuli, a "human-like" response was created which can be expressed as follows: IF (obstacle is "very near" on right (left)) THEN (steer right (left)) AND (back up). This response was further divided into a steer control behavior: "very near" OP $\rightarrow$ TR, and a speed control (back up) behavior: "very near" OP $\rightarrow$ TS, to respect our approach's requirement for independence of behaviors [4]. Note that this latter behavior is intrinsically "human-like" since it implements a human reaction which implicitly utilizes the inertia present in the car in order to produce the desired effect.

Figure 3 displays sample results showing several maneuvers generated by the two "very near" OP behaviors in a simulation of the autonomous navigation of a car using the three "wide sonar" eyes as a perception system. Note that in this simulation, the "front" of the car, where the three wide-sonar perception eyes are mounted, corresponds to the axle with non-steering wheels, while the axle with the steering wheels is to the "back" of the car. This was done to closely duplicate the configuration utilized in the outdoor experiments in which the perception system was positioned on the back trunk of the vehicle, as explained in the next section.

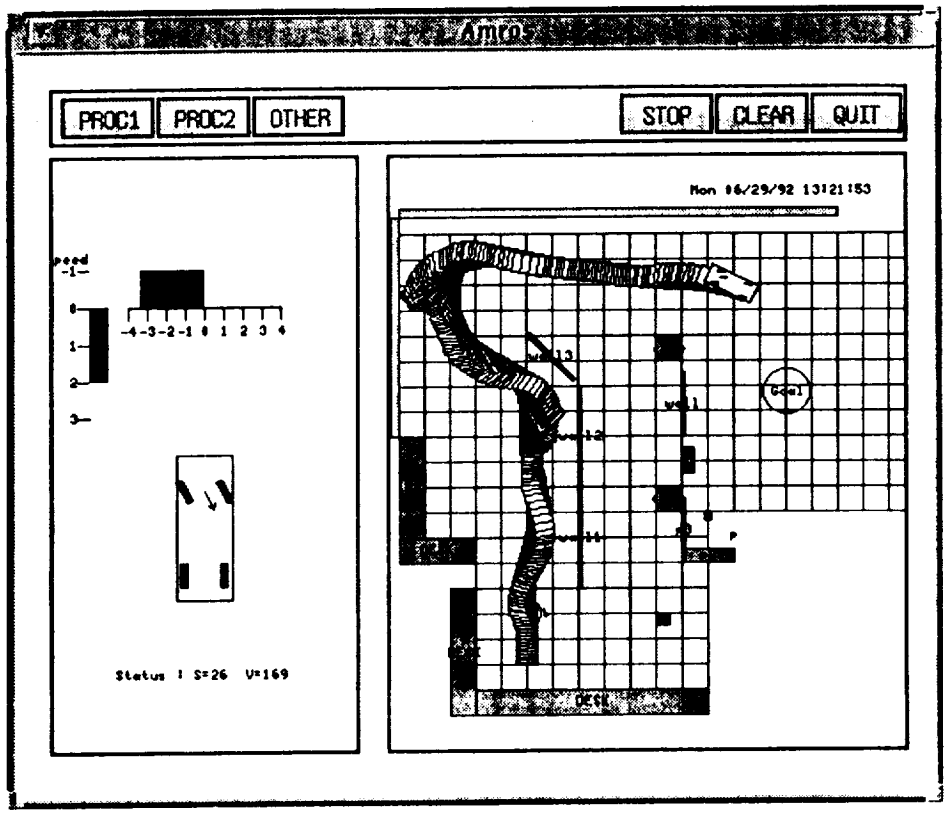


Fig. 3. Simulation example of the autonomous navigation of a car using three "wide" sonars and a maneuvering behavior to overcome the limited radius of turn.

2.3 ADDITION OF A DRIVER'S AID MODE

Once the development of the fuzzy rule base for autonomous navigation was completed and had been tested in various simulated environments, the system was investigated for use as a "driver's aid." In the simulation system, the output of the fuzzy inferencing was conveniently displayed on the screen, as is shown on the left-hand side of Fig. 3. The horizontal and vertical bar scales respectively represent the steering and speed commands which are calculated by the fuzzy inferencing and, in the autonomous navigation mode, are sent to the controls of the vehicle emulator. The schematic of the car below the bars shows the steering of the wheels implemented by the controller. Recall that the car moves "backwards" so that to perform a turn to the right, the wheels have to be steered to the left. In the driver's aid mode, the very same rule base, commands and displays are used to guide the operator in driving the car. In the simulations, the driver uses the keyboard arrow keys to add or subtract increments of speed or steering. In the implementation of the system on one of the company's cars, the driver conventionally uses the gas and brake pedals and the steering wheel to implement the commands.

For the testing and verification experiments, the driver was prohibited from seeing the environment while driving. This was done by covering the vehicle motion display part of the screen in the graphic simulations, and in the outdoor experiments by positioning the sensing platform on the rear trunk of the car and having the operator drive backwards while looking at the portable computer screen located on his/her lap. From this came the requirement for the "backwards" driving in the simulations and the corresponding reverse of the commands. Note that the commands are not displayed to the operator as crisp control values, but as bars of variable lengths over the generic speed and steering scales, effectively providing only the direction of the command (left or right, forward or back) and the *relative strength* (i.e., more steering, faster, slower, etc.) which the driver should apply on the controls between the maximum steering and speed values. It was interesting to observe each operator develop his/her own interpretation of and response to these relative commands, leading to quite different routes and maneuvering situations for the same start and goal positions. From the system's development point of view, this inclusion of the

human in the control chain effectively consisted in including a source of unpredictable noise and delays in the actuation system. The successful operation of the rule base in this mode of driving provided a very stringent robustness test of the inferencing rule base.

3. OUTDOOR DRIVING EXPERIMENTS

Figure 4 shows the experimental set up for the outdoors experiments. The wheels of the omnidirectional platform which was used in previous laboratory experiments [9],[10], have been removed, and its upper plate supporting the sensors, batteries, and computers has been mounted on the trunk of one of the company's cars. Since the car was not equipped with wheel encoders, odometry could not be used and an electric compass provided the goal direction input with the navigation goal specified as a heading (e.g. North). To take into account the relative width of the real car with respect to that used in the simulations (of the same 2 foot width than the omnidirectional platform), the x axis of all membership functions involving distance were linearly scaled by a factor of three. The same input, rules, and behaviors developed in the simulation studies were used in these outdoor experiments. The output of the fuzzy inferencing was sent to a portable computer located in the cabin. The steering and speed commands were displayed on the computer screen using the same format than shown in Fig. 3 for the simulations. Since the car is not currently equipped with automated actuators on the steering column or the speed control system, these experiments were performed using the driver's aid mode of operation. The driver sat in a normal position in the car and was prohibited to look at the environment by having to constantly watch the commands on the computer screen located on the floor in the front compartment.

The type of environments in which the tests were preformed were the diversely occupied parking lots of ORNL, as can be seen in the background of Fig. 4. In this type of non-engineered environments, the car was very successfully driven in the "blind" driver's aid mode. Our future plans include the integration of encoders and servo controls on the wheels, steering, accelerator, and braking systems of the car to experiment with, test, and demonstrate the autonomous control mode in outdoors environment.

Fig. 4. Experimental set up during the outdoor experiments with driver's aid mode in one of the ORNL parking lots.

4. CONCLUSION

VLSI fuzzy inferencing chips and a "fuzzy behaviorist" approach have been used to demonstrate the feasibility of driving a car under sensor-based autonomous navigation or driver's aid mode using only sparse data from very inaccurate sensors. The "subsumption-type" formalism proposed for the development of fuzzy behavior-based systems has been found to allow easy development of the behaviors and progressive augmentation of the fuzzy rule base to deal with situations of increasing complexity, such as in the example treated here of a need for maneuvering due to the car's limited radius of turn. Additionally, the framework has been shown to allow the same behaviors, rules, and inferencing code to be used for systems with similar perceptive and kinematic characteristics, therefore greatly enhancing code transportability among

robots and systems. As shown in the driver's aid feasibility study, the straightforward "linguistic" interfacing capability of the fuzzy behavior-based system is also of great appeal for telerobotics and man-machine decisional systems. Our ongoing activities are focusing on the use of a recently developed multi-chip fuzzy inferencing board, in conjunction with additional on-board image sensors, to increase the car's autonomous navigation capabilities with behaviors such as road following or highway driving, and correspondingly augment the safety enhancing driver's aid system for a variety of outdoor environments.

5. REFERENCES

[1] H. Watanabe, J. R. Symon, W. D. Dettloff, and K. E. Yount, "VLSI Fuzzy Chip and Inference Accelerator Board Systems," in Proceedings of the International Symposium on Multivalued Logic, Victoria, Canada, May 1991, pp. 120–127.

[2] J. R. Symon and H. Watanabe, "Single Board System for Fuzzy Inference," in Proceedings of the Workshop on Software Tools for Distributed Intelligent Control Systems (September 1990), pp. 253–261.

[3] H. Watanabe, W. Dettloff, and E. Yount, "A VLSI Fuzzy Logic Controller with Reconfigurable, Cascadable Architecture," *IEEE J. of Solid State Circuits* **25**(2), 376–382 (1990).

[4] F. G. Pin, H. Watanabe, J. R. Symon, and R. S. Pattay, "Using Custom-Designed VLSI Fuzzy Inferencing Chips for the Autonomous Navigation of a Mobile Robot" in Proceedings of IROS 92, the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems, Raleigh, North Carolina, July 7–10, 1992.

[5] L. A. Zadeh, "Fuzzy Set," *Information and Control* **8**, 338–353 (1965).

[6] L. A. Zadeh, "Outline of a New Approach to the Analysis of Complex Systems and Decision-Making Approach," *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-3**(1), 28–45 (January 1973).

[7] L. A. Zadeh, "Fuzzy Logic," *IEEE Computer* **21**(4), 83–93 (April 1988).

[8] "Fuzzy Sets and Their Applications to Cognitive and Decision Processes," eds. L. A. Zadeh, K. S. Fu, K. Tanaka, and M. Shinmra, Academic Press, Inc., New York (1975).

[9] F. G. Pin, H. Watanabe, J. R. Symon, and R. S. Pattay, "Autonomous Navigation of a Mobile Robot Using Custom-Designed Qualitative Reasoning VLSI Chips and Boards," in Proceedings of the 1992 IEEE International Conference on Robotics and Automation, Nice, France, May 10–15, 1992, pp. 123–128.

[10] S. M. Killough and F. G. Pin, "Design of an Omnidirectional and Holonomic Wheeled Platform Prototype," in Proceedings of the 1992 IEEE International Conference on Robotics and Automation, May 10–15, 1992, Nice, France, pp. 84–90.

[11] R. A. Brooks, "Elephants Don't Play Chess," *Robotics and Autonomous Systems* 6(1–2), 3–15 (1990).

Autonomous Vehicle Motion Control, Approximate Maps, and Fuzzy Logic

Enrique H. Ruspini
Artificial Intelligence Center
SRI International
Menlo Park, California 94025

We present progress on research on the control of actions of autonomous mobile agents using fuzzy logic. The innovations described encompass theoretical and applied developments.

At the theoretical level, we present results of research leading to the combined utilization of conventional artificial planning techniques with fuzzy logic approaches for the control of local motion and perception actions. We examine also novel formulations of dynamic programming approaches to optimal control in the context of the analysis of approximate models of the real world. We review also a new approach to goal conflict resolution that does not require specification of numerical values representing relative goal importance.

Applied developments include the introduction of the notion of approximate map. We propose a fuzzy relational database structure for the representation of vague and imprecise information about the robot's environment. We discuss also the central notions of control point and control structure and present a short video of the application of these techniques in the platform provided by SRI's Autonomous Mobile Vehicle.

A Fuzzy Logic Controller for an Autonomous Mobile Robot

John Yen and Nathan Pfluger
Computer Science Department
Texas A&M University
College Station, TX, 77840

The ability of a mobile robot system to plan and move intelligently in a dynamic system is needed if robots are to be useful in areas other than controlled environments. An example of a use for this system is to control an autonomous mobile robot in a space station, or other isolated area where it is hard or impossible for human life to exist for long periods of time (e.g. Mars). The system would allow the robot to be programmed to carry out the duties normally accomplished by a human being. Some of the duties that could be accomplished include operating instruments, transporting objects and maintenance of the environment.

There are many limitations of current approaches. Methods based on potential fields and stimulus--response paradigms have problems finding paths, even when they exist. The standard graph decomposition method always gives a path, but requires complete knowledge of the environment, and gives a path that is not easily followed. Finally, there are no approaches that have adequately addressed the problems involved with interleaving task planning, path generation and path execution.

The important issues that any realistic robot path planning system must address are:

1. Plan several tasks concurrently.
2. Deal with a dynamic environment.
3. Deal with the problems of incomplete and/or inaccurate knowledge about the environment.
4. Work with the hindrance of limited sensing capability.

The main focus of our early work has been on developing a fuzzy controller that takes a path and adapts it to a given environment. The robot only uses information gathered from the sensors, but retains the ability to avoid dynamically placed obstacles near and along the path.

By using fuzzy logic, our project has been able to address the limitations of existing approaches. Our controller is able to use graph-decomposition methods in a dynamic environment. Fuzzy logic techniques, in general, allow experts to express their planning and control rules in natural-language form. This makes the system easier to develop and more compact than standard logic systems.

OVERVIEW OF ALGORITHM

Our fuzzy logic controller is based on the following algorithm:

1. Determine the Desired Direction of Travel.
2. Determine the Allowed Direction of Travel.
3. Combine the Desired and Allowed Directions in order to determine a direction that is both desired and allowed.

The Desired direction of travel is determined by projecting ahead to a point along the path that is closer to the goal. This gives a local direction of travel for the robot and helps to avoid obstacles.

The Allowed direction is found by combining a set of sensors that give the distance to the nearest obstacle along a set of directions, say 0, 45, 90, -45 and -90 degrees from the robots current heading.

The process of combining the Desired and Allowed directions uses the fuzzy operator 'and' to obtain a fuzzy command that corresponds to the desired control command. We then use defuzzification to obtain a crisp command.

N93-22221

Intelligent Neuroprocessors for In–Situ Launch Vehicle Propulsion Systems Health Management

S. Gulati,. R. Tawel, and A.P. Thakoor

Center for Space Microelectronics Technology
Jet Propulsion Laboratory
Pasadena, CA 91109

Efficacy of existing on-board propulsion systems HMS are severely impacted by computational limitations (e.g., low sampling rates); paradigmatic limitations (e.g., low-fidelity logic/parameter redlining only, false alarms due to noisy/corrupted sensor signatures, preprogrammed diagnostics only); and telemetry bandwidth limitations on space/ground interactions. Ultra-compact/light, adaptive neural networks with massively parallel,asynchronous, fast reconfigurable and fault-tolerant information processing properties have already demonstrated significant potential for inflight diagnostic analyses and resource allocation with reduced ground dependence. In particular, they can automatically exploit correlation effects across multiple sensor streams (plume analyzer, flow meters,vibration detectors, etc.) so as to detect anomaly signatures that cannot be determined from the exploitation of single sensor. Furthermore, neural networks have already demonstrated the potential for impacting real-time fault recovery in vehicle subsystems by adaptively regulating combustion mixture/power subsystems and optimizing resource utilization under degraded conditions. In this paper we present a class of high-performance neuroprocessors, developed at JPL, that have demonstrated potential for next-generation HMS for a family of space transportation vehicles envisioned for the next few decades, including HLLV, NLS, and space shuttle. Of fundamental interest are intelligent neuroprocessors for real-time plume analysis, optimizing combustion mixture-ratio and feedback to hydraulic, pneumatic control systems.This class includes concurrently asynchronous, reprogrammable, nonvolatile, analog neural processors with high speed, high bandwidth electronic/optical I/O interfaces, with special emphasis on NASA's unique requirements in terms of performance, reliability, ultra-high density, ultra-compactness, ultra-light weight devices, radiation hardened devices, power stringency and long life terms.

Initiated with the original goal of developing content addressable, high density,nonvolatile memories based on mathematical models of neural networks, the research program at NASA's Jet Propulsion Laboratory (JPL) in Pasadena, CA has evolved over the years into a major research and technology demonstration activity in hardware implementations of highly parallel feedback and feedforward "neuroprocessing" architectures, with

computing speeds in excess of 10 analog operations per second and unique capabilities not captured by conventional digital and AI technologies. Particular emphasis is placed on development of fully parallel, cascadable "building blocks", such as fully programmable synaptic interconnection arrays and nonlinear analog neuron arrays based on custom-VLSI technology. Building blocks designed to date include programmable 32 X 32 binary and gray level (with 5, 10, and 10 bit resolution) synaptic arrays, using floating gate and capacitor refresh technology. The evolution of neuron development has included several implementations ranging from boards of discrete neurons based on off-the-shelf components, to multi- neuron, cascadable VLSI chips. Some of the neural chips that have been designed include 64-neuron fixed gain and variable gain chips and 64-neuron winner-take-all neuron chip. Current efforts are focusing on wafer level integration, thru-wafer contact technology and 3-D Z-plane interconnection technology (stacked VLSI/ULSI wafers with metal diffused through the thickness of the wafer to provide highly directional, dense interconnectivity between adjacent wafer surfaces).

The development of application-specific neuroprocessors and assessment of their effectiveness on selected applications, which are not easily tackled by conventional computing techniques, at JPL has progressed hand-in-hand with the development of the building block hardware devices. Applications range from fault-addressable CAMs to several classification and optimization problems. Optimization problems such as arbitrary many-to-many (concentrator) assignment problem are handled particularly well by neural networks. JPL developed a new breakthrough concept for hardware implementation of a neuroprocessor for high speed solutions to dynamic assignment problems, e.g., resource allocation, etc. Considerable attention has also focused on evaluation of hardware systems with feedforward architectures. As a first step towards fully parallel hardware with capabilities of supervised and unsupervised learning, JPL demonstrated learning "off-chip", which involves generation of synaptic weights using a digital computer. The weights are then loaded in the hardware.

N93-22222

SYNTHESIS OF NONLINEAR CONTROL STRATEGIES FROM FUZZY LOGIC CONTROL ALGORITHMS

Reza Langari
Department of Mechanical Engineering
Texas A&M University
College Station, TX 7783-3123

Abstract Fuzzy control has been recognized as an alternative to conventional control techniques in situations where the plant model is not sufficiently well known to warrant the application of conventional control techniques. Precisely what fuzzy control does and how it does what it does is not quite clear, however. This paper deals with this important issue and in particular shows how a given fuzzy control scheme can resolve into a nonlinear control law and that in those situations the success of fuzzy control hinges on its ability to compensate for nonlinearities in plant dynamics.

INTRODUCTION

Fuzzy logic control has been recognized as an alternative to conventional control techniques(primarily PID, or switching type control) for application in industrial process control and manufacturing automation(Sugeno 1985). More often than not, however, empirical observation provides the only means to a comparative study of performance of fuzzy controllers in relation to their conventional counterparts. While this fact is recognized and even appreciated by practitioners in the process control area, precisely what a fuzzy controller does, that is from an analytical standpoint, and how it does what it does is still of interest.

In order to investigate this issue, we will consider the notion of *parametrized* fuzzy sets and discuss its implication in analysis of fuzzy control algorithms. This idea, it turns out(Langari and Tomizuka 1990, Langari 1990, Langari 1992) gives rise to a framework for analysis and synthesis of *nonlinear* control strategies that emerge quite naturally from an initial statement of a given control strategy as a fuzzy linguistic control algorithm.

In this article, we will use this framework to explain how a given fuzzy control strategy deals with process nonlinearities that conventional controllers, for instance PID, generally do not. In particular, we apply this framework to the problem of control synthesis in a typical situation where asymmetric response characteristics of the process precludes, or severely encumbers the application of conventional(linear) control theory. We further show how in this situation an appropriately designed fuzzy controller overcomes this difficulty and by in effect compensating for the underlying nonlinearities produces superior behavior.

We start with an overview of fuzzy control

FUZZY CONTROL SYSTEMS

The typical architecture of a fuzzy control systems in shown in Figure 1. As a rule based control strategy, fuzzy linguistic control is based on explicit representation of knowledge of operation of the process as condition action rules of the form

$$R_{j,j}: \text{ if } e(t) \text{ is } A_j \text{ and } de(t) \text{ is } B_l \text{ then } u(t) \text{ is } C_{j,l}$$

where $e(t)$ denotes the instantaneous value of the

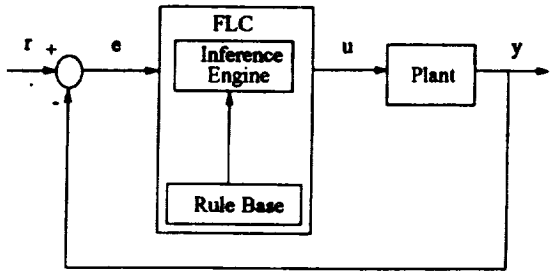


Figure 1 Architecture of a Fuzzy Logic Control System

process error at time t and $de(t)$ is short for $\mathcal{D}(e;t)$, which stands for $\frac{de}{dt}$ or $\int e\,d\tau$. Further, $\tilde{A}_j$, $\tilde{B}_l$, and $\tilde{C}_{j,l}$ belong to collections $\tilde{\mathcal{A}}$, $\tilde{\mathcal{B}}$, and $\tilde{\mathcal{C}}$ of fuzzy subsets defined over the domains of definition of the relevant variables, that is, E, DE, and U respectively and $R_{j,l}$ denotes the j,l^{th} rule in the rule set $\mathbf{R}$. In particular $R_{j,l}$ may be

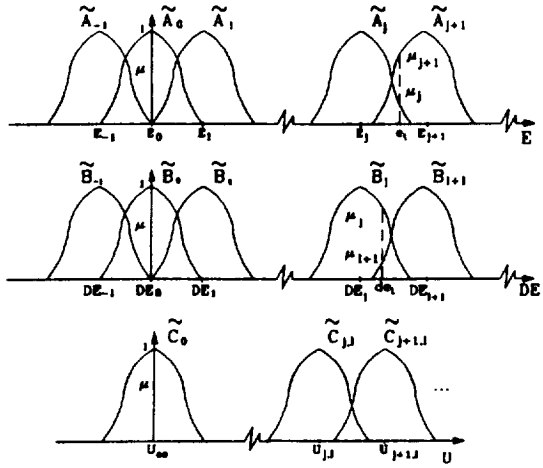


Figure 2. Fuzzy partitioning of the domains of definition.

be viewed as associating elements $\tilde{A}_j$ of $\tilde{\mathcal{A}}$ and $\tilde{B}_l$ of $\tilde{\mathcal{B}}$ with element $\tilde{C}_{j,l}$ of $\tilde{\mathcal{C}}$, thereby forming a *fuzzy relation*[1] $\tilde{R}_{j,l}$ over the Cartesian product space, $E \times DE \times U$. From this standpoint, the given fuzzy control algorithm in effect amounts to a disjunction of such associations, as in $\tilde{\mathcal{R}} = \bigvee_{j,l} \tilde{R}_{j,l}$, which Mamdani and Assilian(1975) refer to as the *fuzzy relation matrix*.

Control Computation

Suppose, at some instance t, as shown in Figure 2, the error $e(t)$ has positive grades of membership, $\mu_{\tilde{A}_j}(e(t))$ and $\mu_{\tilde{A}_{j+1}}(e(t))$ to some pair $\tilde{A}_j$ and $\tilde{A}_{j+1}$ in $\tilde{\mathcal{A}}$. Similarly, suppose $de(t)$ belongs to some pair $\tilde{B}_l$ and $\tilde{B}_{l+1}$ in $\tilde{\mathcal{B}}$. At this instant, the following control rules apply

$R_{j,l}$: if $e(t)$ is A_j and $de(t)$ is B_l then $u(t)$ is $C_{j,l}$

$R_{j+1,l}$: if $e(t)$ is A_{j+1} and $de(t)$ is B_l then $u(t)$ is $C_{j+1,l}$

$R_{j+1,l+1}$: if $e(t)$ is A_{j+1} and $de(t)$ is B_{l+1} then $u(t)$ is $C_{j+1,l+1}$

$R_{j,l+1}$: if $e(t)$ is A_j and $de(t)$ is B_{l+1} then $u(t)$ is $C_{j,l+1}$

with each rule satisfied to some degree. The corresponding *truth value* is defined, for instance for the first rule, by

$$\mu_{j,l} = \min\left(\mu_{\tilde{A}_j}(e(t)), \mu_{\tilde{B}_l}(de(t))\right) \quad (1)$$

or, alternatively by

$$\mu_{j,l} = \mu_{\tilde{A}_j}(e(t)) \cdot \mu_{\tilde{B}_l}(de(t)) \quad (2)$$

The truth values of other rules in the above set are similarly defined.

Note that the *product* instead of *min* results in interactivity between the truth values of the components of the antecedent clause. This fact is essential to our analytic treatment(Langari and Tomizuka 1990.)

Now, representing the consequent clause of each $R_{j,l}$ rule, that is , by its single representative, or *defuzzified*, value that is $U_{j,l}$, defined as

$$U_{j,l} = \frac{\int u \mu_{\tilde{C}_{j,l}}(u)}{\int \mu_{\tilde{C}_{j,l}}(u)} \quad (3)$$

the control action, $u(t)$, is computed as:

$$u(t) = \frac{\sum_{j,l} \mu_{j,l} U_{j,l}}{\sum_{j,l} \mu_{j,l}} \quad (4)$$

where j and l range over the indices of all applicable rules. Note that this approach is based on a variation of the Centroid of Area(COA) defuzzification rule(Zimmermann 1991), but has improved analytical properties(Langari and Tomizuka 1990).

ANALYSIS OF FUZZY LOGIC CONTROL ALGORITHMS.

Consider the single input, single output fuzzy linguistic control system shown in Figure 1. Here we develop an analytic description of the control law in the form, $u = FLC(e, de)$.

[1]Note that the distinction in the notation used, that is $R_{j,l}$ vs. $\tilde{R}_{j,l}$ reflects the distinction between rules and associations.

Definitions and Assumptions

Let us denote the domains of definition of e, de, and u by E, DE, and U respectively. Then, as shown in Figure 2, collections $\tilde{\mathcal{A}} = \{\tilde{A}_j\}$, $\tilde{\mathcal{B}} = \{\tilde{B}_l\}$, and $\tilde{\mathcal{C}} = \{\tilde{C}_{j,l}\}$ of *unimodal, convex,* and *normal* fuzzy subsets(Dubois and Prade 1980) effectively partition E, DE, and U, respectively, as follows.

Each element $\tilde{A}_j$ of $\tilde{\mathcal{A}}$ is centered at some $E_j \in E$ and is further characterized by a pair $L_j(\cdot)$ and $R_j(\cdot)$ of left and right characteristic functions(cf. Appendix A). Similarly, each $\tilde{B}_l \in \tilde{\mathcal{B}}$ is centered at some $DE_l \in DE$ and is characterized by $L'_l(\cdot)$ and $R'_l(\cdot)$. Moreover, each element $\tilde{C}_{j,l}$ is represented by its *defuzzified* value, $U_{j,l}$.

We further place some constraints on $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{B}}$ as follows. First, we require that $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{B}}$ form *true fuzzy partitions* of E and DE respectively.

Assumption 1. *Let $\tilde{\mathcal{A}} = \{\tilde{A}_j\}$ (and $\tilde{\mathcal{B}} = \{\tilde{B}_l\}$) be collection(s) of fuzzy subsets defined over E (and DE.) Then, for each element $e \in E$*

$$\sum_j \mu_{\tilde{A}_j}(e) = 1 \qquad (5)$$

(A similar condition holds for $\tilde{\mathcal{B}}$.)

The interpretation of Assumption 1 is that, externally, fuzzy classification must be compatible with feature based classification in terms of classical sets, where each element is categorized under one and only one class. This assumption is crucial to the development of our results and in effect amounts to *objectification* of the control law.

A sufficient condition for Assumption 1 to hold is that the characteristic functions of $\tilde{A}_j$(and $\tilde{B}_l$) be linear[2]:

Assumption 2. *For each j, let $\tilde{A}_j \in \tilde{\mathcal{A}}$ be defined in terms of a pair $L_j(\cdot)$ and $R_j(\cdot)$ of left and right characteristic functions. Then*

$$R_j(e) = 1 - (e - E_j)/\beta_j \qquad (6)$$

$$L_j(e) = 1 - (E_j - e)/\alpha_j \qquad (7)$$

and given j, and $j+1$, the line segments defined by $R_j(\cdot)$ and $L_{j+1}(\cdot)$ intersect E precisely at E_j and E_{j+1} respectively. (A similar condition holds for $\tilde{\mathcal{B}}$.)

This assumption implies that, as shown in Figure 3,

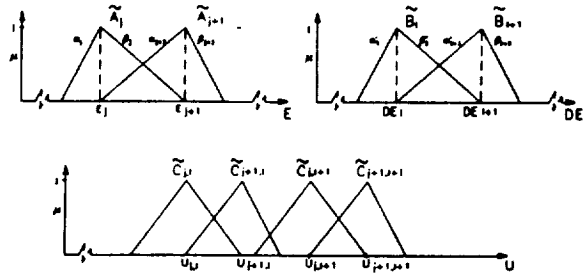


Figure 3. True Fuzzy Partitioning.

β_j and α_{j+1}, respectively representing the inverse of the slopes of the line segments defined by $R_j(\cdot)$ and $L_{j+1}(\cdot)$, must be equal. Let us denote this unique slope by m_j:

$$m_j := \frac{1}{\beta_j} = \frac{1}{\alpha_{j+1}}. \qquad (8)$$

Similarly, α'_{l+1} and β'_l, must also be equal; let us define $m'_l := \frac{1}{\alpha'_{l+1}} = \frac{1}{\beta'_l}$ to clearly indicate this fact as well. Consequently, we can define ΔE_j and ΔDE_l as follows:

$$\Delta E_j = E_{j+1} - E_j, \qquad (9)$$

$$\Delta DE_l = DE_{l+1} - DE_l. \qquad (10)$$

Let us also define $K_{j,l}$ and $K'_{j,l}$ as follows.

Definition 1. *Let us denote the functional relationship between $U_{j,l}$, E_j and E_{j+1} as:*

$$U_{j,l} = K_{j,l} E_j + K'_{j,l} DE_l. \qquad (11)$$

Then for each pair, j and l, $K_{j,l}$ and $K'_{j,l}$ are implicitly defined by (11).

[2] A generalization of this condition, where nonlinear characteristic functions are allowable, is possible. The present discussion, however, does not hinge on this fact. The interested reader may refer to Langari(1992).

Note that (11) simply relates $U_{j,l}$ to E_j and DE_l in a compact form and does not in any way constrain $U_{j,l}$.

We further define $\Delta K^{j}{}_{j+1,l}$...as follows:

$$\Delta K^{j}{}_{j+1,l} = K_{j+1,l} - K_{j,l}, \tag{12}$$

$$\Delta K^{l}{}_{j,l+1} = K_{j,l+1} - K_{j,l}, \tag{13}$$

$$\Delta K_{j+1,l+1} = K_{j+1,l+1} - K_{j,l}. \tag{14}$$

$$\Delta K'^{l}{}_{j,l+1} = K'_{j,l+1} - K'_{j,l}, \tag{15}$$

$$\Delta K'^{j}{}_{j+1,l} = K'_{j+1,l} - K'_{j,l}, \tag{16}$$

$$\Delta K'_{j+1,l+1} = K'_{j+1,l+1} - K'_{j,l}. \tag{17}$$

Now in view of the above assumptions the expression for $u(t)$, given by (4), resolves into

$$\begin{aligned} u(t) = {} & K_{j,l}e(t) + m_j\big(e(t)-E_j\big)\big[\Delta K^{l}{}_{j,l}E_{j+1} + \Delta K'^{j}{}_{j,l}DE_l\big] + \\ & K'_{j,l}de(t) + m'_l\big(de(t)-DE_l\big)\big[\Delta K^{l}{}_{j,l+1}E_j + \Delta K'^{j}{}_{j,l+1}DE_{l+1}\big] + \\ & m_j m'_l\big(e(t)-E_j\big)\big(de(t)-DE_l\big)\begin{bmatrix} \big(\Delta K_{j,l+1} - \Delta K^{l}{}_{j,l+1} - \Delta K^{j}{}_{j,l}\big)E_{j+1} + \\ \big(\Delta K'_{j,l+1} - \Delta K'^{l}{}_{j,l+1} - \Delta K'^{j}{}_{j,l}\big)DE_{l+1} + \\ \big(\Delta K'_{j,l+1} - \Delta K'^{l}{}_{j,l+1}\big)\Delta DE_{l+1} + \\ \big(\Delta K_{j,l+1} - \Delta K^{j}{}_{j,l}\big)\Delta E_{j+1} \end{bmatrix} \end{aligned} \tag{18}$$

The implication of the above formulation is that a given fuzzy logic control algorithm in effect amounts to a nonlinear control law that is further described in terms of three terms: one that is linear in each of $e(t)$ and $de(t)$, one that is linear in each of $e(t)-E_j$ and $de(t)-DE_l$, and finally one that is *bilinear* in the latter two terms. In effect the control law given by (18) reflects the capacity of fuzzy logic control to interpolate across the situations where individual control rules are directly applicable. We will see next how this capacity can be used to develop a control strategy that deals effectively with nonlinearities that commonly occur in process control.

APPLICATION

Let us consider the dynamic system:

$$\begin{aligned} \dot{x}_1 &= a_1x_1 + a_2x_2 + bu, \\ \dot{x}_2 &= x_1, \\ y &= x_2 - x_1, \end{aligned} \tag{19}$$

which reflects the behavior of a rather broad class of industrial processes; it is a relatively low order model, and has the somewhat dubious distinction of being *non-minimum phase*. The parameters, a_1, and a_2 are given by

$$a_1 = a_{10} + \delta a_1, \tag{20}$$

$$a_2 = a_{20} + \delta a_2, \tag{21}$$

where δa_1 and δa_2 reflect the variations in the plant parameters.

Suppose now, as it is commonly done in practice, we knew the process model and were to design a simple *proportional plus integral* control law:

$$u = k_p e + k_i \int_0^t e\, d\tau, \tag{22}$$

perhaps based on nominal values of the plant parameters, a_{10}, and a_{20}, as follows.

The plant and controller transfer functions are given by:

$$G_p(s) = \frac{1-s}{s^2 + a_{10}s + a_{20}}, \tag{23}$$

$$G_c(s) = \frac{k_c(s+\gamma)}{s}, \tag{24}$$

where $\gamma > 0$, $k_c = k_p$ is same as the proportional control gain, and $k_i = \gamma k_c$ is the equivalent integral gain.

Now, assuming that the closed loop system will behave as a dominantly second order system, the closed loop characteristic equation is given by

$$A(s) = (s+p)\big(s^2 + 2\xi\omega\, s + \omega^2\big), \tag{25}$$

where p is assumed large, we can use any number of ways of selecting ξ and ω— and thus k_c and γ — (Franklin, Powell, and Emami-Naeini 1991). For instance, we can simply pre-select γ and then choose ξ for desired response pattern and thus determine the gain k_c.

In practice, however, variations in the parameters of the plant, that is δa_1 and δa_2, affect the behavior of the process, and as a result the desired response is not reproduced as predicted. For instance, let us

suppose that these variations are function of the process error[3], e:

$$\delta a_1 = -\alpha \mathrm{sgn}(e), \qquad (26)$$

$$\delta a_2 = -\alpha \mathrm{sgn}(e), \qquad (27)$$

where $\alpha > 0$.

This situation happens in arc welding, for instance, where active heating and only passive cooling is available(Langari and Tomizuka 1988). A consequence of this change is that a fixed set of gains will not work well, no matter what values one chooses. Alternatively, one may resort to adaptive control. Generally, however, this approach requires slow variation in the plant parameters. One could also, in principle, rely on robust control, perhaps within the H_∞ framework. The drawback of this approach, however, is that while robust performance may be guaranteed, *uniformly* robust performance is not. These claims should not be surprising since neither adaptive control or robust control is really meant to compensate for strong nonlinearities in the plant model.

Given this fact, therefore, one should at least ideally consider nonlinear control— global or feedback linearization. Indeed if the nature and extent of nonlinearity is known reasonably well, through a reasonably accurate plant model, one would do just that. Moreover, even in the absence of a formal model, it is our conjecture that the human operator of the process, having learned the peculiarity of its behavior, develops response behavior that in practice amounts to a nonlinear control scheme that compensates for the dominantly nonlinear, and undesired, characteristics of the process. In effect s/he globally linearize the process *and* compensates for the deficiencies in its dynamic response characteristics.

In the context of the current example, in particular, it seems plausible that a human operator would be able to compensate for variations in the plant parameters, as required and as shown in Figure 4 produce response pattern superior to *any* linear control strategy.

Analysis of Response Pattern

Clearly, assuming that the control action of the human operator is described in linguistic form, the key factor would be the manner of definition of the rule set and its constitutive linguistic term set. This is evident, as shown in Figure 5, in the manner of definition of the linguistic term set defined over the domain of definition of the process error.

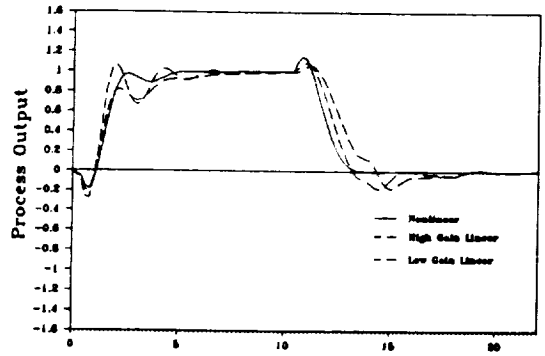


Figure 4. Response patterns of fuzzy vs. linear control.

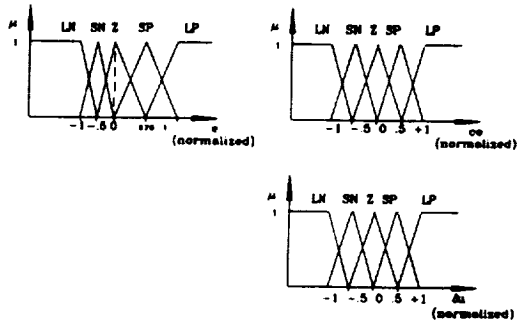


Figure 5. Definition of fuzzy membership functions.

In particular, the asymmetry in the definition of terms such as *small-positive* and *small-negative*, denoted in the figure by SP and SN respectively, reflects the variation in the *proportional* gain across the origin of the domain of definition of e.

Now, using the formalism presented earlier, one can show that the operator's action, interpreted above in linguistic terms, effectively amounts to a nonlinear control scheme

$$u = k_p e + k_i \int_0^t e \, d\tau, \qquad (28)$$

where k_p, is given by $k_p = k_{p0} - \alpha \mathrm{sgn}(e)/b$, which in the case of the regulation problem, in effect *cancels* the nonlinear terms which we attributed earlier to parametric variation[4].

CONCLUSION

[3]Actually it would be more accurate to consider variation as a function of the process input so as to reflect the coupling between state and input variables, however, in closed loop control the input is itself a function of the error.

[4]In reality when the setpoint is changed, this cancellation does not hold in the exact sense, however, since the plant dynamics is still linearized and stable, treating the setpoint change effect as a disturbance which results in diminishing transients is a reasonable assumption.

In this paper we showed how fuzzy control can be viewed as a paradigm for designing nonlinear control strategies in situations where the plant model is not a priori known— at least sufficiently well— to warrant the application of conventional control theory. In particular, we made a point regarding the use of fuzzy control in situations that occur frequently in industrial process control where (nonlinear)dependence of the parameters of the plant on its state variables precludes the application of linear control theory and thus nonlinear control, albeit by means of fuzzy control, seems to be the most appropriate approach. The framework presented here, however, is somewhat restrictive in that it requires a specific form for parametrization of fuzzy sets(LR) and places some restrictions on the manner of definition of the control rules($\sum \mu = 1$). To be more widely applicable, this framework needs to to allow for a wider range of nonlinear control schemes and also to allow for nonparametrized fuzzy sets.

APPENDIX.

A. Parametrization

Although not absolutely essential, parametrization simplifies quantitative description of fuzzy subsets. In LR parametrization(Dubois and Prade 1980), a fuzzy subset $\tilde{A}$, defined on some universe of discourse U, is characterized, in terms of its membership function , as follows:

$$\mu_{\tilde{A}}(u) = \begin{cases} L\left((u_0 - u)/\alpha\right) & \text{if } u \le u_0 \\ R\left((u - u_0)/\beta\right) & \text{if } u > u_0 \end{cases} \qquad (29)$$

where, as shown in Figure 6, $L(\cdot)$ and $R(\cdot)$ characterize the left and right halves of $\tilde{A}$, relative to its *center value*, u_0, that is where the linguistic term that $\tilde{A}$ represents fully achieves its meaning, or is maximally satisfied. Moreover, α(and β) parametrize $L(\cdot)$(and $R(\cdot)$), which typically takes the form

$$L(x) = \begin{cases} \max\left(0, 1 - |x|^p\right), \text{ or} \\ e^{-|x|^p}, \text{ or} \\ \dfrac{1}{1+|x|^p} \end{cases}, \qquad (30)$$

where $p \ge 1$ in all cases.

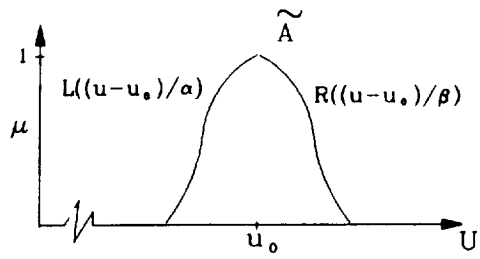


Figure 6. Parametrization of a fuzzy subset.

Finally, it is sometimes sufficient to use a simple linear form, based on $L(x) = \max\left(0, 1 - |x|\right)$, in which case, α(or β), discussed above, would represent the *inverse* of the slope of the characteristic function:

$$R(u) = 1 - (u - u_0)/\beta, \qquad (31)$$

$$L(u) = 1 - (u_0 - u)/\alpha. \qquad (32)$$

REFERENCES

Dubois, D and Prade, H.(1980). *Fuzzy Sets and Systems.* Academic Press, NY.

Franklin G., Powell, D., and Emami-Naeini, A.(1991). *Feedback Control of Dynamic Systems.* Second Edition. Addison Wesley, NY.

Langari, G.(1990) *A Framework for Analysis and Synthesis of Fuzzy Linguistic Control Systems.* Ph.D. Dissertation. University of California, Berkeley.

Langari G.(1992) Nonlinear Formulation of a Class of Fuzzy Linguistic Control Systems. In *Proceedings of the American Control Conference.*

Langari G. and Tomizuka, M.(1988). Fuzzy Linguistic Control of Arc Welding. In *Sensors and Controls for Manufacturing: Proceedings of the ASME Winter Annual Meeting.* Vol. PED-33. ASME, NY.

Langari, G. and Tomizuka, M.(1990) Analysis and Synthesis of Fuzzy Linguistic Control Systems. In R. Shoureshi(Ed.) *Intelligent Control 1990: Proceedings of the ASME Winter Annual Meeting.* Vol. DSC-23. ASME, NY.

Mamdani, E. and Assilian, S.(1975). An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. *International Journal of Man Machine Studies.* Vol. 7, No. 1.

Sugeno, M.(1985). *Industrial Applications of Fuzzy Control.* North Holland, Amsterdam.

Zimmermann, H.(1991) *Fuzzy Set Theory and its Applications.* Second Edition Kluwer Academic Publishers.

Truth-Valued-Flow Inference(TVFI) and Its Applications in Approximate Reasoning

Pei-zhuang Wang
Institute of Systems Science, National University of Singapore, Singapore 0511
On leave from Beijing Normal University, Beijing 100088, China

Hongmin Zhang, Wei Xu
Aptronix, the Fuzzy Logic Technology Company
2150 North first Street, Suite 300, San Jose, CA 95131, U.S.A.

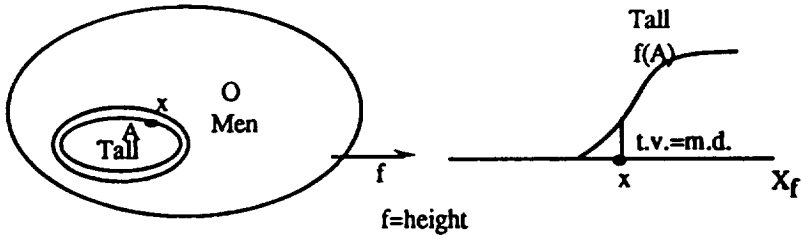
Abstract

In this paper, we introduce the framework of the theory of Truth-valued-flow Inference(TVFI) which was presented by the authors and has been successfully made into products by Aptronix, the Fuzzy Logic Technology Company. Even though there are dozens of papers presented out on fuzzy reasoning, we think it is still needed to explore a rather unified fuzzy reasoning theory which has the following two features: the one is that it is simplified enough to be executed feasibly and easily; and the other is that it is well structural and well consistent enough that it can be built into a strict mathematical theory and is consistent with the theory proposed by L.A.Zadeh. TVFI, introduced in this paper, is one of the fuzzy reasoning theories that satisfies the above two features. It presents inference by the form of networks, and naturally views inference as a process of truth values flowing among propositions.

1. What is inference?

Inference is truth values flowing among propositions. Here, the name 'truth value' is taken by logicians and stands for an abstract quantity who can be calculated by means of logical operations and used to evaluate the truth of propositions.

A proposition is a sentence "u is A" which can be viewed as has to be judged (may be fail). For example, " John is tall" or " John's height is tall" are propositions. Each proposition can be decomposed into two parts: A—a concept, a subset of a universe U; u—an object or its state respects to some factor, a point of U. If u stands for an object, like John, Mary,..., we usually denote the discussion universe U as O which consists of objects; if u stands for some state of an object, like height, weight,... we usually denote the discussion universe as X_f, which is the states space of the factor f.



f=height

A concept TALL, for example, can be represented as a fuzzy subset in an universe U. But U is not uniquely selected, it can be selected as O or X_f (shown in the above figure). Each concept can be represented as not only one but a class of membership functions; how to make a selection depends on what is the universe X or what is the variable x. So that , the combination of a concept A and a variable x, denoted as A(x), determines a conceptual representation. When x is fixed, it is the proposition 'x is A'; when x is varying, it is called a predicate. A predicate corresponds to a fuzzy subset in X.

A(x) offers us making judgment: What about the truth of it? It comes the truth value T(A(x)), the truth degree of proposition 'x is A'. It is equal to the membership degree $\mu_A(x)$. The form of truth values can be real numbers in [0,1] or linguistic values such as RATHER TRUE, VERY FAIL,... for examples, which are described as fuzzy subsets of [0,1].

A(x) also provides us a piece of information; since the concept A is usually a common sense, we are concerned chiefly with the variable x: where does it occur? In this sense, truth value T(A(x)) is the **possibility** of x under the constraint A. It comes the possibility theory presented by L.A.Zadeh.

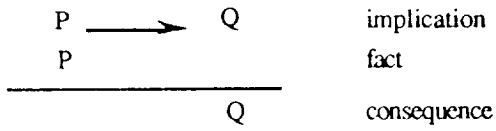
"John is tall" provides the information that the height of John is in the area of tall: it occurs at x with possibility $T(A(x))=\mu_A(x)$.

By means of the Falling shadow theory, a possibility distribution is the covering function of a random set. While the probability distribution of a discrete random variable is also the covering function of it, so that we can view possibility as a generalization of probability as that: possibility is probability if variable x is to have exclusiveness.

2. Introduction of the Concept of Truth Valued Flow Inference

First let's see why can we see the inference processes as truth values flowing among propositions? That is how inference channels realize inference as logic system does. Let us consider the syllogism inference as follows:

If x is a person, then it will die
John is a person
So that John will die



When we face an object, x=John. The fact is: "John is a person". i.e.,

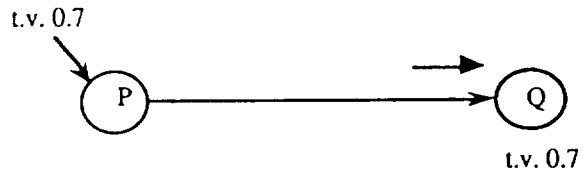
$$T(P(x))=T(\text{Person(John)})=1$$

By means of the implication "If x is a person, then it will die", denoted as $P\rightarrow Q$, we get

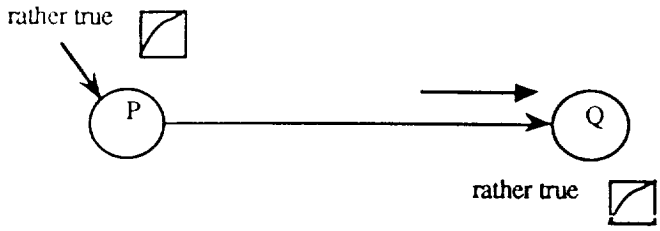
$$T(Q(x))=T(\text{end in dead(John)})=1.$$

Then we get the consequence: John will die. Here, we can see that an implicate likes a channel transferring truth value from head to tail.

When the fact does not qualify the head P completely but partly support it with truth value 0.7 for example, then the consequence is not certainty, we don't accept Q with truth value 1 but 0.7. This is the uncertainty inference, it can be also viewed as the truth value of input transferred to the tail along a inference channel.



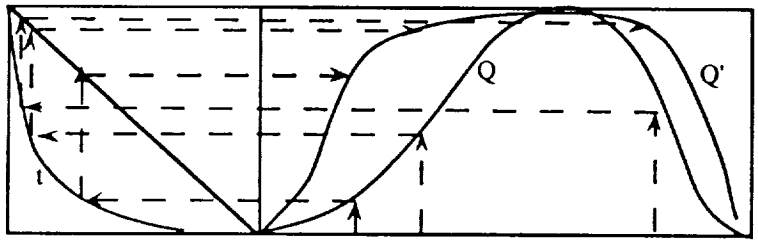
Of course, the truth values can be a linguistic value such as RATHER TRUE, VERY TRUE,..., the inference channel also transfers them from its head to its tail.



In this case we need the theory of Truth valued qualification(Baldwen 1979):

$$(y \text{ is } Q) \text{is } t = Y \text{ is } Q'$$

$$\mu_{Q'}(y) = t[\mu_{Q}(y)]$$



when the variables x, y are given, an implication

$$(\forall(x, y)) \text{ if } P(x) \text{ then } Q(y)$$

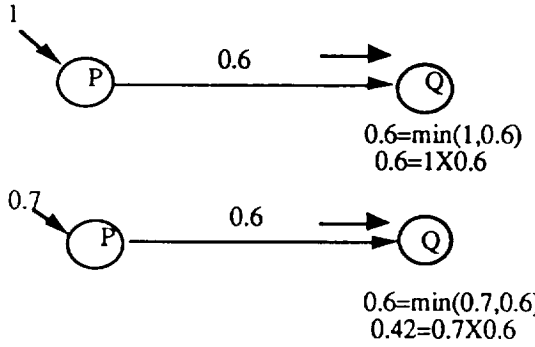
is determined by the pair of concepts P and Q. So an inference channel, through whom truth values can flow, can be denoted as [P,Q]. We call that the channel [P,Q] connects with concepts P and Q; P is its head and Q is its tail. A channel does not connect with propositions but concepts. The function of a channel is only transferring truth values, it is independent of how much truth value does its head have.

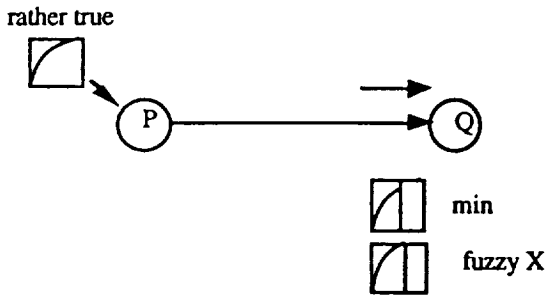
Inference channels have different qualities on transferring truth values. We call a channel [P,Q] has a quality coefficient q or call [P,Q] a q-quality channel if

$$\text{t.v.output } t' = \text{t.v.input } t \wedge^* q$$

Where $\wedge^* = \times$ or min or others.

When $\wedge^* = \times$, we call channel has 1-q friction, when $\wedge^* = \min$, we call q the transfer capacity of the channel.

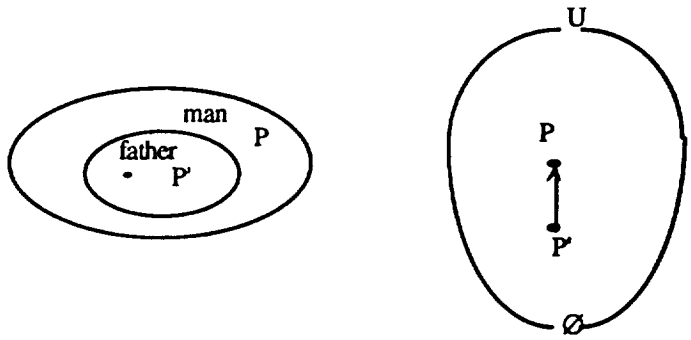




3. Properties of channels

For simple, we consider the head and tail of channels are all ordinary subsets. There are some basic properties of inference channels.

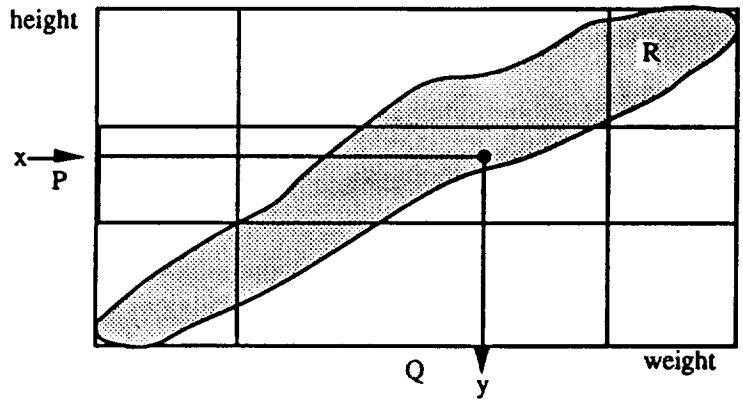
PROPERTY 1. If $P \subseteq Q$ then [p,Q] is an 1-channel, called Natural channel



A concept in the Cartesian product space of X(x-Universe) and Y(y-Universe) is called a **relation** between x and y. For example, O = a group of people, factor f = height, g = weight, $X=X_f$, $Y=X_g$. For any $o \in O$, define x=f(o), ,y=g(o), and denote the set of (x,y) as

$$R = \{(x,y) \mid o \in O\}$$

R is height-weight relation respect to O



R is the promised range of the point (x,y). It means that (x,y) cannot occur outside of it. That is

$$(x, y) \in R = X \times Y \cap R$$

Because of $x \in P \Leftrightarrow (x,y) \in P \times Y \Leftrightarrow (x,y) \in P \times Y \cap R$,

and $y \in Q \Leftrightarrow (x,y) \in X \times Q \Leftrightarrow (x,y) \in X \times Q \cap R$

when $P \times Y \cap R \subseteq X \times Q \cap R$

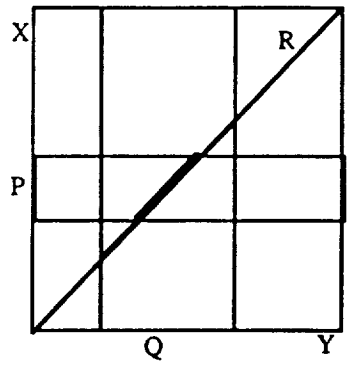
According to Property 1, we can say [P,Q] is an 1-channel. So we get the next property

PROPERTY 2. For a given relation R between X and Y, if

$$P\times Y\cap R \subseteq X\times Q\cap R$$

then [P,Q] is an 1-channel from X to Y. It is called a channel **under relation R**, and R is called the **ground relation** of the channel.

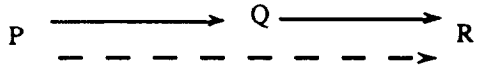
Property 1 is a special case of property 2. Indeed $\subseteq$ is a binary-relation



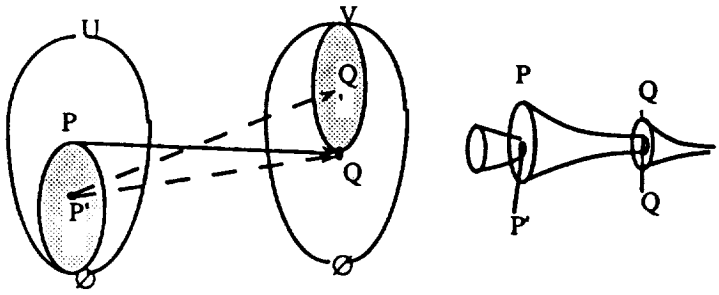
$P \subseteq Q$

Note: A class of inference channels can be generated from a relation.

PROPERTY 3. If [P,Q] and [Q,R] are two 1-channels then [P,R] is a channel



PROPERTY 3'. If [P,Q] is a 1-channel, $P' \subseteq P$ and $Q \subseteq Q'$ then [P',Q'] is a 1-channel.



For simplicity, $[P,Q]\in C(X,Y)$ or C stands for [P,Q] is a 1-channel from X to Y.

PROPERTY 4.

$$[P_1,Q]\in C \text{ and } [P_2,Q]\in C \Rightarrow [P_1\vee P_2,Q]\in C$$

$$[P,Q_1]\in C \text{ and } [P,Q_2]\in C \Rightarrow [P,Q_1\wedge Q_2]\in C$$

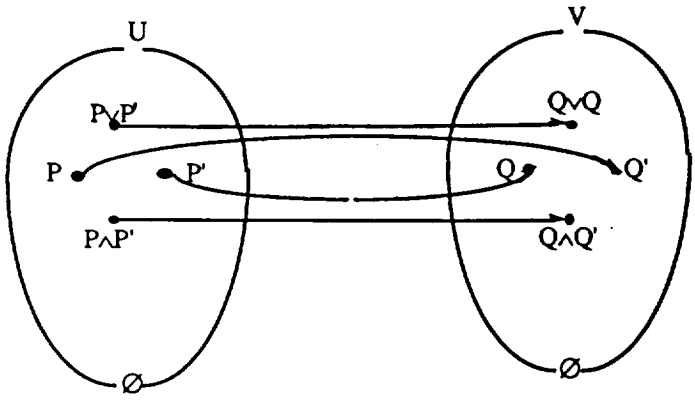
PROPERTY 4'.

$$[P_1,Q_1],\ [P_2,Q_2]\in C \Rightarrow [P_1\vee P_2,Q_1\vee Q_2],[P_1\wedge P_2,Q_1\wedge Q_2]\in C$$

THEOREM. Let $c_1=[P_1,Q_1]$, $c_2=[P_2,Q_2]$ define

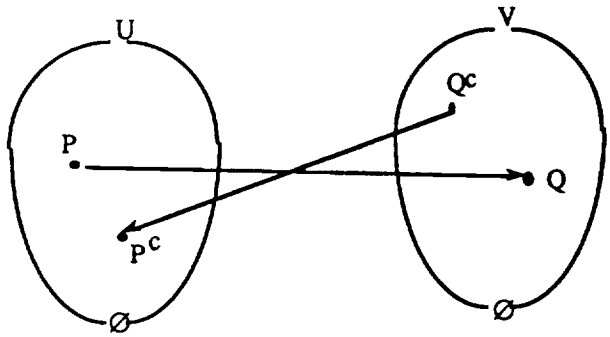
$$c_1\vee c_2=[P_1\vee P_2,Q_1\vee Q_2],c_1\wedge c_2==[P_1\wedge P_2,Q_1\wedge Q_2]$$

Then $(C(X,Y),\wedge,\vee)$ forms a lattice, and it is called the channel lattice.



PROPERTY 5.

$$[P, Q]\in C(X,Y) \Rightarrow [Q^c, P^c]\in C(Y, X)$$



DEFINITION Let $c_1=[P_1,Q_1]$, $c_2=[P_2,Q_2]$ if $P_1\supseteq P_2$, $Q_1\subseteq Q_2$ then c_1 is more valuable than c_2, denoted as $c_1\Rightarrow c_2$. A channel c in C is called valuable channel if there isn't other channel c' in C such that $c'\Rightarrow c$. The subset of valuable channels is denoted as V.

About the concepts of "information value" and "belief degree" of a channel, the bigger the head and the smaller the tail, the more information the channel, and therefore the more valuable the channel; on the other hand, it has the smaller belief degree. They can be represented by the following formula.

Suppose $P \rightarrow Q$ is a channel, $P'\subseteq P$, $Q'\supseteq Q$, then we have know that $P' \rightarrow Q'$ is also a channel. And

$$\text{belief-degree}(P' \rightarrow Q') \geq \text{belief-degree}(P \rightarrow Q),$$

$$\text{information-value}(P' \rightarrow Q') \leq \text{information-value}(P \rightarrow Q).$$

For any $x\in X$, define

$$Q_X=\cap\{Q \mid P\rightarrow Q\in C, x\in P\}$$

and assume that for any $x\in X$, $Q_X\neq\varnothing$, then we have

DEFINITION. Define

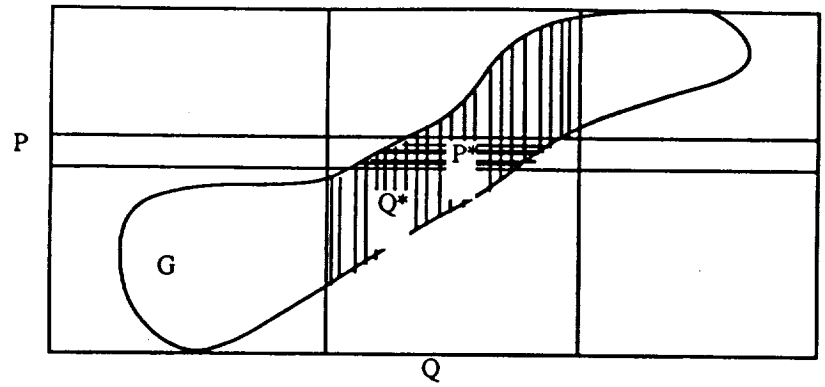
$$G=\cup\{Q_X\times\{x\} \mid x\in X\}$$

G is called the background graph of lattice C.

THEOREM. Let C(X,Y) be the channel lattice generated from a ground relation R, let G be the ground graph of C(X,Y), then we have that G=R.

THEOREM. Lattice C can be determined uniquely by its background graph G. That is to say that $P \rightarrow Q$ is a channel in C if and only if $P^*\subseteq Q^*$.

where $P^*=P\times Y \cap G$, $Q^*=X\times Q \cap G$. (As shown in the following figure)



DEFINITION. Giving channel c=[P,Q],

$$R(c) = P \times Q \cup P^c \times Y$$

is called inference relation of channel c.

THEOREM c=[P,Q] $(P \in X, Q \in Y) \in C$ if and only if $R(c) \supseteq G$.

THEOREM. c=[P,Q] $(P \in X, Q \in X) \in C$ if and only if $Q \supseteq P$.

THEOREM. About the relations of background graphs of channels, we have

$$R(c_1 \text{ and } c_2) = R(c_1) \cap R(c_2)$$

$$R(c_1 \text{ or } c_2) = R(c_1) \cup R(c_2)$$

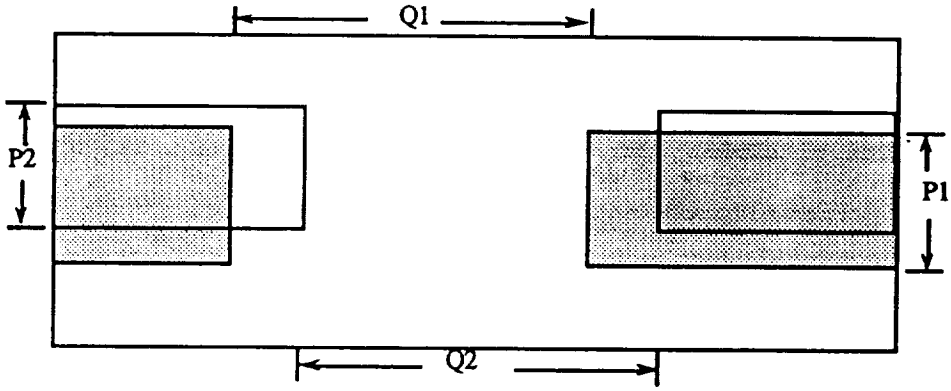
$$R([P,Q_1] \text{ and } [P,Q_2]) = R([P,Q_1 \wedge Q_2])$$

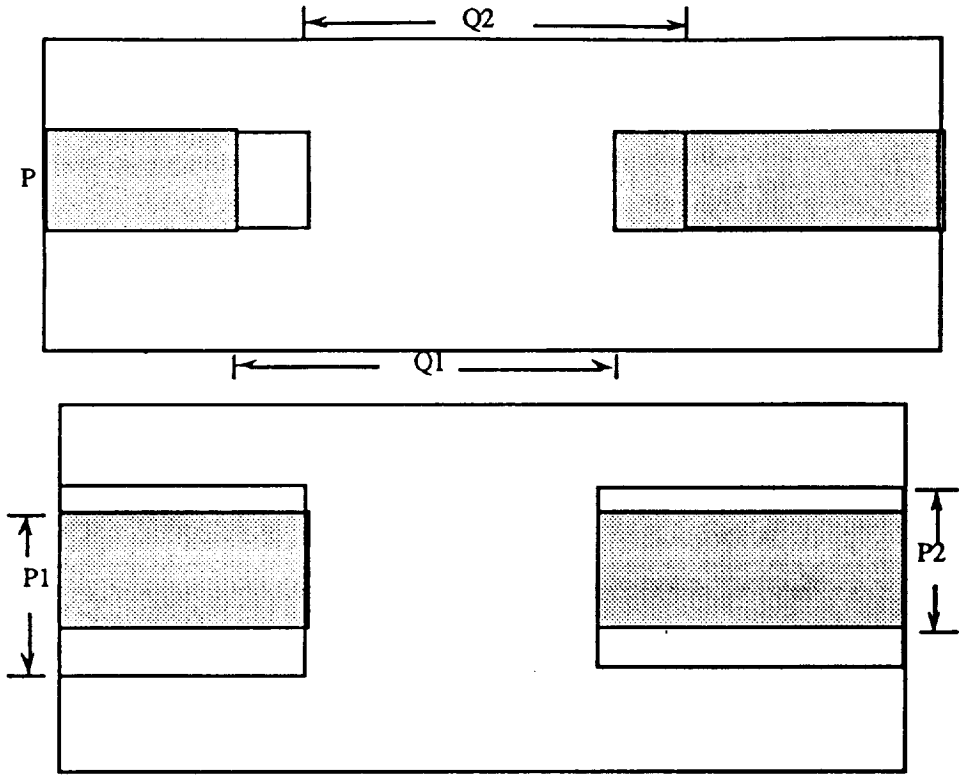
$$R([P,Q_1] \text{or} [P,Q_2]) = R([P,Q_1 \vee Q_2])$$

$$R([P_1,Q] \text{ and } [P_2,Q]) = R([P_1 \vee P_2,Q])$$

$$R([P_1,Q] \text{or} [P_2,Q]) = R([P_1 \wedge P_2,Q])$$

These can be shown in the following figure.





4. Fuzzy channels Lattice

For given $\lambda \in [0,1]$, an λ-channel lattice L_λ consists of those channels who transfers truth value at least λ to the tail whenever the head is fulfilled with truth value 1.

For every definition of truth values operations $\vee^*$ and $\wedge^*$, a channel [P, Q] is a λ-channel if and only if the qualify q of it is equal or larger than λ.

A λ-channel lattice satisfies axioms 1-5 as same as 1-channel lattice.

About the L_λ ($\lambda \in [0,1]$), we obviously have the following proposition:

PROPOSITION: If $\lambda \leq \mu$, then $L_\lambda \supseteq L_\mu$.

Let $L_\lambda(\lambda \in [0,1])$ be a λ-cut subset, then $\{L_\lambda\}$ $(\lambda \in [0,1])$forms a fuzzy set on L called a fuzzy channel lattice, where L is the set of all channels.

Note that

$$\lambda \leq \mu \Rightarrow G_\lambda \supseteq G_\mu$$

$$\lambda \leq \mu \Rightarrow R_\lambda \supseteq R_\mu$$

where G_λ, G_μ and R_λ, R_μ are ground graph and ground relation of L_λ , L_μ respectively.

There is a difference between 1-channel lattice and λ-channel lattice($\lambda<1$). In 1-channels, if [P,Q] and [P.Q'] are both 1-channels then

$$Q \cap Q' \neq \emptyset$$

otherwise, we have $[P,\emptyset]=[P,Q\cap Q']$ hold. From this, we have [P,R] (for any R) hold, especially $[P,Q^C]$. Therefore, we have [P,Q] and $[P,Q^C]$ are both hold in the same time , this is a contradiction in mathematics. But in λ-channels ($\lambda<1$), $Q\cap Q'=\emptyset$ may be hold.

Principles of quality qualification:

1. Let [P,Q] is a q-channel and $[P,Q]=[P, Q_1 \text{ or } Q_2 \text{ or...or } Q_n]$, then for $i=1,...,n$, $[P, Q_i]$ are all q/n-channels.

2. Let [P,Q] is a q-channel and [P,Q]= [P_1 and P_2 and...and P_n, Q], then for i=1,...,n, [P_i,Q] are all q/n-channels.

Following we further discuss this problem from another view of point.

DEFINITION: Given a background graph G on X×Y, which is a fuzzy subset with membership function G(x,y). We can define two fuzzy subsets N and Π on P(X)×P(Y) as follows:

$$N(P,Q)=1-\wedge\{\wedge\{G(x,y) \mid y\notin Q\} \mid x\in P\}$$

$$P(P,Q)=\vee\{\wedge\{G(x,y) \mid y\in Q \mid x\in P\}$$

$P \rightarrow Q$ is called a λ-channel if $N(P,Q)\geq\lambda$. $x\rightarrow y$ is callled a λ–offshoot if $\Pi(\{x\},\{y\})\geq\lambda$.

THEOREM: For any fixed $x\in X$, $N_x=N(\{x\},.)$ and $\Pi_x=\Pi(\{x\},.)$ are necessity measure and possibility measures on P(Y) respectively. That is: $N_x(\varnothing)=0$, $\Pi_x(Y)=1$, and

$$N_x(P\cap Q) = \min (N_x(P), N_x(Q))$$

$$N_x(P\cup Q) \geq \max (N_x(P), N_x(Q))$$

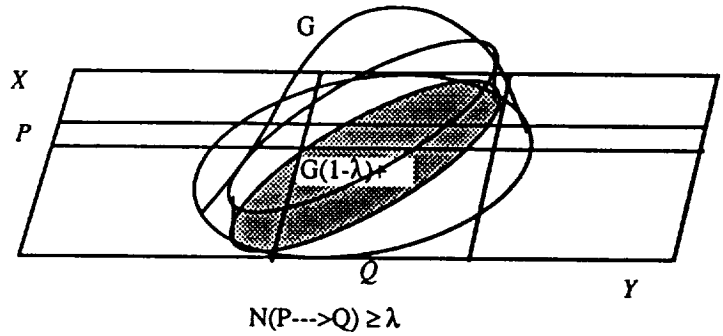
$$\Pi_x(P\cup Q) = \max (\Pi_x(P), \Pi_x(Q))$$

$$\Pi_x(P\cap Q) \leq \min (\Pi_x(P), \Pi_x(Q))$$

$$N_x(P) = 1- \Pi_x(P^c)$$

THEOREM: For any λ(0<λ≤1), N_λ, the λ-cut of N, is a channels lattice with respect to operations ∪ and ∩. The corresponded background graph is $G_{(1-\lambda)+}$, the 1-λ open cut of G, i.e.

$$(P,Q)\in N_\lambda \Leftrightarrow P^*=P\times Y\cap G_{(1-\lambda)+}\subseteq X\times Q\cap G_{(1-\lambda)+}=Q^*$$



$N(P\text{---}>Q) \geq \lambda$

THEOREM: For any λ(0<λ≤1), $(P,Q)\in\Pi_\lambda$ if and only if for any $x\in P$ there is a point y such that

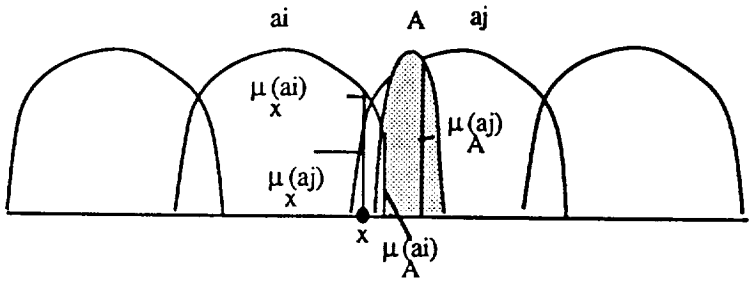
$$(x,y)\in (P\times Q)\cap G_{\lambda+}$$

The membership degree of (x,y) with respect to G is equals to the necessity of offshoot $x\rightarrow y$:

$$G(x,y) = \Pi(\{x\} \rightarrow \{y\})$$

5. Truth Valued Flow Neural Networks

We call a Universe X, or corresponded variable x, is atomlizable if there are only finite possible atoms a_i (i=1,...,n) such that any information about x is stated through them in a problem.



Let X,Y are atomizable, $\underline{X}=\{a_i\}(i=1,...,n)$, $\underline{Y}=\{b_j\}(j=1,...,m)$. The Cartesian product space $\underline{X}\times\underline{Y}$ can be represented as an n×m squares, and a ground relation (or graph) can be represented as a matrix $R_{n\times m}$ with elements 0 or 1. For any head a_i, the valuable channel in the 1-channel lattice L_1 is $[a_i, B_i]$, where the tail can be represented by atoms of Y:

$$B_i = \vee \{b_j \mid r_{ij}=1\}.$$

i.e.,
$$[a_i,B_j] = OR\{[a_i,b_j] \mid r_{ij}=1\}$$
$$=[a_i,b_{i1}] \text{ or } [a_i,b_{i2}] \text{ or...or } [a_i,b_{im_i}], \text{ where } r_{iij}=1.$$

According to the principle of quality qualification, $[a_i,b_{ij}]$ are $1/m_i$-channels.

For a given ground relation matrix Rn×m of an 1-channel lattice L_1, normalizing each arrow of it, we get a matrix $L_{n\times m}$ called TVF(truth valued flow) matrix of L_1:

$$l_{ij} = \begin{cases} r_{ij} / \Sigma_k r_{ik} & \text{if } \Sigma_k r_{ik} \neq 0 \\ 1/m & \text{else} \end{cases}$$

Truth values flow among the atoms from X to Y is a TVF Networks which consists of atom-channels(head and tail are atoms). The weight of $[a_i.b_j]$ is l_{ij} and the Propagation rule is:

$$n_j = f(\vee^*(m_i \wedge^* l_{ij}))$$

where
m_i-truth values at input;

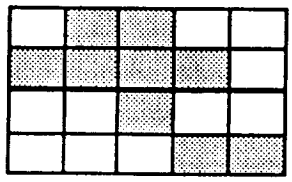
n_j-truth values at output,

f- threshold function,

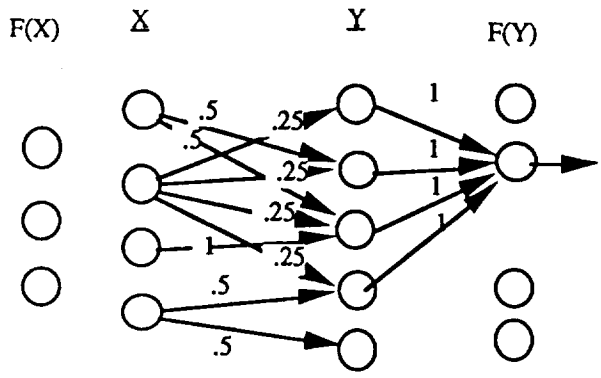
$(\vee^*,\wedge^*)$=(max,min) or (+, ×) or other fuzzy operations.

From the following specific example, we can know the general TVF Networks structure.

EXAMPLE: Let $X=\{a_1,a_2,a_3,a_4\}$ and $Y=\{b_1,b_2,b_3,b_4,b_5\}$, the ground graph is presented by the shadow area (left of the following Fig.), and ground relation R is presented by the $L_{4\times5}$ matrix (right of the following Fig.), then this TVF network has the following structure (down of the following Fig.)



	.5	.5		
.25	.25	.25	.25	
		1		
			.5	.5



6. Applications of TVFI

(1) TVFI Applications in AI

In the above section , we have gotten that for every ground graph, we can get a True-Value-Flow inference network. In AI field, the ground graph is just the database, and the Truth-Value-Flow inference network is just the knowledge base. So we actually realize the transferring from database to knowledge using Truth-Value-Flow inference. In practice, it is also very important to get ground graph from some kinds of database. In the following we will introduce several kinds of database, the ways to get database, and the ways to get knowledge base from database.

The kinds of database we often use are listed as follows:

1) statistical sample: $\{(x_k,y_k)\}$;

2) relation data base: $R(x_k, y_k, z_k,...)$;

3) causality rule: f=ma;

4) experts experiences: if... then...;

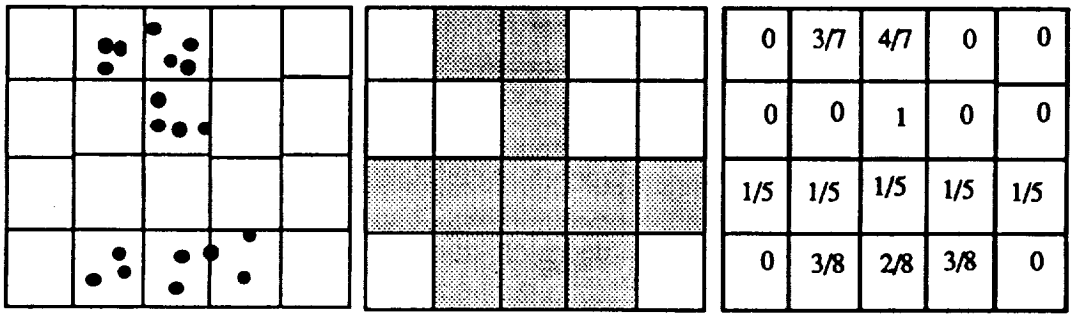
Below we will give a specific method how to get ground graph and ground relation from statistical samples, and how to get TVF neural networks (knowledge base) from ground graph (database).

For each i, get a distribution $\{l_{ij}\}$

$$l_{ij} = \begin{cases} m_{ij}/m_i & \text{if } m_i \neq 0 \\ 1/m & \text{else} \end{cases}$$

where $m_{ij} = \Sigma_k(m_{a_i}(x_k) \times m_{bj}(y_k))$, $m_i = \Sigma_j m_{ij}$.

Note: When there is not point occurred in an arrow(for example, 3th arrow in the following Fig.) the relation or graph is not empty but full in Y, and l_{ij} are uniformly distributed.



0	3/7	4/7	0	0
0	0	1	0	0
1/5	1/5	1/5	1/5	1/5
0	3/8	2/8	3/8	0

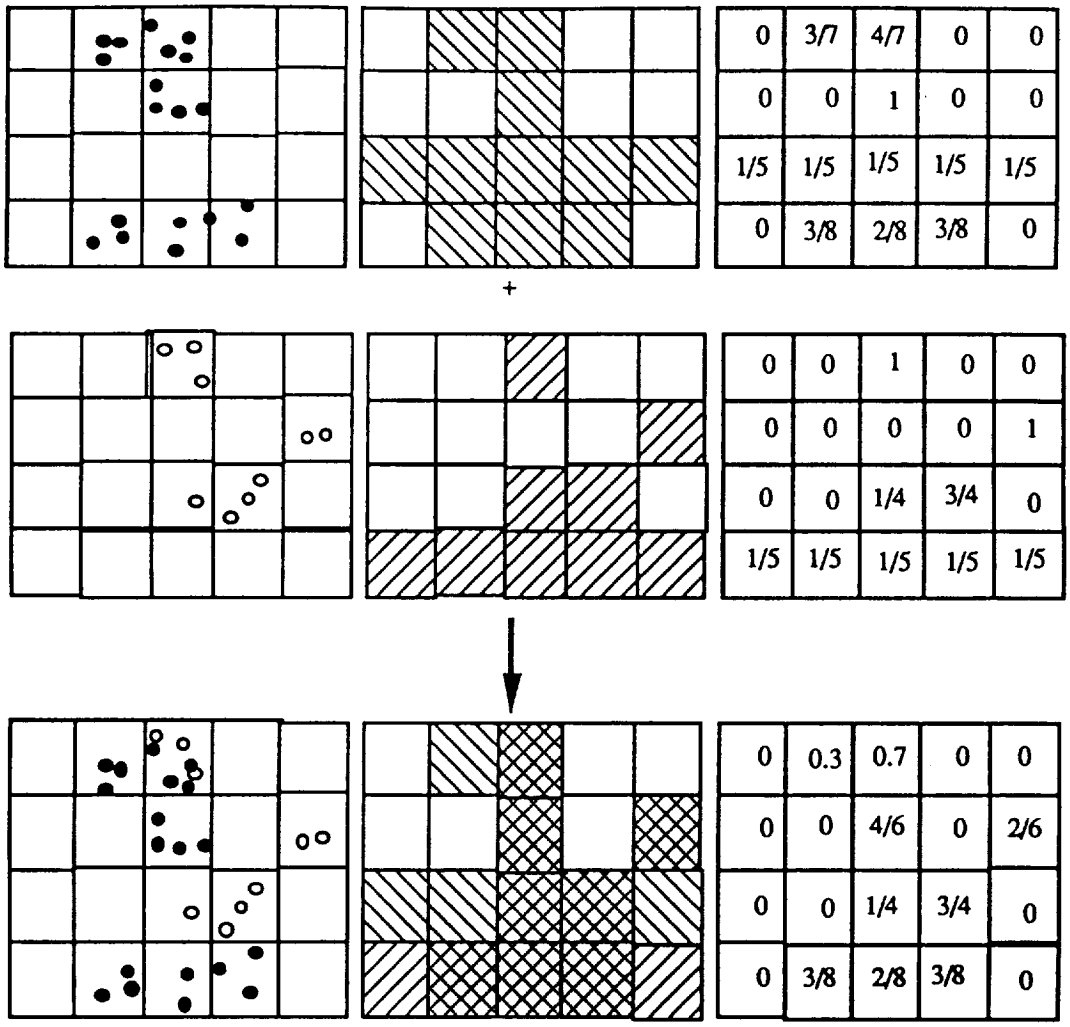
When our information (i.e. data base) is not complete, we can only get a sublattice of an unknown channel lattice.

DEFINITION. A channel lattice L' is called a sublattice of a channel lattice L if the ground graph of L' contains the ground graph of L.

In data base, the sample of statistics or the relation form corresponded to a sublattice L' is more incomplete than that of channel lattice L.

C-3

For an incomplete channel lattice, we can extend data base by adding any kind of information and knowledge.



0	3/7	4/7	0	0
0	0	1	0	0
1/5	1/5	1/5	1/5	1/5
0	3/8	2/8	3/8	0

+

0	0	1	0	0
0	0	0	0	1
0	0	1/4	3/4	0
1/5	1/5	1/5	1/5	1/5

0	0.3	0.7	0	0
0	0	4/6	0	2/6
0	0	1/4	3/4	0
0	3/8	2/8	3/8	0

DEFINITION. Let $L_{n\times m}$ be the TVF matrix of channel Lattice L, then

$$l_j=\max_i l_{ij} \text{ and } l=\min_j l_j$$

are called the inductable degree of L at bj and of L respectively. If $l \geq l^*$(or $l_j \geq l^*$), we call L is l^*-sufficient(or for bj).1-sufficient is called completely sufficient.

To know which head is able to infer to b_j, we are natural to inversely search along the weightiest channel (whose quality equals to l_j), if lj is larger than the given threshold l^*, then we find out the head we want to know.

After adding information to L, if the inductable degree is still smaller than the given threshold l^*, It means that the factor concerned with x is not enough to infer y. We have to move X into another factor space.

Let F be the set of factors concerned with variable y. Let L_f be the channel lattice from x_f to y. Set $X=X_f$, the inductable degree is l_f. The more complex the factor f, the higher the inductable degree of L_f.

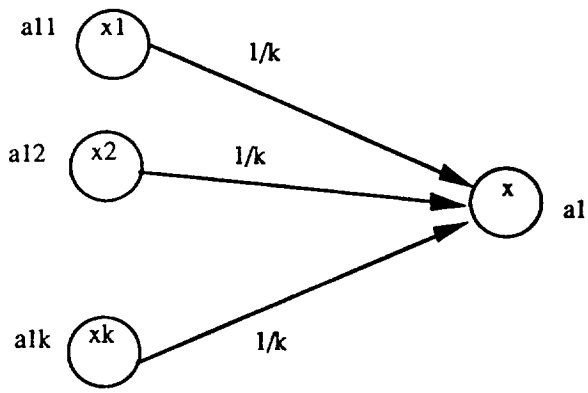
When l_f is enough, suppose that

$$f=f_1 \vee ... \vee f_k$$

where $f_1...f_k$ are simple factors which concerned with variable $x_1,...,x_k$ respectively, then an atom in x is in the form:

$$x_1 \text{ is } a_{1\,1} \wedge ... \wedge x_k \text{ is } a_{1k}$$

According to the principle of quality qualification, we can arrange a neural network as follows:

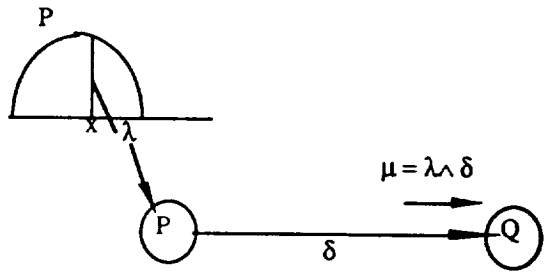


This is a TVFI neural network taken in factor spaces. It is actually the network representation of knowledge base. Thus we complete the transferring from database to knowledge base.

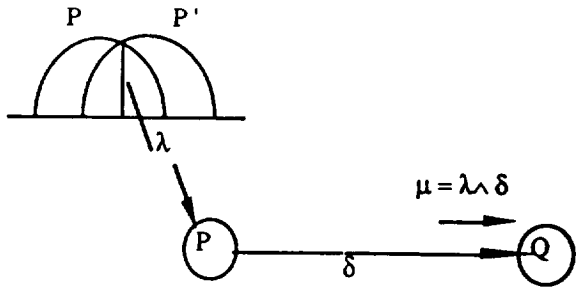
(2) TVFI Applications in Approximate Reasoning

Suppose we have a channel P→ Q, then we may execute many kinds of approximate reasoning along this channel. Following we give the execution of two kinds of most often using approximate reasoning using TVFI channel.

1) The input is an element x, in this case we can do approximate reasoning as follows:



2) The input is a fuzzy set P' (i.e. concept), in this case we can do approximate reasoning as follows:



Reference

[1] D.Dubois, H. Prade, Necessity measures and the resolution principle, IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-17, Nov. 3 (1987) 474-478

[2] G. Shafer, A Mathematical theory of Evidence, Princeton, NJ: Princeton University, 1976

[3] P.Z.Wang, H.M.Zhang, X.T.Peng, W.Xu, P.Z.Wang, Truth-valued-flow Inference, BUSEFAL No. 38, (1989)130-139

[4] P.Z.Wang, Dynamic description of net-inference process and its stability, ZHENGJIANG CHUANPUO XUEYUAN XUEBAO, Vol.2, No.2,3 (1988)156-163

[5] P.Z.Wang, H.M.Zhang, Truth-valued flow inference and its dynamic analysis*, BEIJING SHIFAN DAXUE XUEBAO, No.1(1989)1-12.

[6] P.Z.Wang, A factor spaces approach to knowledge representation, Fuzzy Sets and Systems, Vol.36;(1990)113-124

[7] P.Z.Wang , H.H.Teh, S.K.Tan, Fuzzy inference relation theory based on the shadow-representation approach, Proceedings of 8th International Conference of Cybernetics & Systems, New York,(1990)30-31

[8] P.Z.Wang, Fuzziness vs. randomness, Falling shadow theory, BUSEFAL No. 48 (1991)

[9] P.Z.Wang, H.M.Zhang, X.W.Ma, W.Xu, Fuzzy set-operations represented by falling shadow theory, in Fuzzy Engineering toward Human Friendly Systems, Proceedings of the International Fuzzy Engineering Symposium'91, Yokohama, Japan ,Vol.1,(1991) 82-90

[10] P.Z.Wang, Dazhi Zhang, The netlike inference process and stability analysis,'International Journal of Intelligent Systems, Vol. 7(1992)361-372

[11] X.H. Zhang, Francis Wong, H.C. Lui, P.Z. Wang, Theoretical Basis of Truth Value Flow Inference, the Proceeding of the First Singapore International Conference On Intelligent Systems 1992.

[12] L.A.Zadeh, Fuzzy sets as a basis for a theory of possibility, Fuzzy Sets and Systems, 1(1978) 3-28.

AUTHOR INDEX

Aldridge, J.	237	Kreinovich, V.	174
Atkins, M.	93	Ladage, R.	109
Berenji, H.	169	Langari, R.	348
Bezdek, J.	199	Lea, R.	127
Bonnisone, P.	1	Lerner, B.	73
Buckley, J.J.	170	Lin, Y.	295
Chappell, J.	76	Maor, R.	48
Chen, C.	282	Mitra, S.	309
Copeland, C.	127	Miwa, H.	107
Cox, C.	93	Morrelli, M.	73
Dauherity, W.	75	Murakami, J.	249
de Korvin	54	Murphy, M.	328
Desai, P.	285	Naito, E.	257
Espy, T.	237	Nishino, J.	107
Farber, R.	3	Ogmen, H.	30
Glover, C.	93	Ozawa, J.	257
Hall, L.	13	Pal, N.	199
Huang, S.	295	Pap, R.	93
Hayashi, I.	257	Pfluger, N.	344
Hirota, K.	249	Phelps, A.	113
Hodges, W.	113	Pin, F.	330
Hoy, S.	109	Quintana, C.	174
Imura, A.	95	Raju, G.	69
Isaksen, G.	273	Ramamoorthy, P.	295
Jani, Y.	48, 76, 127	Ramer, A.	282
Jansen, B.	285	Romaniuk, S.	13
Karr, C.	186	Ruspini, E.	343
Keller, J.	227	Saeks, R.	93
Khedkar, P.	169	Shenoi, S.	282
Kim, C.	273	Sherman, P.	108
Kissell, R.	93	Shipley, M.	54
Krishnapuram, R.	227	Smith, R.	183

Sousa, G. 76
Spiegel, R. 76
Sugeno, M. 107
Takagi, H. 196
Takagi, T. 95
Tawel, R. 346
Thakoor, A. 346
Tsao, E. 199
Tsoukkas, A. 174
Turner, W. 76
Urnes, J. 109
Ushida, H. 95
Valenzuela-Rendon, M. 184
VanLangingham, H. 174
Villarreal, J. 127
Wakami, N. 257
Walker, G. 113
Wang, H. 75
Wang, P. 196
Wang, P.Z. 354
Warburton, F. 108
Watanabe, Y. 330
Xu, W. 354
Yager, R. 49
Yamaguchi, T. 95
Yamauchi, K. 249
Yen, J. 75, 344
Ying, H. 40
Zhang, H. 354
Zhang, S. 295
Zhou, J. 69

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3 REPORT TYPE AND DATES COVERED
	January 1993	Conference Publication

4. TITLE AND SUBTITLE
Proceedings of the Third International Workshop on Neural Networks and Fuzzy Logic

5. FUNDING NUMBERS

6. AUTHOR(S)
Christopher J. Culbert, Editor

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Lyndon B. Johnson Space Center
Information Technology Division
Houston, TX 77058

8. PERFORMING ORGANIZATION REPORT NUMBER
NASA CP-10111
Vol. I Vol. II

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, D.C. 20546

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
S-701

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Unclassified/Unlimited Volume I - unlimited
Subject category 63 Volume II - unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Documented here are papers presented at the Neural Networks and Fuzzy Logic Workshop sponsored by the National Aeronautics and Space Administration and cosponsored by the University of Houston, Clear Lake. The workshop was held June 1-3, 1992 at the Lyndon B. Johnson Space Center in Houston, Texas. During the three days approximately 50 papers were presented. Technical topics addressed included adaptive system; learning algorithms; network architectures; vision; robotics; neurobiological connections; speech recognition and synthesis; fuzzy set theory and application, control, and dynamics processing; space applications; fuzzy logic and neural network computers; approximate reasoning; and multiobject decision making.

14. SUBJECT TERMS
fuzzy logic; non-Lipschitzian dynamics, parallel distributed models; algorithms; neural network; spatiotemporal patterns; neuron ring; fuzzy controllers; signal processing; pattern recognition

15. NUMBER OF PAGES
393

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	unlimited