

A HYBRID INTELLIGENT SYSTEM AND ITS  
APPLICATION TO MEDICAL DIAGNOSIS

By

PHAYUNG MEESAD

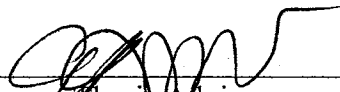
Bachelor of Science  
King Mongkut's Institute of Technology North Bangkok  
Bangkok, Thailand  
1994

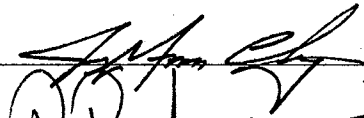
Master of Science  
Oklahoma State University  
Stillwater, Oklahoma  
1998

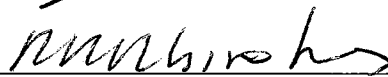
Submitted to the Faculty of the  
Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
DOCTOR OF PHILOSOPHY  
May, 2002

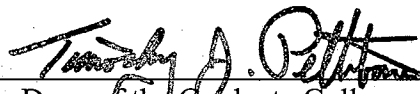
A HYBRID INTELLIGENT SYSTEM AND ITS  
APPLICATION TO MEDICAL DIAGNOSIS

Thesis Approved:

  
Thesis Advisor

  
R. Ramakrishna



  
Dean of the Graduate College

## PREFACE

In this study, a novel hybrid intelligent system (HIS) that provides a unified framework of numerical and linguistic knowledge representations is proposed. The proposed HIS is a hierarchical integration of an incremental learning fuzzy neural network (ILFN) and a linguistic model, i.e., fuzzy expert system (FES), optimized via the genetic algorithm (GA). The ILFN is a self-organizing network with the capability of fast, one-pass, online, and incremental learning. The linguistic model is constructed based on knowledge embedded in the trained ILFN or provided by the domain expert. The knowledge captured from the low-level ILFN can be mapped to the higher-level linguistic model and vice versa. The GA is applied to optimize the linguistic model to maintain high accuracy, comprehensibility, and completeness. The resulting HIS is capable of dealing with low-level numerical computation and higher-level linguistic computation. After the system is successfully constructed, it can incrementally learn new information in both numerical and linguistic forms. To evaluate the system's performance, several medical data sets have been used. The simulation results have shown that the proposed HIS achieved performance classification better than the individual standalone systems. The comparison results based on performance classification show that the linguistic rules extracted are competitive with, or even superior to, some well-known approaches in literature.

## ACKNOWLEDGMENTS

I wish to express my sincere thankfulness to my academic advisor, Professor Gary G. Yen, for his understanding guidance, productive instruction, brainstorming, and companionship. My genuine appreciation draws to my other committee members Professor Rama G. Ramakumar, Professor Jong-Moon Chung, and Professor R. Russell Rhinehart, whose direction, support, reassurance, and acquaintance are also worthwhile.

I would also like to thank the Intelligent Systems and Control Laboratory at Oklahoma State University for supporting resources. Furthermore, I would like to thank several people who are past and present members of the Intelligent Systems and Control Laboratory for their discussions and recommendations. In addition, I would like to thank Nitin Sharma, a Ph.D candidate under the supervision of Professor R. Russell Rhinehart, for providing data and discussion on a hot-and-cold mixing water simulator in the additional study on fuzzy temporal representation and reasoning.

I would also like to give my special gratitude to my wife and my daughter, Pongpun and Natchaporn, for their long lasting sedulousness and solace through times of dilemma, and for their love and understanding throughout this investigation. My gratitude also goes to my mother, my sisters, and my brother for their devotion, encouragement, and tolerance.

Finally, I would like to thank the Royal Thai Government, Ministry of University Affairs, for financial support during these three and one-half years of study.



## TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION .....	1
1.1 Medical Diagnosis Overview.....	1
1.2 Motivation of the Research.....	2
1.3 Organization of the Dissertation .....	6
II. LITERATURE REVIEW.....	7
2.1 Basic Architecture of Pattern Classification .....	7
2.2 Statistical Classifiers.....	10
2.3 Genetic Algorithms.....	11
2.4 Neural Networks .....	12
2.4.1 Multilayer Perceptron (MLP) Neural Network.....	13
2.4.2 Probabilistic Neural Network (PNN).....	14
2.4.3 Radial Basis Function (RBF) Neural Network .....	15
2.4.4 Self-Organizing Map (SOM) Neural Network .....	16
2.5 Expert (Knowledge-Based) Systems .....	17
2.5.1 Knowledge Representation .....	20
2.5.2 Mycin Expert System for Medical Diagnosis .....	22
2.6 Fuzzy Systems .....	24
2.6.1 Structure of Fuzzy Rules.....	25
2.6.2 Mamdani Models .....	25
2.6.3 Takagi-Sugeno (TS) Models.....	27
2.6.4 Constructing Fuzzy If-Then Rules.....	28
2.7 Hybrid Intelligent Systems .....	30
2.7.1 Single-Structure Hybrid Intelligent Systems (SHIS).....	33
2.7.1.1 Fuzzy Neural Networks .....	34
2.7.1.2 Neuro-Genetic Systems .....	36
2.7.1.3 Fuzzy Genetic Systems.....	36
2.7.2 Multi-Module Hybrid Intelligent Systems (MHIS) .....	37
2.7.2.1 Unified Architectures.....	38
2.7.2.2 Loosely Coupled Architectures.....	38
2.7.2.3 Tightly Coupled Architectures.....	42
2.7.2.4 Fully Integrated Architectures .....	43

Chapter	Page
2.7.3 Existing Hybrid Intelligent Systems .....	43
2.7.3.1 Knowledge-Based Artificial Neural Network .....	43
2.7.3.2 NeuroRule .....	44
2.7.3.3 Taha's Hybrid Intelligent Architecture .....	45
2.7.4 Combined Numerical and Linguistic Paradigms .....	46
III. THE ARCHITECTURE OF THE PROPOSED HYBRID INTELLIGENT SYSTEM .....	51
3.1 Incremental Learning Fuzzy Neural Network (ILFN) .....	52
3.1.1 Input Subsystem .....	54
3.1.2 Target Subsystem .....	57
3.1.3 Controller Module .....	58
3.1.4 Decision Layer .....	60
3.1.5 ILFN System Dynamics .....	60
3.1.6 Learning Process .....	61
3.1.7 Decision Boundaries .....	63
3.1.8 ILFN Classification Algorithm .....	64
3.2 Fuzzy Expert System (FES) .....	68
3.2.1 Knowledge Base of the proposed FES .....	69
3.2.2 Fuzzy If-Then Rule Generation .....	73
3.3 Network-To-Rule Module .....	75
3.3.1 ILFN2RULE Algorithm .....	76
3.3.2 Genetic Algorithm for Rule Optimization .....	80
3.3.3 Fuzzy If-Then Rule Encoding .....	80
3.3.4 Fuzzy If-Then Rule Decoding .....	83
3.3.5 Genetic Selection for the Number of Linguistic Variables .....	85
3.4 Rule-To-Network Module .....	89
3.4.1 RULE2ILFN Algorithm .....	90
3.5 The Decision-Explanation Module .....	91
3.5.1 Decision Boundaries .....	93
3.5.2 Conflict and Conflict Resolution Between Low Level and Higher Level .....	94
3.6 Increment Learning Characteristic of the Proposed HIS .....	95
IV. QUANTITATIVE MEASURES ON THE ACCURACY, COMPREHENSIBILITY, AND COMPLETENESS OF A FUZZY KNOWLEDGE BASE .....	97
4.1 Accuracy Measures .....	97
4.2 Comprehensibility Measures .....	100
4.2.1 Compactness .....	101
4.2.2 Linguistic Similarity .....	105
4.2.3 Consistency of Fuzzy Rules .....	108
4.3 Completeness Measure .....	110

Chapter	Page
4.4 Fitness Functions Implemented for the Genetic Algorithm.....	114
V. BENCHMARK SIMULATION RESULTS.....	115
5.1 Iris Data Set.....	115
5.1.1 Simulation Results for the Iris Data Set.....	116
5.2 Wisconsin Breast Cancer Data (WBCD).....	120
5.2.1 Simulation Results for the WBCD .....	121
5.2.2 Comparison Results for the WBCD.....	125
5.3 Additional Medical Diagnosis Data Sets .....	128
5.3.1 Breast Cancer data .....	128
5.3.1.1 Simulation Results for the Breast Cancer Data .....	129
5.3.2 Lymphography Domain.....	133
5.3.2.1 Simulation Results for the Lymphography Domain .....	134
5.3.3 Primary Tumor Domain.....	136
5.3.3.1 Simulation Results for the Primary Tumor Domain.....	137
VI. DEVELOPMENT OF A HIS GRAPHICAL USER INTERFACE .....	141
6.1 The Main Window of HIS-GUI.....	141
6.2 Load Data .....	142
6.3 Setting the Number of Training and Test Patterns.....	144
6.4 Setting ILFN Tuning Parameters.....	144
6.5 Training ILFN Network.....	145
6.6 ILFN Evaluation .....	147
6.7 Setting GA Parameters and Training of FES.....	150
6.8 FES Evaluation .....	152
6.9 Fuzzy Knowledge Base.....	154
6.10 Training to HIS .....	160
6.11 Evaluation of HIS .....	162
6.12 Test Networks .....	163
6.13 Plot Command Menu .....	164
6.14 Visualization .....	166
VII. ADDITIONAL STUDY:	
FUZZY TEMPORAL REPRESENTATION AND REASONING .....	171
7.1 Constructing Fuzzy Temporal Models from Data .....	172
7.2 Temporal Representation and Reasoning by Fuzzy Temporal Systems.....	174
7.3 An Algorithm for Generating Fuzzy Temporal Systems.....	178
7.4 Simulation Study.....	180
7.4.1 A Process of Hot-And-Cold Water Mixing Simulator .....	180
7.4.2 Rule Generation .....	182
7.4.3 The Resulting Linguistic Rules.....	184

Chapter	Page
VIII. CONCLUSIONS AND FUTURE RESEARCH.....	188
8.1 Concluding Remarks.....	188
8.2 Suggestions for Future Research .....	191
BIBLIOGRAPHY .....	194

## LIST OF TABLES

Table	Page
4.1 Binary-class Contingency Table .....	98
4.2 Multi-class Contingency Table .....	99
5.1 $W_P$ and $W_T$ from the ILFN Classifier for the Iris Data .....	117
5.2 The Standard Deviation, $S$ , and the Number of Patterns, <b>count</b> , from the ILFN Classifier for the Iris Data .....	117
5.3 Resulted Linguistic Labels and Their Parameters for the Iris Data .....	118
5.4 Original Fuzzy If-Then Rules for the Iris Data.....	118
5.5 Final Fuzzy If-Then Rules for the Iris Data.....	119
5.6 Simulation Results for the WBCD.....	122
5.7 ILFN Parameters for the WBCD .....	123
5.8 Resulted Linguistic Labels and Their Parameters for the WBCD.....	123
5.9 Fuzzy Expert Rules for the WBCD .....	124
5.10 Comparison Results for the WBCD Among Well-Known Methods.....	126
5.11 Breast Cancer Data .....	131
5.12 Simulation Results for the Breast Cancer Data.....	131
5.13 The Compressibility of Fuzzy Rules for the Breast Cancer Data.....	132
5.14 Lymphography Domain .....	133
5.15 Simulation Results for the Lymphography Domain .....	134
5.16 The Compressibility of Fuzzy Rules for the Lymphography Domain .....	134
5.17 Primary Tumor Domain.....	137

Table	Page
5.18 Simulation Results for the Primary Tumor Domain.....	138
5.19 The Compressibility of Fuzzy Rules for the Primary Tumor Domain .....	138
7.1 Simulated Data from Hot-And-Cold Water Mixing Process.....	181

## LIST OF FIGURES

Figure	Page
1.1 Computer-Based Medical Diagnostic System .....	1
2.1 The Conceptualized Pattern Classification Problem.....	7
2.2 Example of a Two-Dimensional Vector Pattern Space .....	8
2.3 A Linearly Separable Problem.....	9
2.4 A Nonlinearly Separable Problem .....	9
2.5 An Overlapping Problem.....	9
2.6 Flowchart Diagram of the Genetic Algorithm.....	11
2.7 The MLP Neural Network .....	13
2.8 The PNN Network Architecture .....	14
2.9 Radial Basis Function Neural Network .....	16
2.10 A Block Diagram of an Expert System.....	19
2.11 A Block Diagram of a Fuzzy System .....	25
2.12 Architecture of a Neurofuzzy Network.....	35
2.13 Loosely Coupled Models .....	39
2.14 A Trained Feedforward Neural Network.....	40
2.15 The Local Approach to Rule Extraction for a Multilayer Neural Network.....	41
2.16 Three Tightly Coupled Models: (a) A Partially Overlapped Hybrid Model; (b) and (c) Two Schemes of Embedded Hybrid Models .....	42
2.17 Taha's Hybrid Intelligent System Architecture .....	46

Figure	Page
3.1 The Architecture of the Proposed Hybrid Intelligent System.....	51
3.2 Network Architecture of the ILFN Classifier in the Supervised Learning Mode .....	52
3.3 Network architecture of the ILFN Classifier in the Unsupervised Learning Mode.....	53
3.4 The Input Subsystem of the ILFN Classifier .....	54
3.5 The Target Subsystem of the ILFN Classifier .....	57
3.6 The Decision Boundaries Among Prototypes of the ILFN Classifier .....	63
3.7 ILFN Decision Boundaries of a Three-Class, Two-Dimensional Pattern Space.....	67
3.8 A Fuzzy Expert System (FES).....	68
3.9 a) Projection of ILFN to One-Dimensional Fuzzy Sets; b) FES Grid Partition with its Parameter Projected from a Trained ILFN .....	74
3.10 Mapping from ILFN to Linguistic Rules .....	78
3.11 Crossover Operation .....	88
3.12 Mutation Operation.....	89
3.13 Hybrid System Combined from ILFN and FES .....	92
3.14 Hybrid Decision Boundaries.....	93
3.15 Conflict Decision between the ILFN and the FES.....	95
4.1 Higher Comprehensible Fuzzy System with Few Linguistic Terms .....	101
4.2 Low Comprehensible System with Too Many Linguistic Terms.....	101
4.3 A Structure of Two-dimensional Fuzzy System with Too Many Fuzzy Rules .....	102
4.4 A Structure of Two-dimensional Fuzzy System with Fewer Fuzzy Rules.....	103
4.5 A Two-Dimensional Fuzzy System with Two Conditions Per Rule.....	103



Figure	Page
4.6 A More Compact Fuzzy System with 1.67 Conditions Per Rule .....	104
4.7 Comprehensible Linguistic Terms and Complete Fuzzy Partitions .....	105
4.8 Poorly Comprehensible Linguistic Terms and Incomplete Fuzzy Partitions .....	105
4.9 Intersection of $l_{jk_1}$ and $l_{jk_2}$ .....	108
4.10 Union of $l_{jk_1}$ and $l_{jk_2}$ .....	108
4.11 a) A complete Rule Structure; b) An Incomplete Rule Structure .....	111
4.12 Antecedent Structure of a Complete Fuzzy System .....	111
4.13 Antecedent Structure of an Incomplete Fuzzy System .....	112
5.1 Scatter Plot of Sepal Width and Length Features of the Fisher's Iris Data .....	116
5.2 Scatter Plot of Petal Width and Length Features of the Fisher's Iris Data .....	116
5.3 Linguistic Rules for Breast Cancer Data .....	132
6.1 HIS Graphical User Interface.....	141
6.2 Data Preparation for the GUI.....	142
6.3 Main Menu of the GUI .....	143
6.4 Dialog for Load Data .....	143
6.5 Dialog for Setting the Number of Training and Testing Data .....	144
6.6 Training ILFN Network.....	145
6.7 ILFN Weights Appeared in The Main Window .....	146
6.8 The Trained ILFN Weights and Parameters .....	147
6.9 Evaluation of the ILFN by the Test Data.....	148
6.10 ILFN Evaluation Dialog .....	148
6.11 ILFN Performance Based on Individual Node in Count .....	149

Figure	Page
6.12 ILFN Performance Based on Individual Node in Percentage.....	149
6.13 Select GA Parameter Before Training FES and HIS.....	150
6.14 Select Train Fuzzy to Begin Training FES.....	151
6.15 Result After Training FES.....	151
6.16 Evaluation of the FES by the Test Data.....	152
6.17 Rule Quality Evaluation Dialog.....	153
6.18 Selecting Fuzzy Knowledge Base from Command Menu.....	154
6.19 Fuzzy Knowledge Base Window.....	155
6.20 Edit Feature's Names Dialog.....	156
6.21 Edit Linguistic Labels Dialog.....	156
6.22 Edit Fuzzy Rules Dialog.....	157
6.23 A New Rule Is Added in the Fuzzy Rule Base.....	158
6.24 Many Rules Are Deleted from the Fuzzy Rule Base.....	159
6.25 Fuzzy Rule Surface Viewer.....	159
6.26 Select Train HIS to Begin Training HIS.....	160
6.27 After Training HIS.....	161
6.28 Evaluation of the FES by the Test Data.....	161
6.29 ILFN Evaluation Dialog.....	162
6.30 Test Networks Command Menu.....	164
6.31 A Plot of ILFN Weight.....	165
6.32 A Plot of Initial Fuzzy Rules.....	165
6.33 A Plot of Optimized Rules.....	166

Figure	Page
6.34 Visualization Command Menu .....	167
6.35 Visualize Data .....	167
6.36 Visualize Clusters of ILFN .....	168
6.37 Visualize Fuzzy Rule Surface .....	168
6.38 Visualize Hybrid Surface .....	169
6.39 View Surface of ILFN .....	169
6.40 View Surface of FES .....	170
7.1 A Fuzzy Temporal Model.....	174
7.2 A Process of Hot-And-Cold Water Mixing Simulator .....	180
7.3 The Plot of Simulated Data from Hot-And-Cold Water Mixing Process.....	182
7.4 Linguistic Variables for Flow Rate 1 and Flow Rate 2 .....	185
7.5 Linguistic Variables for Inlet Temp1 and Outlet Temp 3.....	185
7.6 Linguistic Variables of Time Delay.....	186
7.7 Sample Temporal Representation and Reasoning .....	186

## LIST OF ACRONYMS

ART	adaptive resonance theory
ES	expert system
FES	fuzzy expert system
GA	genetic algorithm
GUI	graphical user interface
HIS	hybrid intelligent system
HIA	hybrid intelligent architecture
ILFN	incremental learning fuzzy neural network
LVQ	learning vector quantization
MDSS	medical diagnostic decision support system
MLP	multilayer perceptron
MOM	mean of maximum
MHIS	multi-module hybrid intelligent systems
PNN	probabilistic neural network
RBF	radial basis function
SHIS	single-structure hybrid intelligent system
SOM	self-organizing map
WBCD	Wisconsin breast cancer data

# CHAPTER I

## INTRODUCTION

### 1.1 Medical Diagnosis Overview

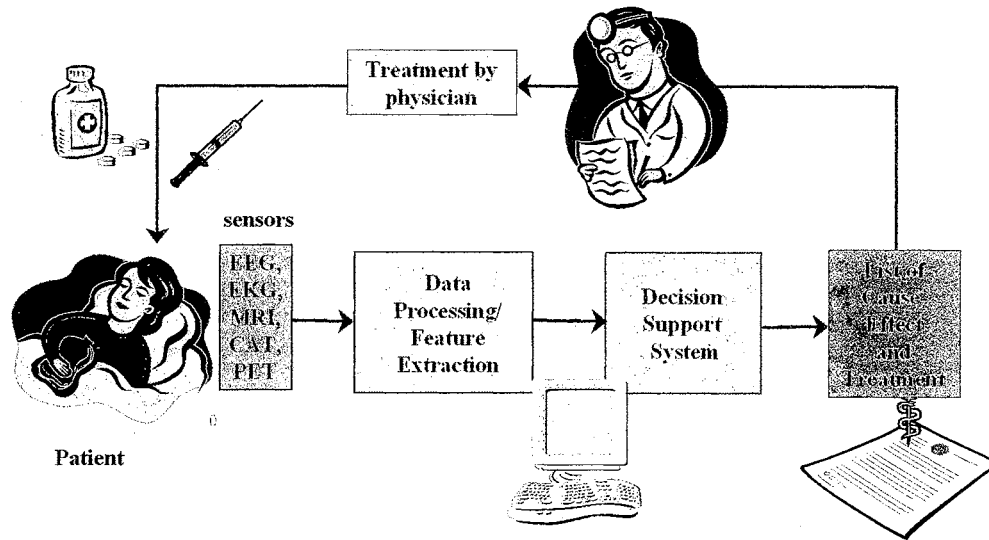


Figure 1.1: Computer-Based Medical Diagnostic System

Conventional medical diagnosis in clinical examinations highly relies upon physicians' experience. Physicians intuitively exercise knowledge obtained from symptoms of previous patients. In everyday practice, the amount of medical knowledge grows steadily such that it may become difficult for physicians to keep up with all the essential information gained. For physicians to quickly and accurately diagnose a patient, there is a critical need in the area of employing computerized technologies to assist in medical diagnosis and to access to the information related. Computer-assisted technology is certainly helpful for inexperienced physicians in making medical decisions as well as for experienced physicians in supporting complex diagnoses. Computer-assisted technology has become an essential tool to help physicians in retrieving the medical

information and making decisions in medical diagnosis [Seka97], [Dupuits98], [Makris98], [Conforti99], [Foran00], [Adlassnig01], [Economou01]. Figure 1.1 shows a diagram of a computer-based medical diagnostic system.

As shown in Figure 1.1, a computer-based medical diagnostic system consists of several units: 1) a sensory unit/data-receiving unit; 2) a data processing/feature extraction unit; 3) a decision support system or a classification unit; and 4) a user interaction unit. The sensory unit or the data-receiving unit is used to acquire data from the patient. The data can be from Electroencephalography (EEG), Electrocardiogram (EKG), Magnetic Resonance Imaging (MRI), Computerized Axial Tomography (CAT), Positron Emission Tomography (PET), or from patient's interview. The data processing or feature extraction unit is used to prepare data in a form that is easy for a decision support system or a classification unit to use. Compared to the input, the output data from the feature extraction unit is usually of a much lower dimension as well as a much easier form to classify. The decision support system or the classification unit makes decision using data from the feature extraction unit. The decision support system provides a list of cause-and-effect reasoning from the symptoms and their corresponding treatment. Finally, the physician uses the output of the decision support system to assist in the diagnosis. With direct access to a computer-based medical diagnostic system, the physicians can treat the patient promptly and can be more accurate and consistent.

## **1.2 Motivation for the Research**

A number of medical diagnostic decision support systems (MDSS) based on computational intelligence methods have been developed to assist physicians and medical

professionals. Some medical diagnosis systems based on computational intelligence methods use expert systems (ESs) [Buchana84], [Wiegerinck99], [Kovalerchuk00], [Yan00], fuzzy expert systems (FESs) [Peña99], [Walter00], [Zahan01], [Belacel01], artificial neural networks (ANNs) [Durg93], [Pattichis95], [Yao97], [West00], and genetic algorithms (GAs) [Podgorelec99], [Man00], [Peña00]. ESs and FESs use symbolic and linguistic knowledge, respectively, and are well recognized as applicable tools in medical diagnosis. Physicians and medical professionals can easily understand the decisions from ESs and FESs. However, the development of an ES or a FES for medical diagnosis is not a trivial task. It demands an intensive and iterative process from medical experts who may not be readily available. On the other hand, ANNs have been employed to learn numerical data recorded from sensory measurements, images, patient's history, or some physical symptoms. After being trained, ANNs keep knowledge in numerical weights and biases that are often regarded as a *black box* scheme. The knowledge stored in weights and biases is just a numerical representation that is used in a mapping from the input to the output. This numerical representation makes it difficult for physicians or medical professionals to understand the underlying rationale. Physicians may want to know the meaning of those numbers and how they are related to the causes and effects of symptoms. It would be easier to generate a meaningful explanation if knowledge representation is in a symbolic form. Recently, numerical weights of ANNs have been translated to symbolic/linguistic rules by using rule extraction algorithms [Setiono96], [Tan97], [Tickle98], [Taha99], [Tino99], [Mitra00], [Setiono00]. Symbolic/linguistic rules extracted are then used as a knowledge base for an ES or a FES to support physicians in making decisions [Setiono96], [Taha99], [Mitra00], [Setiono00],

[Hayashi00]. However, the resulting knowledge base is often incomplete and inefficient. It may perform poorly in unseen data.

The integration of symbolic/linguistic processing and numerical computation, or hybrid intelligent architectures, was motivated by a need to improve the accuracy of a decision-making system. [Gallant93], [Medsker94], [Goonatilake95], [Taha97], [Wermter00]. Hybrid intelligent architectures tend to be more appropriate in applications that require both numerical computation for generalization and symbolic/linguistic reasoning for explanation. It is found that hybridization between symbolic/linguistic and numerical representations can achieve higher correct classification rate as compared to either of them employed alone [Tan97], [Taha97], [Wermter00].

Most investigations on hybrid intelligent systems have focused on the accuracy and the interpretability. In a learning system, an incremental learning capability is considered an important attribute aside from its accuracy and interpretability. As for medical diagnosis, patient data grows everyday, and novel medical knowledge should be quickly incorporated into a medical diagnosis system without spending large amounts of time in the learning process. In a hybrid system, usually multilayer perceptron (MLP) neural networks are used as a numerical model that is trained by backpropagation algorithms [Taha97]. One well-known problem of the backpropagation learning algorithms is that it is difficult to employ an incremental learning feature. The standard backpropagation algorithm lacks the ability to dynamically incorporate additional nodes or connections needed during learning [Tan97], [Fu96]. In medical problems, new knowledge of diagnosis may be found after the diagnosis system has been constructed. In using the backpropagation learning algorithms, all old and new data have to be retrained



in order to update the knowledge to cover the new symptoms. With an incremental learning algorithm, the new knowledge can be learned and added to the system without retraining previously learned data [Fu96], [Carpenter92], [Yen99], [Yen01].

The contribution of this study is in the development of a pattern classifier system (i.e., a decision support system) that is concerned not only with accuracy and interpretability but also on an incremental learning concept. We propose a hybrid intelligent system (HIS) that is composed of a numerical model in the low level and a linguistic model in the higher level and is equipped with an incremental learning algorithm. The proposed system is a hierarchical integration of an incremental learning fuzzy neural network (ILFN) [Yen99], [Yen01], and a fuzzy expert system (FES). The ILFN is a self-organizing network with the ability for fast online learning. The ILFN can learn incrementally without retraining old information. The linguistic model, FES, is constructed based on knowledge embedded in the trained ILFN. The knowledge captured from the low-level ILFN can be mapped to the higher-level FES and vice versa. The system is equipped with a conflict resolution scheme to maintain consistency in decision-making. The low-level ILFN contributes fast, incremental learning while the higher-level FES offers advantages of dealing with fuzzy data. It provides easy interpretation and explanation to the decision made. A genetic algorithm (GA) is then applied to optimize the linguistic model to maintain high accuracy and comprehensibility. The resulting HIS is capable of dealing with low-level numerical data and higher-level linguistic information. After being completely constructed, the system can incrementally learn new information in both numerical and linguistic forms.

### **1.3 Organization of the Dissertation**

The rest of this dissertation is organized as follows. Literature review related to general pattern classification is discussed in Chapter II. Following this, Chapter III gives the details of the proposed hybrid intelligent system (HIS). Quantitative measures on the accuracy, comprehensibility, and completeness of a fuzzy knowledge base are discussed in Chapter IV. To demonstrate the effectiveness and efficiency of the proposed system, numerical simulations and benchmark comparisons are presented in Chapter V. Chapter VI details the development of a hybrid intelligent system graphical user interface. Additional study on fuzzy temporal representation and reasoning is described in Chapter VII. Finally, Chapter VIII provides some concluding remarks and outlines possible future research directions.

## CHAPTER II

### LITERATURE REVIEW

Pattern classification presents a fundamental solution to various problems in real world applications such as sonar detection and classification [Sabatini01], [Kundu94], image processing [Bors99], [Pan99], process control [Marcu97], [Bensaoula98], signature identification [Qi95], machinery conditional health monitoring [Filippetti2000], [Li2000], computer-assisted medical diagnosis [Chen98], [Chen2000], [Tilbury2000], [Sacha2000], and etc. The function of pattern classification is to categorize an unknown pattern into a distinct class based upon some suitable similarity measures. Thus, similar patterns are designated into the same classes, while dissimilar ones are classified into different classes.

#### 2.1 Basic Architecture of Pattern Classification

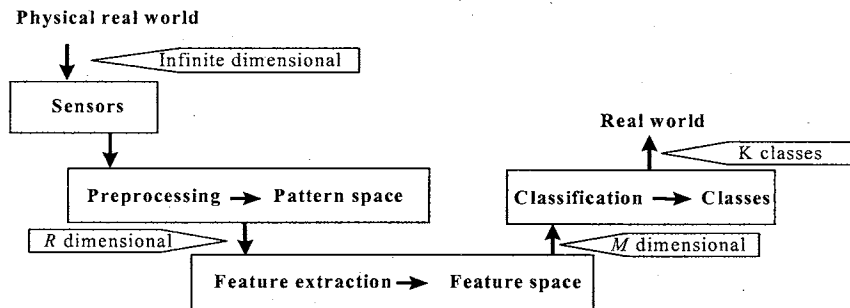


Figure 2.1: The Conceptualized Pattern Classification Problem

Figure 2.1 illustrates the framework of the pattern classification problem. The physical real world is sensed by a transducer system that feeds its data into the pattern space after a preprocessing procedure. The physical real world, or sensory system, can be characterized by a continuum of parameters that are basically infinite in dimensionality.

Transducers are used to transform signals from real environment to the pattern vector space with the dimensionality of  $R$ , typically a large value. Then a feature extractor is employed to reduce the dimension from  $R$  to a much smaller value,  $M$ , while still preserving the discriminatory features for classification expectation. Using an  $M$ -dimensional feature space, a classifier performs much faster than using an  $R$ -dimensional pattern space. Finally, in the classification space, one of  $K$  classes is chosen for a given input pattern [Andrews72].

The data that will be classified are presented into pattern classifiers by sets of measurements. Each measurement can be associated to an axis in a multidimensional space called “hyperspace.” For visualization purpose, Figure 2.2 shows a two-dimensional space with three groups, i.e., “classes,” of patterns. Figure 2.3 illustrates a linear separable problem in which a line exists to separate the two classes. Figure 2.4 demonstrates a non-linear separable problem where a straight line cannot separate the two classes. A non-linear decision boundary is needed to solve this problem. An overlapping class is depicted in Figure 2.5. Neither a linear nor a non-linear boundary can separate this problem. However, the decision can be made by using “Bayes strategy” to minimize misclassification rate for this problem.

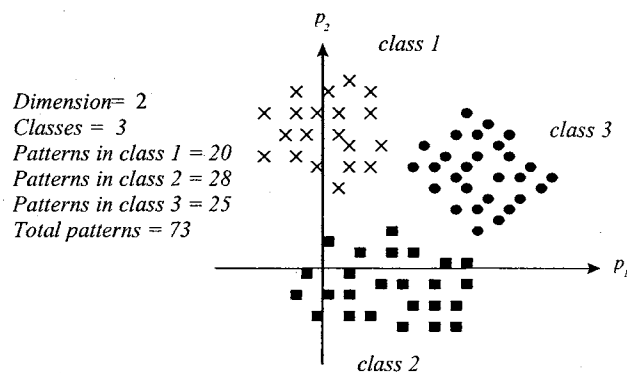


Figure 2.2: Example of a Two-Dimensional Vector Pattern Space

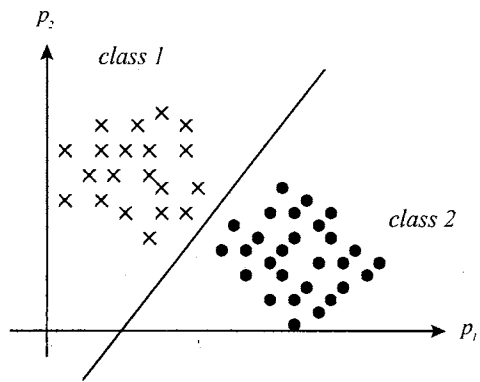


Figure 2.3: A Linearly Separable Problem

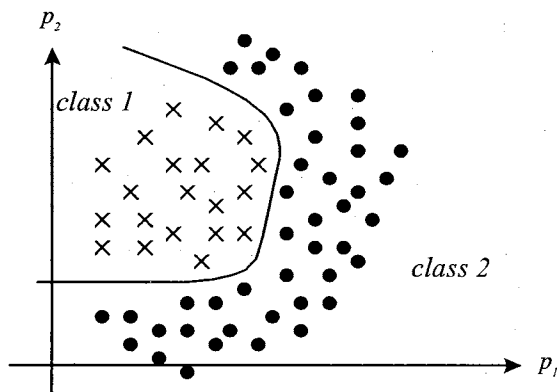


Figure 2.4: A Nonlinearly Separable Problem

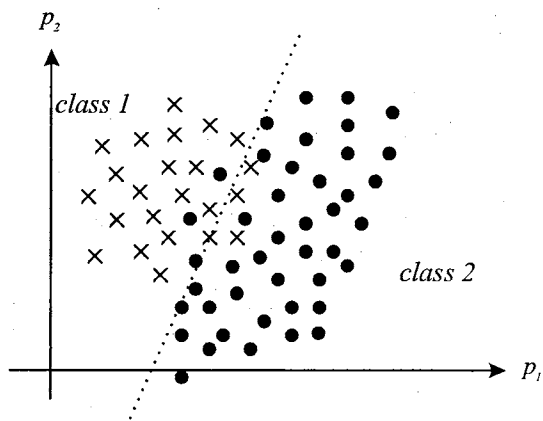


Figure 2.5: An Overlapping Problem

A large number of classification techniques have been developed to deal with pattern classification problems. Some of these classification techniques are statistical-based approaches, neural networks, expert (knowledge-based) systems, fuzzy systems, and hybrid systems, which involve the combination of two or more techniques mentioned above.

## 2.2 Statistical Classifiers

A statistical classifier is a conventional technique for pattern classification problems [Heng98], [Kumar97]. Much of the early work focused on linear classifiers [Smith68] and parametric classifiers such as Bayesian classifier [Duda73]. Because statistical approaches are usually based on the assumption that the decision problem is posed in probabilistic terms and that all of the related probability parameters are known beforehand [Duda73]. These techniques can rarely be ideally applied in practice. However, due to its effectiveness, even when approximate values are used, statistical classifiers are still widely applied (see [Wiegerinck99], [Dennis96], [Pradhan96], [Yan2000], [Nikovski2000], [Matthews2001].) Statistical approaches may limit to the simple problems in which the probabilistic values are known in *a priori* or can be reliably estimated. In many complex applications, the probabilistic values are not known beforehand. Computational intelligence methods, such as genetic algorithms, neural networks, expert systems, fuzzy systems, are motivated in order to handle more complicated applications.

## 2.3 Genetic Algorithms

The genetic algorithm [Holland75] is a stochastic search useful for optimization problems. It is motivated by the mechanisms of evolution in nature [Darwin58]. The genetic algorithm operates on populations of strings, with the string coded to represent some underlying parameter sets. Reproduction, crossover, and mutation are applied to successive string populations to create new string populations. These operators involve random number generation, string copying, and partial string exchange. Figure 2.6 illustrates a flow chart diagram of the genetic algorithm.

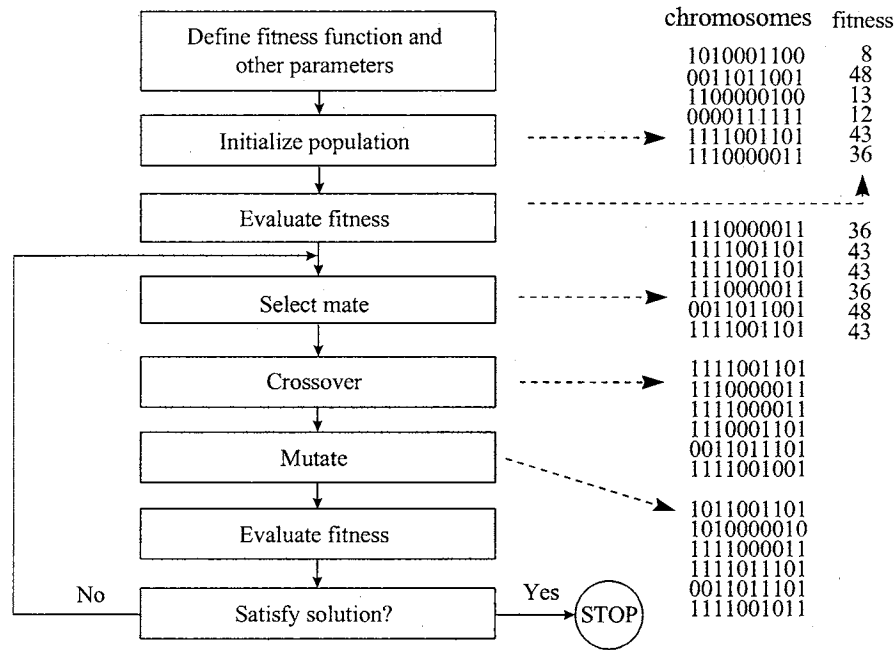


Figure 2.6: Flowchart Diagram of the Genetic Algorithm.

In practice, we can implement this genetic model of computation by having arrays of bits or characters to represent the chromosomes. Simple bit manipulations allow the implementation of crossover, mutation, and other operations. When the genetic algorithm is implemented, it usually proceeds in a manner that involves the following cycle:

evaluate the fitness of all of the individuals in the population; create a new population by performing operations such as crossover, fitness-proportionate reproduction and mutation on the individuals whose fitness has just been evaluated; discard the old population and iterate using the new population.

## 2.4 Neural Networks

A neural network is a data processing system consisting of a massive number of simple and highly interconnected processing units operated in a parallel manner. The networks are inspired by the structure and the function of the human brain. The characteristics of an artificial neural network are model-free (i.e., the model can be considered as a “black box”) and trainable systems with parallel computation. These properties are considered as benefits to many applications in the real world, including pattern classification problems.

Learning algorithms of neural networks applied to pattern classification have two main categories: supervised learning algorithms and unsupervised learning (clustering) algorithms. The use of supervised learning algorithms assumes that the input and the corresponding target pairs are known. This approach assumes that appropriate input features have been chosen and that the training data are representative of all the problem conditions. Some examples of supervised learning networks are the multilayer perceptron network (MLP) [Rumelhart86] trained by the Backpropagation algorithm (BP) [Rumelhart86], probabilistic neural network (PNN) [Specht88]-[Specht94], the learning vector quantization (LVQ) neural network [Kohonen97], and the radial basis function networks (RBFN) [Broomhead88], [Haykin94], [Hwang97], [Moody89].



On the contrary, in unsupervised learning neural networks, the input does not have a corresponding target. Since there are no target outputs available, the network distinguishes the input data into a number of clusters. The system learns to categorize the input patterns into a finite number of classes using some similarity measures. The two most used unsupervised neural networks are adaptive resonance theory networks (ART) [Carpenter87] and self-organizing maps (SOM) [Kohonen97].

### 2.4.1 Multilayer Perceptron (MLP) Neural Network

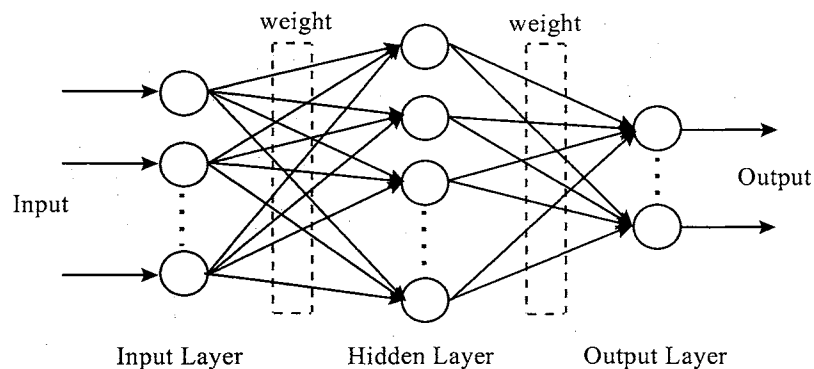


Figure 2.7: The MLP Neural Network

The MLP neural network trained by the backpropagation algorithm has been a good candidate for pattern classification problems. The MLP is a fully connected feedforward network with sigmoidal activation functions. There are many developed algorithms that are used to train the network such as the steepest descent, Newton's methods [Battiti92], conjugate gradient [Charalambous92], and Levenberg-Marquardt algorithm [Hagan94]. Rumelhart and McClelland in [Rumelhart86] present an extensive detail of the MLP network. The steepest descent backpropagation algorithm requires a long training time. In addition, similar to the other nonlinear optimization methods, there

is no guarantee of convergence to a global minimum. Especially when complex decision boundaries are required and networks have more hidden layers, the performance index becomes more complicated [Dimartino96], [Lippmann89]. Levenberg-Marquardt backpropagation is a very fast learning algorithm for training the MLP network but it requires a considerable amount of memory [Hagan94]. The architecture of the MLP network is shown in Figure 2.7.

### 2.4.2 Probabilistic Neural Network (PNN)

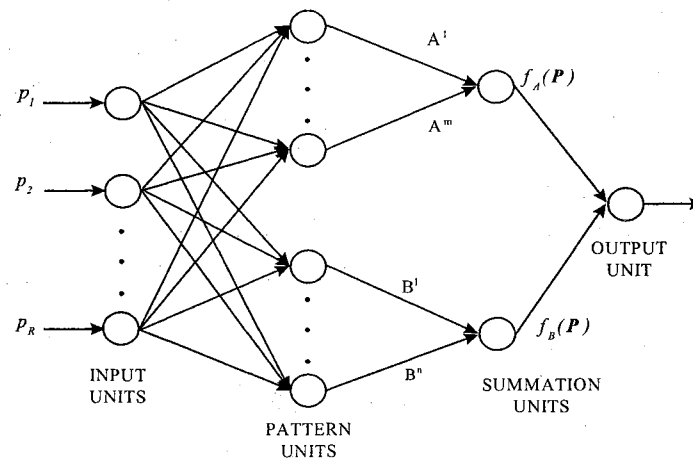


Figure 2.8: The PNN Network Architecture

The probabilistic neural network (PNN), developed by Specht in 1988, is a useful methodology for solving pattern classification problems. “Decision boundaries” which are the lines separating the different classes are formed by characteristic values from conditional probability density functions (PDF). The network is able to form complex nonlinear decision boundaries created by the Bayes strategy when given enough examples. The training speed of the PNN is faster than the MLP trained by the backpropagation algorithm to achieve the same level of generalization. On-line learning

is another advantage of the PNN; thus it is suitable to use the PNN for real-time applications. However, the PNN uses extensive memory requiring one neuron for each training pattern. Therefore, researchers have proposed various remedies to solve the memory problem, such as using clustering techniques to implement a cluster center which represents a prototype of training patterns [Specht88], [Specht94]. Figure 2.8 shows the architecture of the PNN.

### **2.4.3 Radial Basis Function (RBF) Neural Network**

Another good candidate for pattern classification is the RBF neural network [Broomhead88], [Moody89], [Haykin94], [Hwang97]. The network is a feedforward network consisting of three layers: an input layer, a hidden layer, and an output layer. Each neuron of the input layer connects to each element of an input vector. Neurons of the input layer are fully connected to neurons of the hidden layer via weights that represent the centers of radial basis functions in the hidden layer. The hidden layer has kernel functions (activation functions), usually Gaussian types, which are centered on the mean vectors of clusters or prototypes in the input space.

Training of the RBF network can proceed in two steps. First, the hidden layer is trained. Training patterns are clustered to a reasonable number of groups by using SOM clustering [Kohonen97], k-means clustering [Moody89], a successive approximation method [Linkens93], or the APC-III algorithm [Hwang94]. After the training of the hidden layer, the output layer is trained by a gradient descent method or a least mean square error method [Devijver82]. It is worth noting that both APC-III and the successive approximation method are equipped with incremental learning ability (meaning that it

learns new information without forgetting old information). It can cluster input patterns within only a single pass through all patterns. A variety of techniques for training radial basis function networks are discussed in the literature [Broomhead88], [Moody89], [Haykin94], [Hwang97]. In general, the training procedure of RBF networks requires an order of magnitude less in training time compared to the MLP trained by the backpropagation algorithms [Musavi92]. Moreover, their functions can be interpreted equivalent to a fuzzy inference system [Jang93]. The architecture of the radial basis networks is shown in Figure 2.9.

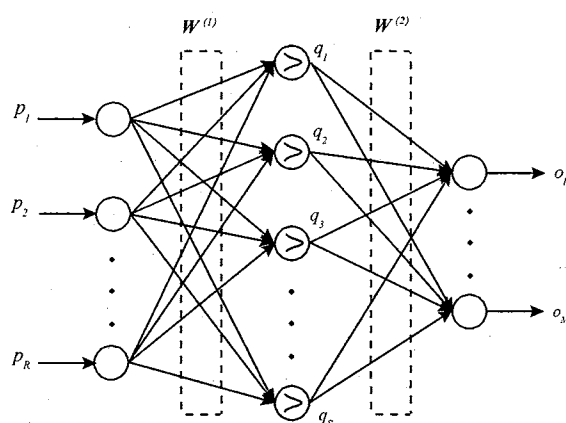


Figure 2.9: Radial Basis Function Neural Network

#### 2.4.4 Self-Organizing Map (SOM) Neural Network

The self-organizing map (SOM) neural network is an unsupervised learning algorithm that is very effective for pattern classification problems. The SOM network is usually composed of an input layer and an  $M$ -dimensional Kohonen or competitive layer. Typically the Kohonen layer is a two-dimensional layer. The weight vector is the same as the dimension of the input feature vectors. The weight vectors are randomly initialized in the feature space at the first stage. Then, the network determines the winning neuron for a

given input vector. Next, all neurons within a certain neighborhood of the winning neuron are updated moving toward the input. The moving step is controlled by the learning rate [Kohonen97]. One drawback of the SOM network is that the user needs to estimate the number of clusters in advance. For some applications, it may not be feasible to estimate the number of clusters beforehand. In addition, the choice of learning rate forces a trade-off between the speed of learning and the stability of the final weight vectors. Moreover, the SOM network needs iterative presentations of input patterns in the learning process.

A generalization of the SOM network, namely the learning vector quantization (LVQ) neural network, has been extensively used for pattern classification problems. The LVQ network uses both an unsupervised and a supervised learning algorithm. The LVQ algorithm applies a reinforced or a punished learning principle. If the current training pattern is correctly classified, the winning prototype vector will be moved closer toward the input pattern. If the input pattern is incorrectly classified, the prototype vector will be moved away from the input [Kohonen97]. A drawback of the LVQ network is that the number of clusters in the competitive layer needs to be determined *a priori*. Moreover, it needs off-line training provided that all input patterns and the corresponding targets are available. Furthermore, in the learning process, the LVQ requires iterative presentations of the input patterns.

## **2.5 Expert (Knowledge-Based) Systems**

A neural network is a distributed approach where one unit in the hidden layer corresponds to a knowledge representation. Even though this distributed approach is

advantageous in efficiency of memory usage or adaptability, it is disadvantageous in lacking the understandability for the human beings [Narazaki96]. Neural networks belong to a family of models that are based on a learning-by-example archetype in which problem solving knowledge is automatically created according to nonsymbolic or numerical data presented to the network. The knowledge, however, is represented at a subsymbolic level in terms of connections and weights. Neural networks act like a black box providing little insight into how decisions are made. They have no explicit, declarative knowledge structure that allows the representation and reasoning of decision made [Huang97].

Recently, knowledge-based systems or expert systems have gained a broad recognition as powerful tools for solving complex pattern recognition and pattern classification problems. Knowledge-based systems are finding increasing use as a practical option for problems involving human judgment. These systems operate using a set of knowledge that captures the logic needed to address the problem at hand. This knowledge may be represented in a variety of formats including production rules, semantic nets, decision trees, frames, and objects [Gonzalez93]. Most knowledge-based systems contain a vast set of knowledge that covers some problems presented to the system [Chaturvedi92]. The domain knowledge in expert systems is typically emphasized over formal reasoning methods; hence, expert systems are called “knowledge-based expert systems.” A knowledge base contains all knowledge regarding a domain of interest that has been captured through a knowledge acquisition module [Lu97]. Knowledge-based networks establish a class of artificial neural networks that use crude

domain knowledge to generate the initial network architecture that is later refined in the presence of training data [Mitra97].

Experience with rule-based expert systems has shown that the ability to generate explanations is absolutely crucial for user acceptance of computational intelligence systems. Hence, it is very important to understand the behavior of artificial neural networks. One way to generate an understanding of the behavior of artificial neural networks is to extract their problem solving knowledge in terms of rules [Huang97].

Knowledge-based system is a computational intelligence system which mimics the knowledge of human experts in order to exercise a heuristic to a given problem. The solution from knowledge-based system is essentially the same as that obtained by human experts when faced with the same problem. A knowledge-based system reflects the problem solving abilities of a human expert within a specific problem domain [Gonzalez93]. The major logical components of an expert system are a knowledge base, an inference engine, and an explanation facility.

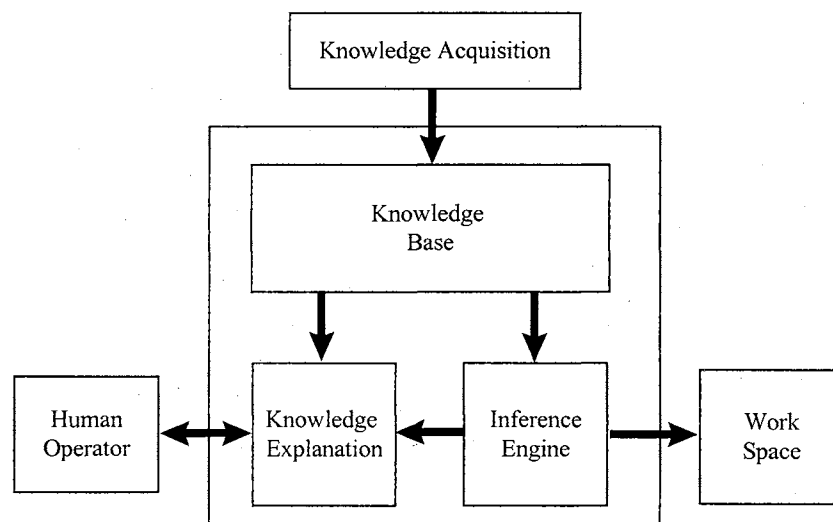


Figure 2.10: A Block Diagram of an Expert System

The knowledge base represents the most important component of a knowledge-based system. It contains the entire relevant, domain-specific, problem-solving knowledge which is collected by a knowledge engineer from human experts. The format of the knowledge refers to how this knowledge is represented internally within the knowledge-based system so that it can be used in problem solving. The knowledge representation forms widely used include logic, procedure, semantic network, production systems (if-then rules), and frames and scripts. The inference engine is the interpreter of the knowledge stored in the knowledge base. It examines the contents of the knowledge base and the data accumulated about the current problem and derives additional data and conclusions. The data structure selected for the specific form of knowledge representation determines the nature of the inference engine. The knowledge explanation facility provides the reasoning to the users. Figure 2.10 shows a block diagram of an expert system.

### **2.5.1 Knowledge Representation**

Various methods of representing knowledge within an expert system are reported in the literature. Some of the widely used methods are semantic networks, frames, and production rules.

*Semantic Networks:* A semantic network has the structure of a graph where a node represents a concept and an arc connecting the nodes represents the relationship between the concepts. A good property of semantic networks is that semantic networks permit the statement of important associations explicitly and compactly. Compared to frames and productions rules, the search time in the semantic networks is less because the nodes are



directly connected to related nodes. A drawback of semantic networks is that no standard interpretation exists for knowledge representation within semantic network. Besides, there is a high possibility of incorrect inferences drawn using semantic networks.

*Frames:* Frame is one of the knowledge representation schemes which include knowledge about a concept. A frame structure may consist of various slots. Each slot may consist of properties of a single concept. Some advantages of frames are: 1) the knowledge engineer can specify the actions that should take place when certain conditions arise during the knowledge processing; 2) inference process is speedy; and 3) frames can be made self driven. A disadvantage of frames is that frame based systems can be too complex. Furthermore, it is not easy to accommodate the new situations in frame-based systems.

*Production Rules (IF-Then Rules):* The production rules, also called “if-then rules,” are widely used in the majority of the expert systems. The rules are the If-Then-Action rules; that is if condition is met, then some action is performed. These rules may contain certainty factors. In the absence of certainty factor, the decision is assumed to be 100% confidence.

Some of the advantages of the production rules are as follows. 1) It is easy to incorporate additional knowledge, modify knowledge and eliminate knowledge since the rules are independent from each other. 2) Rules are generally “crystal clear” to humans in the sense that they use the same form of natural language. 3) There is a provision to incorporate heuristic knowledge and inexact information using uncertainty factors. 4) It is easy to incorporate the explanation facility in the expert system by simply restating the rules.

On the other hand, production rules have some disadvantages. For example, when the knowledge base grows, it may become difficult to keep track of the rules. In addition, when new knowledge is introduced to fix some problem in the knowledge base, a contradiction may be introduced.

### **2.5.2 Mycin Expert System for Medical Diagnosis**

Mycin is an expert system developed at Stanford in the 1970s. Its objective is to diagnose and recommend treatment for certain blood infections. Without Mycin, to do the diagnosis properly will involve growing cultures of the infected organism. Unfortunately this takes around 48 hours, and if doctors waited until this was complete their patient might be dead. So, doctors have to come up with quick guesses about the likely problems from the early symptoms, and use these intelligent guesses to provide necessary treatments. Mycin was developed partly in order to explore how human experts make these rough but important guesses based on partial information. However, the problem is also a potentially important one in practical terms; there are lots of junior or non-specialized doctors who sometimes have to make such a bold diagnosis, and if there is an expert tool available to help them then this might allow a more effective treatment to be prescribed. Mycin represented its knowledge as a set of IF-THEN rules with certainty factors. The following is an English version of one of Mycin's rules:

IF the infection is primary-bacteremia ,  
AND the site of the culture is one of the sterile sites,  
AND the suspected portal of entry is the gastrointestinal tract,  
THEN there is suggestive evidence (0.7) that infection is bacteroid.

The 0.7 is roughly the certainty that the conclusion will be true given the evidences. If the evidence is uncertain, the certainties of the bits of evidence will be multiplied with the certainty of the rule to give the certainty of the conclusion.

Mycin is a goal-directed system, using the *backward chaining* reasoning strategy that is the inference engine selects a possible conclusion and try to prove its validity by searching for supporting evidences. However, Mycin used various heuristics to control the search for a solution or proof of some hypotheses. These were needed both to make the reasoning efficient and to prevent the user being asked too many unnecessary questions. One strategy is to first ask the user a number of more or less preset questions that are always required and from which allow the system to rule out totally unlikely diagnoses. Once these questions have been asked the system can then focus on particular, more specific possible blood disorders, and go into full backward chaining mode to try and prove each one. This rules out a lot of unnecessary search, and also follows the diagnostic pattern of human patient-doctor interviews. The other strategies relate to the way in which rules are invoked. The first one is simple: given a possible rule to use, Mycin first checks all the premises of the rule to see if any are known to be false. If so, there is not much point using the rule. The other strategies relate more to the certainty factors. Mycin will first look at rules that have more certain conclusions, and will abandon a search once the certainties involved get below 0.2.

A dialogue with Mycin has three main stages. In the first stage, initial data about the case is gathered so the system can come up with a very broad diagnosis. In the second, more directed questions are asked to test specific hypotheses. At the end of this section a diagnosis is proposed. In the third section questions are asked to determine an

appropriate treatment, given the diagnosis and facts about the patient. This obviously concludes with a treatment recommendation. At any stage the user can ask why a question was asked or how a conclusion was reached, and when treatment is recommended the user can inquire alternative treatments if the first is not viewed as satisfactory.

## 2.6 Fuzzy Systems

The idea of fuzzy sets was originated by Zadeh [Zadeh65]. The main concept of fuzzy sets is that many problems in the real world are imprecise rather than exact. It is believed that the effectiveness of the human brain is not only from precise cognition, but also from fuzzy concept, fuzzy judgment, and fuzzy reasoning. Fuzzy systems reason with *multi-valued* sets or *fuzzy sets* (i.e., the sets of values between 0 and 1) instead of *bi-valued* sets or *crisp sets* (i.e., the sets of value of 0 and 1). An advantage of fuzzy classification techniques lies in the fact that they provide a *soft decision*, a value that describes the degree to which a pattern fits within a class, rather than a *hard decision*, in which a pattern either belongs to a class or not. In later development, fuzzy systems are successfully used to handle many applications in the real world such as control systems and pattern classification problems. Some well-known fuzzy systems are fuzzy-rule-base methods [Ishibuchi92], fuzzy c-means [Bezdek81], fuzzy k-nearest-neighbor [Bezdek86], [Keller85], and fuzzy decision tree [Chang77].

A typical fuzzy logic system has four components: a fuzzifier, a fuzzy rule base, an inference engine, and a defuzzifier. The function of the fuzzifier is to determine the degree of membership of a crisp input in a fuzzy set. The fuzzy rule base is used to

represent the fuzzy relationships between input-output fuzzy variables. The output of the fuzzy rule base is determined based on the degree of membership specified by the fuzzifier. The inference engine calculates the rule's conclusion based on its membership degree. Optionally, if needed, a defuzzifier is used to convert outputs of the fuzzy rule base into crisp values. Figure 2.11 illustrates a block diagram of a fuzzy system.

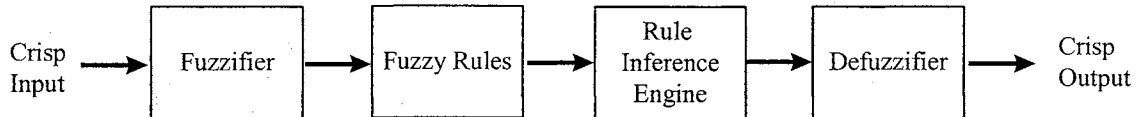


Figure 2.11: A Block Diagram of a Fuzzy System

### 2.6.1 Structure of Fuzzy Rules

A fuzzy rule is the basic unit for capturing knowledge in many fuzzy systems. A fuzzy rule has two components: 1) an IF-part or the *antecedent* and 2) a THEN-part or the *consequent* (i.e., If <antecedent> THEN <consequent>). The antecedent describes a condition, and the consequent describes a conclusion that can be drawn when the condition holds. The most popular models of fuzzy systems are the Mamdani models [Mamdani74] and the Takagi-Sugeno (TS) models [Takagi85]. Brief discussions of fuzzy systems for both Mamdani models and TS models are given in the following subsections.

### 2.6.2 Mamdani Models

Mamdani fuzzy-rule based systems [Mamdani74] consist of a linguistic description in both the antecedent parts and the consequent parts. Each rule is a description of a condition-action statement that may be clearly interpreted by the users.

To describe a mapping from input  $U_1 \times U_2 \times \dots \times U_n$  (where  $\times$  is the Cartesian product) to output  $W$ , the linguistic rule structure of Mamdani models is as follows:

$$R_i: \text{IF } x_1 \text{ is } A_{i1} \text{ and } \dots \text{ and } x_n \text{ is } A_{in} \text{ THEN } y \text{ is } C_i, i = 1, \dots, L \quad (2.1)$$

where  $L$  is the number of fuzzy rules,  $x_j \in U_j, j = 1, 2, \dots, n$ , are the input variables,  $y$  is the output variable, and  $A_{ij}$  and  $C_i$  are the linguistic variables or fuzzy sets for  $x_j$  and  $y$  respectively.  $A_{ij}$  and  $C_i$  are characterized by membership functions  $m_{A_{ij}}(x_j)$  and  $m_{C_i}(y)$ , respectively. Assuming normalized membership values  $m_i$ , and denoting  $\mathbf{W} = \{w_1, w_2, \dots, w_n\}$  as the finite set of possible normalized output values, the resulting output,  $\hat{y}$ , can be obtained by applying defuzzification methods as follows.

*Center of Gravity (COG)*: The COG is the most popular defuzzification technique. The COG method takes into account the entire possibility distribution in calculating its representative point. Alternatively, we can view the COG method as calculating a weighted average, where  $m_i(x)$  serves as the weight for value  $x$ .

$$\hat{y} = \frac{\sum_{i=1}^n m_i(x) w_i}{\sum_{i=1}^n m_i(x)} \quad (2.2)$$

*Mean of Maximum (MOM)*: The MOM method calculates the average of those output values that have the highest possibility degrees or the center of gravity of the area under the maxima of fuzzy output.

$$\hat{y} = \frac{\sum_{i \in M} w_i}{|M|} \quad (2.3)$$

where  $M = \{ i \mid m_i = \max( m_1, \dots, m_n ) \}$ .

### 2.6.3 Takagi-Sugeno (TS) Models

Instead of working with linguistic rules as in Mamdani models [Mamdani74], Takagi, Sugeno, and Kang [Takagi85], [Sugeno88] proposed a new model based on rules where the antecedent was composed of linguistic variables, while the consequent was represented by a function of the input variables. The most usual form of these kinds of rules is the one shown in the following, in which the consequent constitutes a linear combination of the variables involved in the antecedent:

$$R_i: \text{IF } x_1 \text{ is } A_{i1} \text{ and } \dots \text{ and } x_n \text{ is } A_{in} \text{ THEN } y_i = b_{i0} + b_{i1}x_1 + b_{i2}x_2 + \dots + b_{in}x_n, \quad (2.4)$$

where  $x_j, j = 1, \dots, n$ , are the system input variables,  $y_i$  is the output variable, and  $b_{ij}, j = 0, 1, \dots, n$ , are the numerical constant parameters.  $A_{i1}, \dots, A_{in}$  are linguistic labels associated in the form of fuzzy set. The entire rule base consists of  $L$  rules:  $R = \{ R_i \mid i = 1, 2, \dots, L \}$ . The aggregated output of the model,  $\hat{y}$ , is calculated by taking the weighted average of the rule consequents:

$$\hat{y} = \frac{\sum_{i=1}^L \phi_i y_i}{\sum_{i=1}^L \phi_i} \quad (2.5)$$

where  $\phi_i$  is the degree of activation of the  $i$ th rule:

$$\phi_i = \prod_{j=1}^n m_{A_{ij}}(x_j), \quad i = 1, 2, \dots, L, \quad (2.6)$$

and  $m_{A_j}(x_j) : \mathfrak{R} \rightarrow [0, 1]$  is the membership function of the fuzzy set  $A_{ij}$  in the antecedent of  $R_i$ .

The inference performed by the TS model is an interpolation of the entire relevant linear model. The degree of relevance of a linear model is determined by the degree of the input data belonging to the fuzzy subspace associated with the linear model. These degrees of belonging become the weight in the interpolation process.

#### 2.6.4 Constructing Fuzzy If-Then Rules

Fuzzy linguistic rules are usually constructed by employing knowledge from human experts in a problem domain. However, experts in a particular problem are not always readily available for constructing a fuzzy rule base. Even if there are experts available, it is very time-consuming to derive fuzzy linguistic rules. To cope with this problem, automatic construction of fuzzy rules from numerical data has been extensively investigated.

With the emerging technologies of computational intelligence, such as artificial neural networks (ANNs), genetic algorithms (GAs), and other learning algorithms, the generation of fuzzy rules has been improved. Clustering algorithms such as c-means, fuzzy c-means, and self-organizing feature map algorithms have been employed to assist fuzzy rule generation. In clustering approaches, training patterns are clustered into subspaces that are mapped to fuzzy rules. ANNs are also applied to help in constructing fuzzy rules. Linguistic rules can be extracted from the trained ANNs [Fu94].

Ishibuchi *et al.* [Ishibuchi95] proposed the concept of distributed fuzzy if-then rules where all fuzzy if-then rules corresponding to several fuzzy partitions were



simultaneously employed in fuzzy inference. A disadvantage of this method is that an excessive number of rules are generated especially for high-dimensional pattern spaces. Abe *et al.* [Abe95] proposed a method to construct fuzzy rules directly from numerical data. In this method, activation hyperboxes are used to define fuzzy regions. This method is unclear to human users since fuzzy linguistic variables are not being used. In addition, Ishibuchi *et al.* [Ishibuchi97] used the genetic algorithm to derive fuzzy rules from numerical data. Using this method, a compact rule set is obtained since several unnecessary fuzzy partitions are removed by the genetic operations. Nozaki *et al.* [Nozaki96] studied a method for selecting significant fuzzy rules by pruning unnecessary ones. Their method employs the error correction-based learning procedure and the concept of forgetting. Russo [Russo98] developed FuGeNeSys, a method based on the genetic algorithm to model fuzzy systems from input-output data. Tang *et al.* [Tang98] proposed a scheme to obtain optimal fuzzy subsets and rules derived from the use of genetic algorithms. Wang *et al.* [WangHong98] proposed a methodology to construct fuzzy knowledge by integrating multiple sources using genetic algorithms. Ishibuchi *et al.* [Ishibuchi99] proposed a method to construct a fuzzy classifier using antecedent fuzzy sets of each fuzzy if-then rule and pre-specified linguistic values with fixed membership functions while the consequent class and the grade of certainty of each fuzzy if-then rule are determined by a simple heuristic procedure. The genetic algorithm was used as an optimization method to reduce the number of rules. Using the “don’t care” membership function, this method is claimed to handle high-dimensional problems very effectively. Figueiredo and Gomide [Figueiredo99] proposed a method to extract fuzzy rules from a neurofuzzy network. Chow *et al.* [Chow99] used a fuzzy neural network to extract

knowledge by heuristic constraint enforcement. Jin *et al.* [Jin99] proposed evolution strategy to generate a fuzzy rule set which is considered to be flexible, complete, consistent, and compact. Cordon and Herrera [Cordon99] proposed a method to construct TS fuzzy rule-based systems using a two-stage evolutionary process. In the first stage, a preliminary TS-type knowledge base is obtained from the training data. The second stage performs a hybrid genetic local search to obtain an optimal global solution. Shi *et al.* [Shi99] proposed a method to design fuzzy systems by using evolutionary algorithms. In their study, the shapes and types of membership functions and the fuzzy rule set including the number of rules used are evolved. In addition, the genetic parameters of the evolutionary algorithm are adapted via a fuzzy system.

## 2.7 Hybrid Intelligent Systems

The computational intelligence methods such as expert systems, fuzzy systems, neural networks, and genetic algorithms, are complementary to each other in terms of their advantages and disadvantages. Expert systems provide symbolic interpretation and explanation capability. Fuzzy systems offer the important concept of computing with words that deal with imprecision and information granularity. In addition, fuzzy systems have a benefit on approximate reasoning. Neural networks have the capability of learning and adaptation. Genetic algorithms make use of an evolutionary search that is essential for optimization. A lot of research projects have been investigating on how to combine two or more methods in order to overcome limitations of each individual system alone. This synergism can be exploited to achieve benefits such as accuracy, interpretability, robustness, and low solution cost. A *hybrid intelligent system* is a system resulting from

the combination of two or more intelligent systems into the same framework. A hybrid intelligent system is designed to improve system interpretability or performance over that of each individual technique used alone.

Earlier research proposals for classifying hybrid systems can be found in many publications [Medsker94], [Hilario94], [Goonatilake95], [Taha97], [McGarry99], [Wermter00]. Medsker has defined the classes of hybrid system in three categories: loosely, tightly, and fully integrated systems [Medsker94]. In loosely coupled systems, the communication between the modules is performed by shared files, while tightly integrated systems and fully integrated systems use shared memory structures.

Hilario has proposed two hybrid classifications for integration: the unified approaches and hybrid approaches [Hilario94]. The unified approaches use neural networks to implement all the processing activities including a symbolic one. The hybrid approach integrates separate symbolic and neural elements. In the hybrid approaches, Hilario uses four distinct classes based on the flow of data between the modules as well as two degrees of coupling, i.e., loosely coupled and tightly coupled.

Goonatilake and Khebbal [Goonatilake95], based on neural networks, rule-based systems, genetic algorithms, and neuro-fuzzy logic, have defined hybrid systems into three terms: function-replacing hybrids, intercommunicating hybrids, and polymorphic hybrids. In the function replacing hybrids, a basic of one technology is replaced by another technology, for example the use of genetic algorithms to train neural networks. The intercommunicating hybrids have modular structures. Different modules use different kind of systems and they are basically independent. The function replacing hybrids can be considered as a sub-class within the intercommunicating class. The

function-replacing hybrids use a neural network to replace the inferencing engine of an expert system. The intercommunicating class is a redundant term of the function-replacing hybrids. In the last term, the polymorphic hybrids base on one technology but achieve the functionality of different processing techniques [Goonatilake95].

Taha has classified hybrid intelligent systems that are the combination of an expert system and a neural network into four groups: standalone models, transformational models, tightly coupled models, and fully coupled models [Taha97]. Standalone hybrid intelligent models have two separate components, an expert system and an artificial neural network, where there is no interaction between them. Transformational models are sequential in their operational nature, starting up with one component and ending with the other one. (Standalone hybrid models and transformational models are also called loosely coupled models.) Tightly coupled models refer to the hybrid systems that the two components use part but not all of their internal data structure to communicate instead of using external data files. Fully coupled models represent hybrid systems of dual nature that the architecture can be viewed as an expert system or as a neural network architecture.

By focusing on those hybrid systems using two elements, namely rule-based components and neural networks, in [McGarry99] and [Wermter00], hybrid neural systems can be classified into three groups: unified hybrid systems, transformational hybrid systems, and modular hybrid systems. The unified hybrid systems use all processing activities implemented by neural network elements. The transformational hybrid systems consist of insertion modules, rule extraction modules, and rule refinement module within the framework of a neural network system. The modular hybrid systems

are comprised of several neural network and rule-based modules which can have different degrees of coupling and integration [MacGarry99], [Wermter00].

Based on the classification schemes available in the existing literature, some terminologies defined are respectively inconsistent. Some hybrid systems cannot be classified into any scheme; the existing schemes do not cover all the hybrid systems. The existing schemes focused only on the combination of expert system and neural networks. Some of the existing schemes do not include fuzzy neural network, neuro fuzzy network, fuzzy genetic systems, and neuro genetic systems as hybrid architectures. In this research, a new taxonomy for the hybrid intelligent systems is proposed to classify the combination of two or more methods of computational intelligence including expert systems, fuzzy systems, neural networks, and genetic algorithms.

The hybrid intelligent systems on which most researches are currently being done may be classified into two main groups: *single-structure hybrid intelligent systems (SHIS)* and *multi-module hybrid intelligent systems (MHIS)*. The details of the two groups of hybrid intelligent systems are discussed in the following subsections.

### **2.7.1 Single-Structure Hybrid Intelligent Systems (SHIS)**

Single-structure hybrid intelligent systems (SHIS) are hybrid systems that have only one structure to represent concepts. SHIS are self-contained and independently operating without links to other systems or modules. The integrated system cannot be distinguished between the two techniques that have been combined. Examples of SHIS are fuzzy neural networks, neurofuzzy systems, fuzzy genetic systems, and neuro-genetic systems.

### 2.7.1.1 Fuzzy Neural Networks

In general, neural networks have good learning abilities but are not suitable for representing symbolic or qualitative knowledge, while fuzzy systems are good at this. A drawback of fuzzy systems is that it has no learning abilities. However, fuzzy systems and artificial neural networks share some similar properties. Both of them are able to map the naturally nonlinear relation of the input-output without a precise mathematical model between the input and output variables. Much research combines the fuzzy set theory [Zadeh65] with some neural network architectures to address the deficiencies and to enhance the performance of each individual technique. The combination of neural network and fuzzy logic forms a *fuzzy neural network* or a *neuro-fuzzy network*.

A growing number of researchers have designed and examined various forms of fuzzy-neural or neuro-fuzzy networks. The idea is to merge the capabilities of model-free and trainable systems, parallel computation, and noise tolerance of neural networks with the ability of dealing with imprecise situations from the fuzzy set theory. The integration of neural networks and the fuzzy set theory results in a classifier that borrows useful properties from both neural networks and fuzzy sets. The combination of neural networks and fuzzy sets forms a hybrid network that handles pattern classification problems very effectively and efficiently. Because of their massive parallel computational units, neural networks have the advantage of fast computation so that it is possible to process real time estimation of extensive information. The benefit of fuzzy systems lies in their ability to handle the imprecise data usually experienced in real world problems [Zadeh65]. As a result of using the fuzzy logic method, some initial experiences and knowledge can induce some rules directly from the original data. On the other hand, using the learning

capability of neural networks to tune some fuzzy logic parameters, for example the membership functions and the weights of aggregation and defuzzification, the efficiency of function approximating will be largely improved. Fuzzy neural networks have shown to be very effective in dealing with realistic problems in real life.

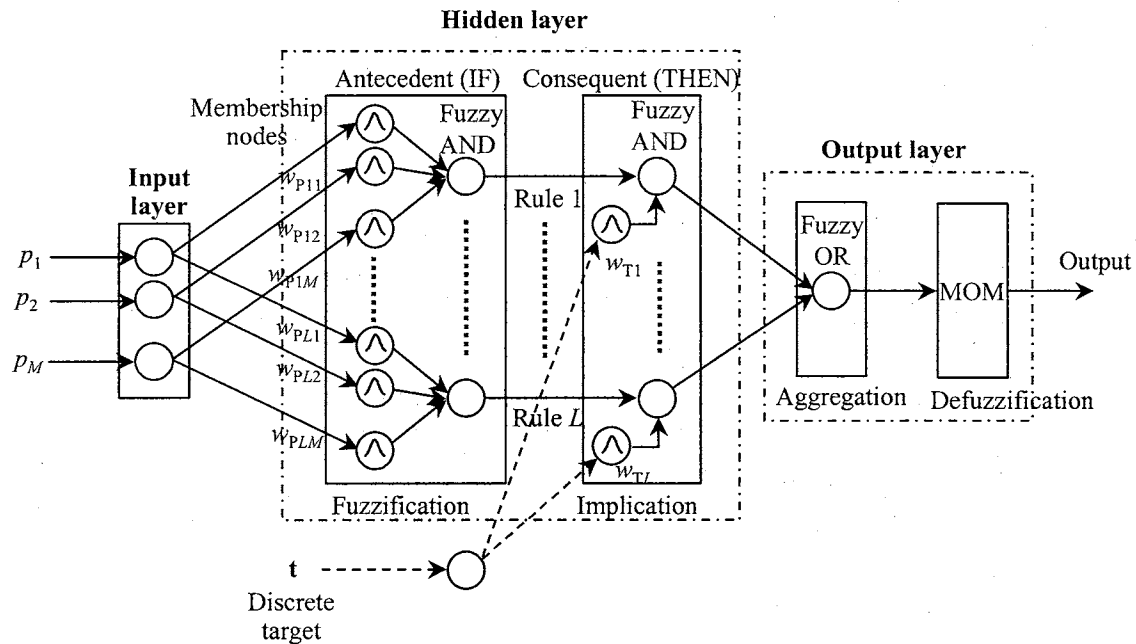


Figure 2.12: Architecture of a Neurofuzzy Network [Meesad00]

Some representative examples of fuzzy neural networks and neural-fuzzy systems for pattern classification problems are: neural-network-based fuzzy classifier [Uebele96], neuro-fuzzy system [Vuorimaa95], adaptive neural fuzzy inference system (ANFIS) [Jang93], on-line self-constructing neural fuzzy inference network (SONFIN) [Juang98], fuzzy min-max neural network [Simpson92], fuzzy ART neural network [Carpenter91], fuzzy ARTMAP neural network [Carpenter92], Gaussian ARTMAP neural network [Williamson96], RBF fuzzy ARTMAP neural network [Tontini96], and Neurofuzzy Network [Meesad00]. An example of a neural fuzzy network is shown in Figure 2.12. The details of the neurofuzzy network shown in Figure 2.12 can be found in [Meesad00].

### **2.7.1.2 Neuro-Genetic Systems**

The use of evolutionary computation for neural network designs has been growing. Several different and possibly synergistic approaches have been proposed. Most of the applications regard the use of evolutionary approaches (such as genetic algorithms) as a means to learn connection weights. Some applications also involve learning of network topology. Some applications use genetic algorithms to define the parameters of a learning algorithm that will be used in the training phase. Examples of neuro-genetic systems or evolutionary neural networks can be found from [Angeline94], [Maniezzo94], [Huang97b], [Liu99], [Chen99].

### **2.7.1.3 Fuzzy Genetic Systems**

The optimization abilities of genetic algorithms are used to develop the optimal set of rules to be used by a fuzzy inference engine and to optimize the choice of membership functions. A particular use of this system is in fuzzy classification systems, where the linguistic values of the attributes of an object enable the object to be classified. The most difficult part of building a system like this is to find an appropriate set of fuzzy rules. The most direct approach is to obtain knowledge from experts and translate this into a set of fuzzy rules. However, aside from the time-consuming nature of this solution, experts may be unable to extract their knowledge into an accurate linguistic form. A second approach is to obtain the fuzzy rules through machine learning, whereby the knowledge is automatically extracted or deduced from sample cases.

A fuzzy genetic algorithm is a directed random search over all fuzzy subsets of an interval, and has features that make it applicable to solving this problem. It is capable of



creating the classification rules for a fuzzy system where objects are classified by linguistic terms. Coding the rules genetically enables the system to deal with multi-value fuzzy logic, and is more efficient as it is consistent with numeric coding of fuzzy examples. The training data and randomly generated rules are combined to create the initial population, giving a better starting point for reproduction. Finally, a fitness function measures the strength of the rules, balancing the quality and diversity of the population. The use of the genetic algorithm to construct fuzzy rules can be found from [Cordon99], [González99], [Jin00], [Nawa99], [Tang98].

### **2.7.2 Multi-Module Hybrid Intelligent Systems (MHIS)**

Multi-module hybrid intelligent systems (MHIS) usually composed of two or more intelligent systems that can be easily distinguish from each other. Since the brain has not only a neuronal structure but has the capability to perform symbolic reasoning [Wermter00], an MHIS mimics the human brain to have two main modules: a low-level non-symbolic (or numerical) module and a higher-level symbolic (or linguistic) module. The low-level numerical module represents neuronal structure, while the higher-level represents symbolic reasoning. Those hybrid systems that use rule-based components and neural networks are classified to MHIS. A SHIS can be considered as a sub-module or a component of MHIS. For example, a neurofuzzy network can function as a low-level module, and a fuzzy genetic system can function as a higher-level linguistic module.

Based on their functionality and interconnectivity, MHIS can be classified into four sub-groups: unified architectures, loosely coupled architectures, tightly coupled architectures, and fully integrated architectures.

### 2.7.2.1 Unified Architectures

The unified architectures consist of those systems that have all processing activities implemented by neural network elements. The motivation for this area of research is that neural networks can perform certain forms of rule-based processing. Neural networks are used both in low-level numerical representation and higher-level symbolic representation. There can be two types of representations: 1) *local connectionist architectures* containing one distinct node for representing each concept; and 2) *distributed neural architectures* comprising of a set of non-exclusive, overlapping nodes for representing each concept. The use of local and distributed representations allows the generalization capabilities of neural networks to be supported by the ability to assign conceptual meanings to individual neurons or groups of neurons [McGarry99], [Wermter00]. Some examples of unified hybrid architectures are CONSYDERR [Sun95], RUBICON [Samad92], and SC-NET [Hall92],

### 2.7.2.2 Loosely Coupled Architectures

Loosely coupled architectures [Taha97], [McGarry99], [Wermter00] can be separated into two groups: standalone models and transformational models. Standalone models have two separate components, and an expert system and a neural network. There is no interaction or shared data structure between the two systems. Each network of these models has the unique behavior: the reasoning and explanation of expert systems and the generalization and adaptability of neural networks. Figures 2.13 (a) and (b) show loosely couple models of hybrid intelligent systems. Figure 2.13 (a) has two independent outputs

that can be compared with each other while the output of Figure 2.13 (b) is the combined decisions of the expert system and the neural network modules.

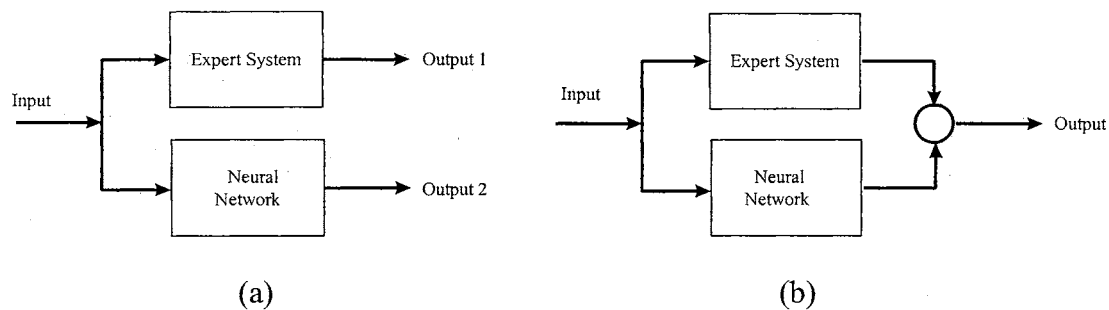
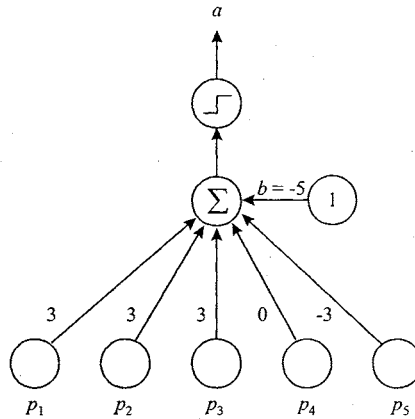


Figure 2.13: Loosely Coupled Models

Transformational models have similar characteristics to the standalone models where the expert system and the neural network modules do not share any of their internal data structure. However, transformational models are sequential in their operational nature. A transformational model usually starts up with one component and ends with the other one. The interaction between the two components of transformational models takes place by passing the output of the first module to the second one for further processing to get the benefits of the latter's unique features [Taha97], [McGarry99], [Wermter00].

Rule extraction from trained neural networks is a kind of transitional hybrid models in the category of loosely coupled architectures. The goal of rule extraction is to interpret knowledge of trained artificial neural networks into a human comprehensible manner. There are an increasing number of techniques that pursue to explain the behavior of trained artificial neural networks by extracting rules from the networks. The field of rule extraction is expanding to include techniques for connectionist knowledge representation and connectionist explanation based generalization and methods for

describing the behavior of artificial neural networks [Andrews95]. In the next section, a basic rule extraction technique is given. The basic rule extraction reviewed here is a rule extraction technique for trained feedforward artificial neural networks.



extracted rules:

$$\begin{aligned}
 & p_1 \wedge p_2 \wedge p_3 \rightarrow a \\
 & p_1 \wedge p_2 \wedge \neg p_5 \rightarrow a \\
 & p_1 \wedge p_3 \wedge \neg p_5 \rightarrow a \\
 & p_2 \wedge p_3 \wedge \neg p_5 \rightarrow a
 \end{aligned}$$

Figure 2.14: A Trained Feedforward Neural Network

Figure 2.14 shows the task of rule extraction from a very simple network. This one-layer perceptron network has five Boolean inputs and one Boolean output. A finite set of symbolic if-then rules can be extracted since there are a finite number of possible input vectors. The symbolic rules specify conditions on the input features that guarantee a given output state. We assume that the value “false” for a Boolean input feature is represented by an activation of “0,” and the value “true” is represented by an activation of “1.” Also we assume that the output unit employs a threshold function to compute its activation:

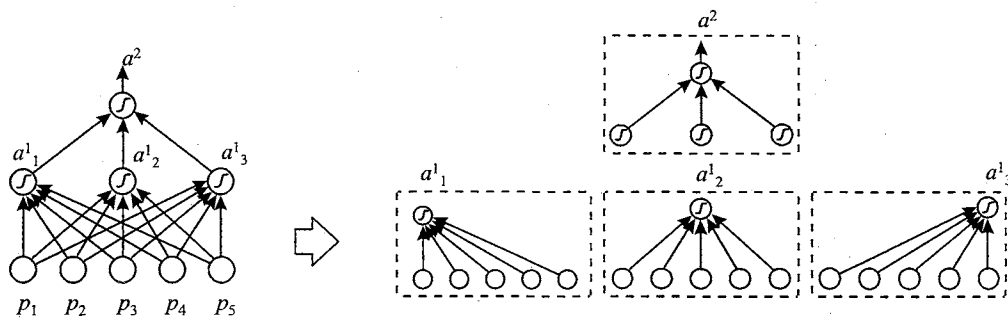
$$a = \begin{cases} 1 & \text{if } \sum_i w_i p_i + b > 0 \\ 0 & \text{otherwise} \end{cases} \quad (2.7)$$

where  $a$  is the activation of the output,  $p_i$  is the input,  $w_i$  is the weight for the  $i$ th input to the output unit, and  $b$  is the bias or threshold parameter for the output.

Figure 2.15 illustrates three conjunctive rules which describe the most general conditions under which the output unit has an activation of unity. Consider the rule:

$$p_1 \wedge p_2 \wedge p_3 \rightarrow a$$

This rule states that when  $p_1$  is true and  $p_2$  is true and  $p_3$  is true, then  $a$  is true (i.e., the output unit  $a$  will have an output activation of “1”).



extracted rules:

$$\begin{aligned}
 a_1^1 \vee a_2^1 \vee a_3^1 &\rightarrow a^2 \\
 p_1 \wedge p_2 &\rightarrow a_1^1 \\
 p_2 \wedge p_3 \wedge p_4 &\rightarrow a_2^1 \\
 p_5 &\rightarrow a_3^1
 \end{aligned}$$

Figure 2.15: The Local Approach to Rule Extraction for a Multilayer Neural Network

Whenever a neural network is used for a classification problem, there is always an implicit decision procedure that is used to decide which class is predicted by the network for a given case [Craven96]. In Figure 2.15, the decision procedure was simply to predict  $a^2 = \text{“true”}$  when the activation of the output unit was “1,” and to predict  $a^2 = \text{“false”}$  when it was “0.” In case of using logistic transfer function instead of a threshold function at the output unit, then the decision procedure might be to predict  $a^2 = \text{“true”}$  when the

activation exceeds a specified value, say 0.5. More details on this basic concept to extract rule from a trained artificial neural network can be found in [Craven96].

### 2.7.2.3 Tightly Coupled Architectures

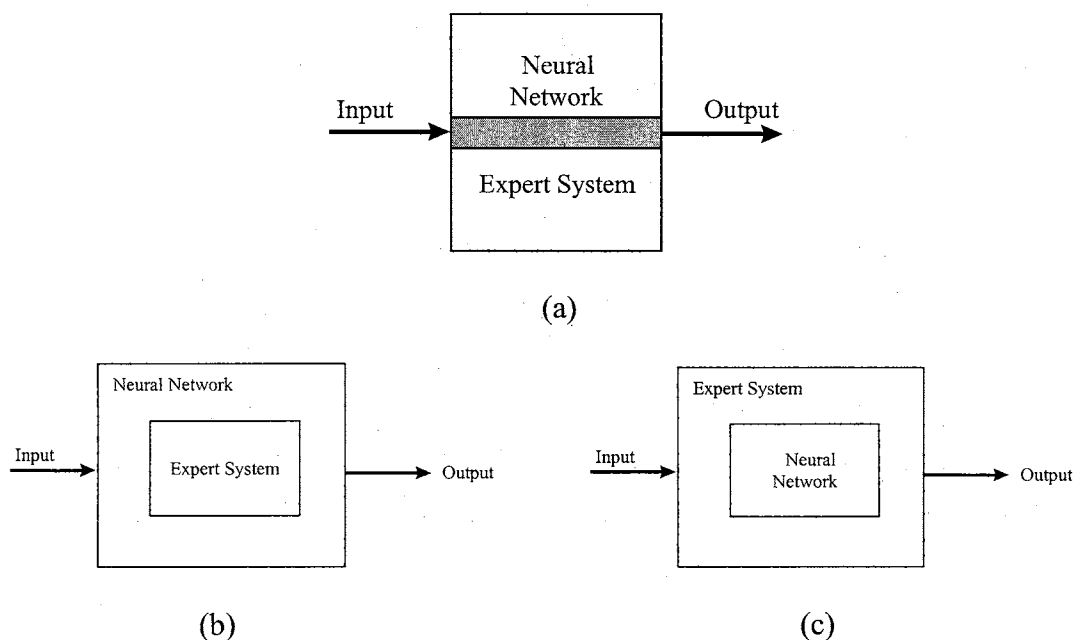


Figure 2.16: Three Tightly Coupled Models: (a) A Partially Overlapped Hybrid Model; (b) and (c) Two Schemes of Embedded Hybrid Models

In tightly coupled architectures [Medsker94], [Hilario94], [Taha97], [McGarry99], [Wermter00], the two components of the models partly share their internal data structure for communication instead of using external data. Interaction speed between the two components increases. Figures 2.16 (a), (b), and (c) shows tightly coupled models of hybrid expert system. Figure 2.16 (a) is a partially overlapped architecture where the two components interact through part of their internal data structures. Figure 2.16 (b) and (c) are embedded architecture where one of the two components is embedded inside the other [Pal92].

#### **2.7.2.4 Fully Integrated Architectures**

Fully integrated architectures represent hybrid systems of dual nature [Taha97]. The architecture can be viewed as an expert system or as a neural network architecture and still have the unique features from both networks. Fully coupled models share all the internal data structure and the knowledge representation of both components. Input symbols of the expert system can be used as input nodes to the neural network and the output nodes of the neural network can be viewed as the output decisions of the expert system. Both components can be converted to each other via a mapping mechanism. Fully integrated models have the capability to combine the complementary features of the symbolic and non-symbolic paradigms. The system designer can build such knowledge representation that can be utilized by both modules and can come up with an efficient mapping mechanism that can translate the expert system to neural network or vice versa. The resulting efficiency and robustness of the hybrid learning system can be further increased. In addition, the overall system performance can be optimized [Pal92].

#### **2.7.3 Existing Hybrid Intelligent Systems**

There are many existing hybrid architectures that combine two or more computational intelligence methods into the same framework. The following sub-sections briefly discussed a few methods.

##### **2.7.3.1 Knowledge-Based Artificial Neural Network**

Knowledge-based artificial neural network (KBANN) [Towell94] is a refinement system that translates a rule base into a neural network and then refines it using

backpropagation. The translation into a neural network proceeds in a straightforward manner. First, a logical circuit is created using the AND-OR graph of the theory, and the weights of the units in the network are set to simulate AND and OR gates. In addition, all remaining features in the data are added to the input layer. The network is then fully connected by adding low-weighted links from every node in layer  $n$  to every node in layer  $n+1$ .

Once built, the network is trained using backpropagation. To help minimize the size of the network, weight-decay [Hinton96] is utilized. By adjusting each weight in the network slightly towards zero after each weight update, interconnection weights that are not contributing to the network are eliminated.

After training, symbolic rules can be extracted from the network. By analyzing the weights of the incoming links, each unit is translated into a set of M-of-N rules, that are satisfied if at least M of their N antecedents are true. The resulting rule base is generally much simpler than the revised network; however, there is no guarantee that the two representations are semantically equivalent.

### **2.7.3.2 NeuroRule**

NeuroRule is a rule extraction algorithm proposed by Setiono [Setiono96]. It is an algorithm that extracts rules from trained feedforward neural networks with a single hidden layer. Two key components of this algorithm are a network pruning method and a hidden unit-clustering algorithm. An effective pruning algorithm removes the redundant connections and units from the network. A robust clustering algorithm clusters or discretizes the hidden unit activation values of the input patterns into a small number of



clusters. The rules are extracted in two steps. The network outputs are first described as classification rules in terms of the clustered hidden unit activation values. Each cluster of hidden unit activation values is then explained as rules that involve the input attributes. By merging the two sets of rules, the algorithm obtains a set of rules that explains the network outputs in terms of the input attributes of the data. A more concise set of rules can be thus expected from a network with fewer connections and fewer clusters of hidden unit activations [Setiono96], [Setiono00].

### **2.7.3.3 Taha's Hybrid Intelligent Architecture**

Taha's hybrid intelligent architecture (HIA) [Taha97] is a fully integrated hybrid architecture. It combines knowledge-based and artificial neural network systems. It has four phases involving domain knowledge representation, mapping of this knowledge into an initial connectionist architecture, network training, and rule extraction, respectively. The final phase is important because it can provide a trained connectionist architecture with explanation power and validate its output decisions. It can be used to refine and maintain the initial knowledge acquired from domain experts. Taha's HIA is show in Figure 2.17.

In Taha's HIA, there are seven main components: 1) a knowledge-based module, 2) a statistical module, 3) a mapping algorithm (node links algorithm), 4) a discretization module, 5) a connectionist module, 6) a rule extraction module, and 7) an integration module. Complete details of Taha's HIA can be found from [Taha97].

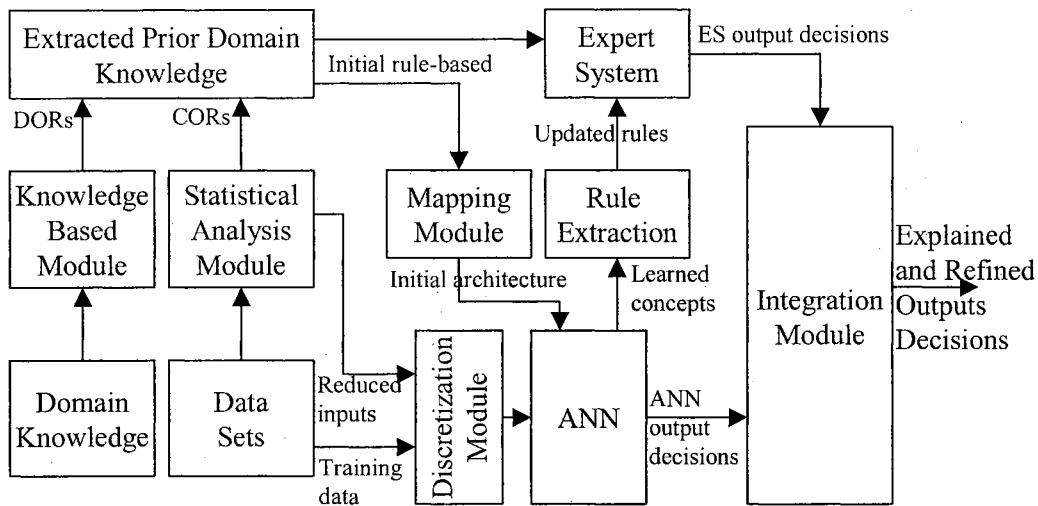


Figure 2.17: Taha's Hybrid Intelligent System Architecture

#### 2.7.4 Combined Numerical and Linguistic Paradigms

Combined numerical and linguistic paradigm is a type of multi-module hybrid system. Data in real applications can be in a numerical form or a linguistic form. Numerical data comes from sensor measurements while linguistic information comes from human experts. Numerical quantity of a decision-making system is more robust in noisy data; however, its drawback is the lack of representation power. Numerical data alone may be not sufficiently available for training. As a result, the resulted numerical learning system may be incomplete. Linguistic knowledge is more preferable when an explanation to the decision is called for, but linguistic rules alone may be not enough to build a rule base to cover the domain problem. In addition, some knowledge may be lost when the expert transfers his knowledge in constructing linguistic rules. Integration of both numerical data and linguistic data can improve the system to achieve higher generalization as well as to provide knowledge representation in an interpretable form.

Both features from numerical computation and linguistic computation should be preserved.

There are a number of researchers attempting to hybrid low-level numerical neural computations and higher-level linguistic rules to perform complicated tasks. Yang and Asada [Yang92] used hybrid numerical control of deburring robot. In their work, human skills and knowledge of performing a given task are transferred to robots through the acquisition and interpretation of human linguistic information. Based on the linguistic information, a nonlinear control structure is obtained in which the input space is partitioned in accordance with the linguistic labels that a human expert uses for describing his expertise. Zhou *et al.* [Zhou98] developed a hybrid intelligent controller that is a combination of linguistic and numerical information resulting in a neurofuzzy based integration and fuzzy rules extraction based integration. The neurofuzzy network structure uses linguistic information to set the initial parameters while input-output numerical data is used for updating the parameters.

A large number of researchers studied and developed networks that can deal with both numerical data and linguistic information. Ishibuchi *et al.* [Ishibuchi93] studied multilayer perceptron neural networks to handle linguistic data where an interval numerical generated from an  $\alpha$ -level method. Lin and Lu [Lin95], [Lin96] studied neural fuzzy learning system that is trained by fuzzy if-then rules. The system of Lin and Lu can process both numerical information and linguistic information. Based on an  $\alpha$ -level method, membership functions are transformed to fuzzy numbers that will be trained to the network. A crisp number is viewed as a fuzzy singleton thus can be treated the same way as the fuzzy numbers. Similarly, based on an  $\alpha$ -level method, Chen and Chang

[Chen2000] also developed a fuzzy perceptron neural network that learns from fuzzy if-then rules. In addition, Zhang *et al.* [Zhang2000] studied a granular neural network (GNN) to deal with numerical-linguistic data fusion and granular knowledge discovery in numerical-linguistic databases. The GNN is capable of compressing low-level granular data to high-level granular knowledge. Chakraborty *et al.* [Chakraborty99] designed a neurofuzzy feature selector using a modified multilayer perceptron neural network that generates fuzzy rules for input to a classifier. The neurofuzzy feature selector can receive inputs from several sources of knowledge including numerical data and human knowledge in linguistic form.

In the other extremes, several researchers have studied methodologies for linguistic rules extraction from numerical data. Linguistic rules can be directly generated from numerical data by using searching tools such as the genetic algorithm. In addition, numerical data can be trained to neural networks or neurofuzzy network before linguistic rules are extracted from the trained networks. The methods can be used to combine both linguistic data and numerical data in a fuzzy rule-based system. Fuzzy rules generated from numerical data consist of expert knowledge. For instance, Wang and Mendel [Wang92] developed a method for directly generating fuzzy rules from numerical data. Fuzzy rules can be generated from the input and output space of given data that is divided into fuzzy region. Ishibuchi *et al.* [Ishibuchi95], [Ishibuchi99] studied a method for generating linguistic rules from numerical data using a fuzzy partition method searching for a solution by using the genetic algorithm. Fahn *et al.* [Fahn99] studied a methodology for fuzzy rules extraction by using the combination of evolutionary algorithms and multilayer perceptron networks. Based on heuristic constraint on membership functions,

Chow *et al.* [Chow99] developed a method for fuzzy rules extraction from fuzzy/neural architectures. They stated that the method ensures that the final membership functions conform to prior heuristic knowledge.

In summary, from the literatures there are different ways to combine numerical data and linguistic information processing. The first groups of hybrid systems are the systems that use numerical data to derive linguistic if-then rules. Usually the multilayer perceptron networks, radial basis function networks, or fuzzy neural/neuro-fuzzy networks are used for learning from data. After the networks have been trained, rule extraction algorithms are then used to extract knowledge embedded (i.e., weight connections and biases) in the trained networks. The linguistic rules are then used in the decision-making mechanism in a fuzzy rule-based system. In these hybrid systems, the neural networks are used for rule construction purposes. The fuzzy rule-based systems are used as the main decision making process. The second groups of hybrid systems are the systems that learn both linguistic rules and numerical data. After being trained, these systems can deal with both fuzzy data and numerical data. These approaches have a black box structure because they often use multilayer perceptron neural networks. The explanation for the system's decision may not be understandable. The third groups of hybrid systems are hierarchically composed of two systems: a numerical model and a linguistic model. The two models receive input from numerical data and linguistic knowledge. The output of the hybrid system for these groups is the decision fusion from both the numerical model and the linguistic model.

Based on the literature in the area of hybrid intelligent systems, there remains a lot of work yet to be done. Especially, there is an interesting study on the investigation of

computation of low-level and high-level modules that are simultaneously operated, since human brains are not only the synaptic connections but also cognitive level for reasoning in linguistic sense. There is no unique method to bridge between the synaptic computation level and the cognitive linguistic level. This leads to the focus of the study in this dissertation.

**CHAPTER III**  
**THE ARCHITECTURE OF THE PROPOSED**  
**HYBRID INTELLIGENT SYSTEM**

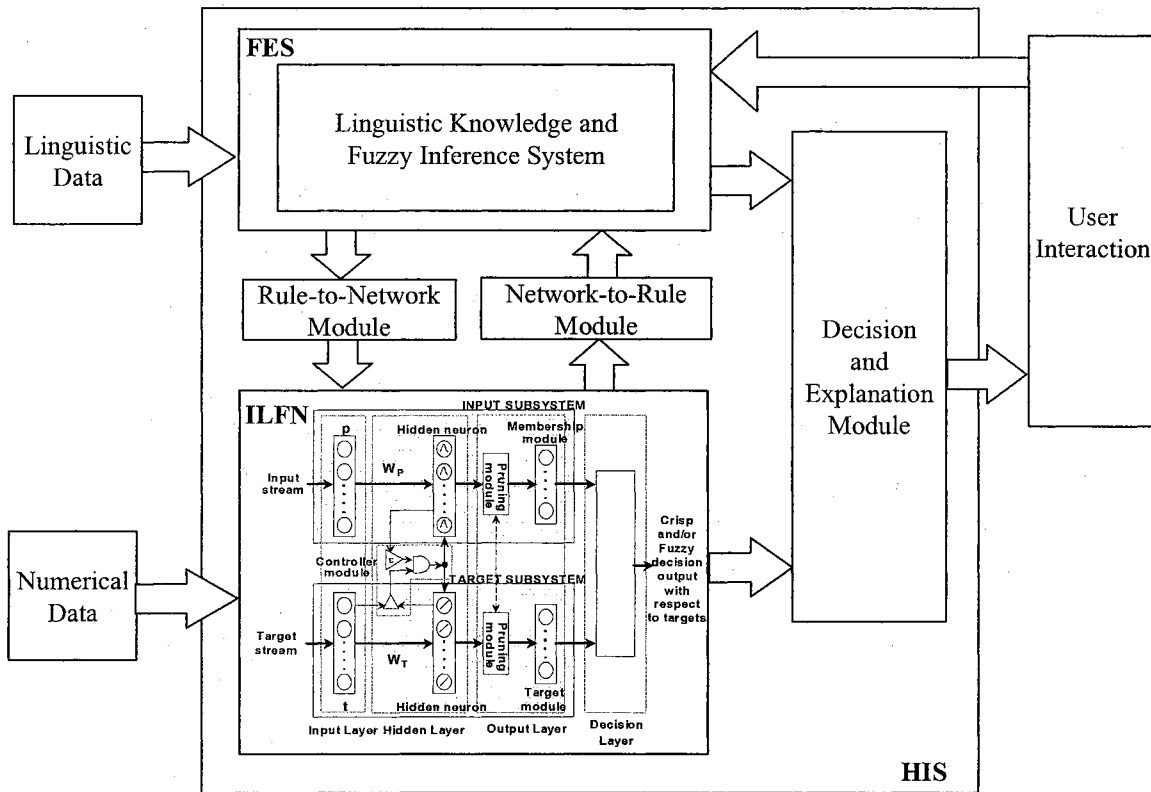


Figure 3.1: The Architecture of the Proposed Hybrid Intelligent System

The proposed HIS, shown in Figure 3.1, consists of five components: 1) an ILFN, 2) a FES, 3) a network-to-rule module, 4) a rule-to-network module, and 5) a decision-explanation module. Input data are brought into the system through both the ILFN and the FES. The ILFN and the FES are linked together by a network-to-rule module that is a rule extraction algorithm for mapping the ILFN to the FES, and a rule-to-network module that is a process for mapping the FES to the ILFN. The outputs of the ILFN and the FES connect to the decision-explanation module which makes decisions and provides

explanations based on the information received from both the ILFN and the FES. The human operators can interact with the system through an additional user interaction module. The details of each module are discussed in the following sections.

### 3.1 Incremental Learning Fuzzy Neural Network (ILFN)

The ILFN network is advanced from the fuzzy ARTMAP basic idea of on-line and incremental learning behavior. The architecture of the ILFN network is similar to the fuzzy ARTMAP; however, in details, the ILFN network operations are completely different from the fuzzy ARTMAP. While the fuzzy ARTMAP uses hyperbox membership functions, the ILFN network employs Gaussian membership functions that can prevent full membership of overlapping classes. The ILFN network architecture is detailed as follows.

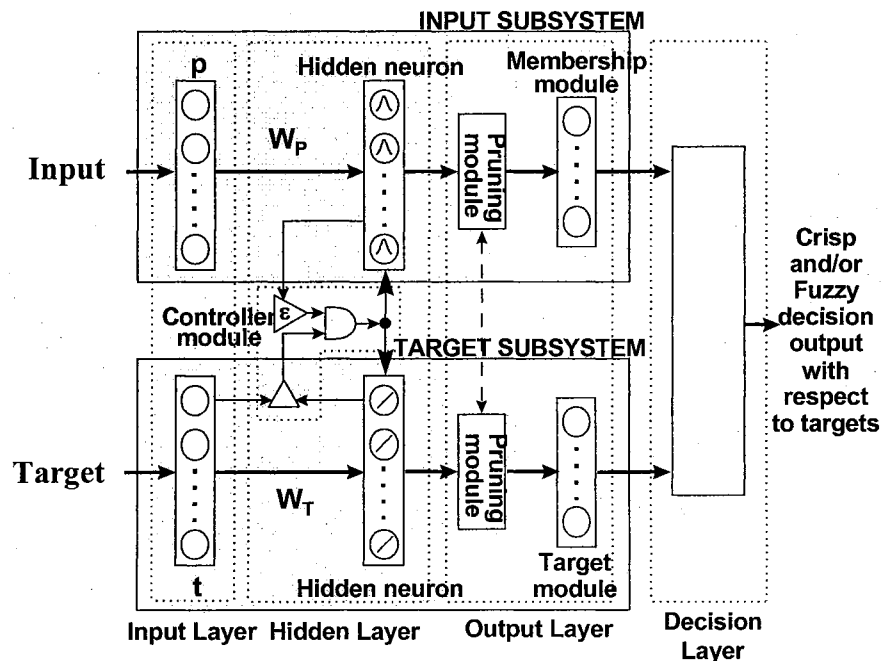


Figure 3.2: Network Architecture of the ILFN Classifier in the Supervised Learning Mode



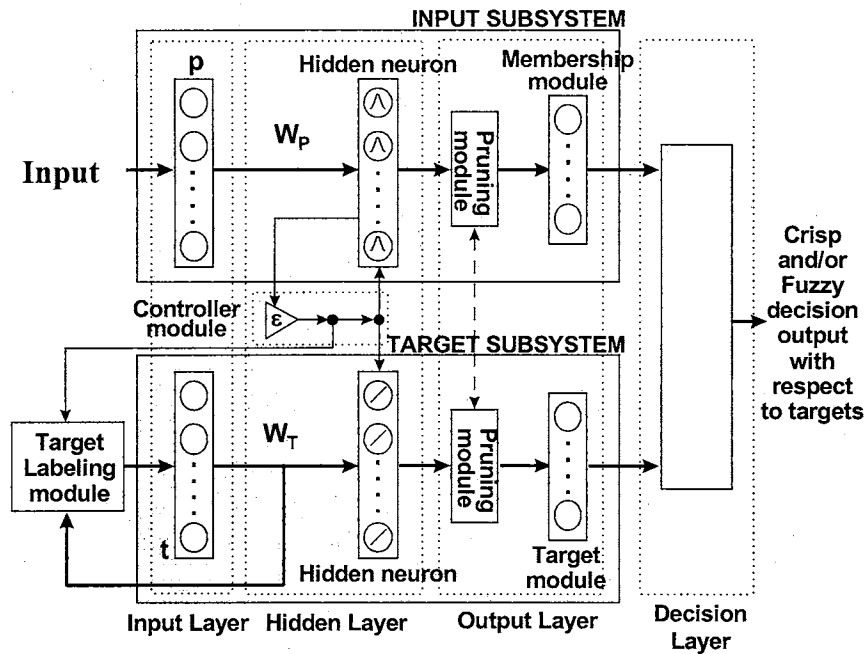


Figure 3.3: Network architecture of the ILFN Classifier in the Unsupervised Learning Mode

The architecture of the ILFN network is distinguished by two different modes: a supervised learning mode (as shown in Figure 3.2) and an unsupervised learning mode (as shown in Figure 3.3). The two learning modes differ only in the controller module and the target labeling module. Whereas the supervised learning mode requires pairs of input and target of patterns to construct “prototype” vectors that are centroid locations in hyper space of clusters, the unsupervised learning mode uses the target labeling module to assign the target class for a given input pattern.

The ILFN network has four layers: one input layer, one hidden layer, one output layer, and one decision layer, as shown in Figures 3.2 and 3.3. Generally, the first three layers of the system are composed of two subsystems: an input subsystem and a target subsystem. These subsystems are linked together via three connections: 1) the controller module in the hidden layer; 2) the pruning modules in both the input subsystem and the

target subsystem of the output layer; and 3) decision layer which is the link between the membership module in the input subsystem and the target module in the target subsystem. The following subsections present the details of the input subsystem, the target subsystem, the controller module as well as the fourth layer, the decision layer.

### 3.1.1 Input Subsystem

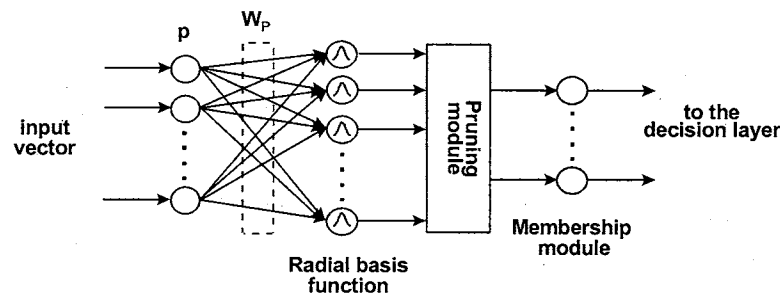


Figure 3.4: The Input Subsystem of the ILFN Classifier

Figure 3.4 illustrates the input subsystem of the ILFN classifier. The input subsystem is composed of three parts: 1) input neurons in the input layer, 2) a synaptic weight  $\mathbf{W}_P$  and Gaussian neurons in the hidden layer, and 3) a pruning module and a membership module in the output layer.

In the input layer, an element of an input vector  $\mathbf{p}$  connects to each neuron. The neurons in the input layer have no activation functions. The neurons in the input layer are fully connected to the neurons of the hidden layer via a synaptic weight matrix,  $\mathbf{W}_P$ , whose rows represent prototype vectors which are the centroids of radial basis functions in the hidden layer.  $\mathbf{W}_P$  is a trainable weight matrix using learning rules that will be discussed later.  $\mathbf{W}_P$  grows when a new prototype is detected. Growing of  $\mathbf{W}_P$  means that

additional rows are added to  $\mathbf{W}_p$  each time a neuron is added to the hidden layer. (How a new prototype is detected will be discussed in Subsection 3.1.8.)

In the hidden layer of the ILFN network, Gaussian membership functions are used. The Gaussian functions are centered on the mean vectors of clusters which are called prototypes of the input pattern space. The Gaussian membership functions are employed to fuzzify the input vectors,  $\mathbf{p}$ , into membership values,  $m_i$ , with respect to the distance measure between the input vectors,  $\mathbf{p}$ , and the  $i$ th prototypes. The membership function at the  $i$ th neuron,  $m_i(\mathbf{p}, \mathbf{w}_{p_i})$ , is defined by the following equation:

$$m_i(\mathbf{p}, \mathbf{w}_{p_i}) = \exp\left(-\frac{\|\mathbf{p} - \mathbf{w}_{p_i}\|^2}{2\sigma_i^2}\right), \quad i = 1, 2, \dots, L \quad (3.1)$$

where  $\|\cdot\|$  denotes the Euclidean distance which is used as a similarity measure between two vectors. The weight vector between the input layer and the  $i$ th hidden neuron,  $\mathbf{w}_{p_i}$ , is the center or mean vector at the  $i$ th neuron in the hidden layer.  $\sigma_i$  represents the standard deviation of the  $i$ th neuron in the hidden layer. For simplicity,  $\sigma_i$  used in (3.1) is the same in all directions.  $\sigma_i$  is determined by the average of the standard deviations in all directions. The membership function,  $m_i(\mathbf{p}, \mathbf{w}_{p_i})$ , of the hidden layer is used to fuzzify the distance between a given input vector  $\mathbf{p}$  and the  $i$ th centers  $\mathbf{w}_{p_i}$  into a real value  $m_i$  which represents the degree of similarity between  $\mathbf{p}$  and  $\mathbf{w}_{p_i}$ . The membership functions produce localized, bounded, and radially symmetric kernels. The value of these membership functions monotonically decreases as the distance from the function's center increases.

It is worth noting that the Euclidean distance can be replaced by other complicated distance metrics to obtain more complicated forms of clusters. Using the Euclidean distance, the clusters are formed hyper circles in hyperspace. Other metric

distance such as *Mahalanobis distance* [Duda73] which forms hyper ellipsoidal may be used; however, it is difficult, if not impossible, to learn online in real-time.

In the ILFN network, a class may have several prototypes. These prototypes have different degrees of belonging assigned to a pattern. Only one prototype with the highest degree of belonging is needed to represent a pattern. The prototypes with lower degrees of belonging generate redundant classes. To eliminate redundant classes, the pruning module is used in the output layer of the ILFN network. Instead of passing many duplicated classes, only distinguished classes are passed to the membership module. This makes the system easier for human users to interpret the output.

The pruning procedure of the ILFN network is different from the usual pruning procedures that eliminate insignificant neurons or weights [Mitra97]. The pruning module used in the ILFN network is a *short-term memory* which refers to the information that is stored for a short period of time. The information does not stay the same for each input pattern that is processing. In this case, for each input pattern presented, the information in the pruning module is processed and stored in the memory until there is another input pattern coming in. (The pruning algorithm is described in Step 6 of the “ILFN classification algorithm” in Subsection 3.1.8.)

In addition, the pruning module in the input subsystem and the pruning module in the target subsystem work in the same way. From each prototype, the highest membership value in the input subsystem is selected (by the algorithm) to represent the degree of similarity with respect to a class in the target subsystem. The output of the pruning module is presented to the membership module.

The membership module in the output layer of the input subsystem receives information, which is the pruned membership values, transmitted from the pruning module and passes it (without any processing) to the decision layer. The information stored in the membership module is a short-term memory, which means that the information in the membership module differs for different input vectors. Each membership value in the membership module indicates the degree of similarity of an input vector with respect to the target classes of the classifier. The membership values are then mapped in the same order of indices to classes in the target module in the target subsystem via the decision layer.

### 3.1.2 Target Subsystem

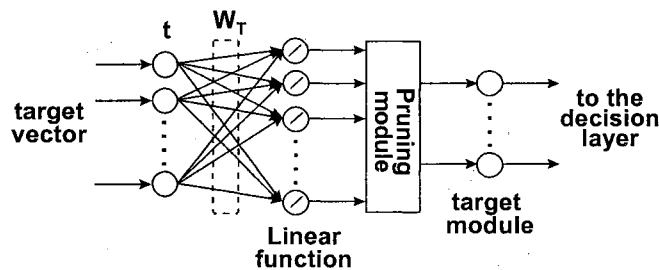


Figure 3.5: The Target Subsystem of the ILFN Classifier

The target subsystem of the ILFN classifier is depicted in Figure 3.5. Each neuron of the input layer in the target subsystem is fully connected to each element of a target vector. A synaptic weight matrix  $W_T$ , which needs no training, connects the neurons of the input layer to the neurons of the hidden layer. At the same time that  $W_P$  is growing,  $W_T$  is automatically constructed by using the corresponding target of a current input pattern that inputs to the system. When a neuron is added to the hidden layer, an

additional row is added to  $\mathbf{W}_T$ . These hidden neurons of the target subsystem are activated by linear functions.

As in the input subsystem, the pruning module of the output layer in the target subsystem is used to eliminate redundant classes in the hidden layer. Instead of passing many duplicate classes, only classes with prototypes that have the highest degree of membership for a given input are passed to the membership module. As mentioned before, the pruning module in the target subsystem works the same way as the pruning module in the input subsystem does.

The target module, which is in the output layer of the target subsystem, receives information passed from the pruning module and submits it to the decision layer. Each neuron of the target module is a class or a target of an input vector. The target module is a short-term memory as is the membership module of the input subsystem. In the same order of indices, the target module is then mapped to the membership module of the input subsystem via the decision layer.

### **3.1.3 Controller Module**

The controller module is used to control the growing number of neurons in the hidden layers of both the input subsystem and the target subsystem. It operates differently in the controller module in supervised learning mode and unsupervised learning mode.

In the supervised learning mode, there are three components in the controller module: two comparators and one AND gate. One comparator is used to compare the winning membership value from the output of the hidden layer of the input subsystem to the threshold,  $\varepsilon$ . ( $\varepsilon$  is heuristically chosen by the user.) The output of this comparator

becomes “true” if the winning membership value is smaller than the  $\epsilon$ . This implies that the input vector is significantly different from all existing prototype vectors. The output is sent to one input of the AND gate. Another comparator, which has two inputs, is used to compare the desired target with the predicted output which is stored in the hidden layer of the target subsystem. The output of the comparator becomes “true” if both the desired target and the predicted output are the same. It is sent to another input of the AND gate. If both inputs of the AND gate are “true,” its output becomes “true.” This allows the system to add one more neuron to the hidden units. In other words, the system generates more neurons whenever the membership value of the winning neuron is smaller than the threshold,  $\epsilon$ , and the desired target and the decision output are the same.

In the unsupervised learning mode, the controller module of the ILFN classifier has only one component which is a comparator. The comparator is used to compare the winning membership value in the hidden layer to the threshold,  $\epsilon$ . The output of this comparator becomes “true” if the winning membership value is smaller than  $\epsilon$ . If the output of the comparator is “true,” meaning that a new category is detected, the system adds a new neuron to the hidden layer using the input pattern as the new prototype, then the target labeling module distinguishably assigns a corresponding target to the new prototype.

In addition to a comparator, the controller module in the unsupervised learning mode has a target labeling module used to assign a target for a new prototype. The target labeling module receives one input from the output of the controller module in the hidden layer of the target subsystem. This input from the controller module tells the target labeling module to assign a target when a new neuron is added to the system. Another

input of the target labeling module, representing targets of prototypes, is used to check the existing targets in order to assign a new target that differs from the existing targets.

#### **3.1.4 Decision Layer**

The decision layer is used to map the membership values in the membership module of the input subsystem to the target classes in the target module of the target subsystem. The output from the decision layer is the output of the system. The decision output can be interpreted as a soft decision or a hard decision. For the soft decision, the decision output assigns different membership values to the pattern classes or prototypes. This allows a given pattern to belong to more than one class with different degrees of similarity measures. For the hard decision, the decision output selects only one class with the highest membership value.

#### **3.1.5 ILFN System Dynamics**

Both  $\mathbf{W}_P$  and  $\mathbf{W}_T$  are allowed to grow when the system detects new classes. However, only  $\mathbf{W}_P$  can adaptively change its information or learn new prototypes. At the initialized state, there are no neurons in the hidden layer. The first neuron in the hidden layer is setup after the first input vector  $\mathbf{p}$  is presented to the input subsystem of the network while the first target vector  $\mathbf{t}$  is presented to the input layer in the target subsystem. Then both  $\mathbf{W}_P$  and  $\mathbf{W}_T$  setup the first neuron using  $\mathbf{p}$  and  $\mathbf{t}$  respectively. The next input vector is compared to the existing prototype. If there is a significant difference (depending on the threshold,  $\epsilon$ ), then a new neuron is added to the hidden layer;  $\mathbf{p}$  is added to  $\mathbf{W}_P$  and  $\mathbf{t}$  is added to  $\mathbf{W}_T$ . On the other hand, if the input vector meets the



similarity criterion then, instead of adding a new neuron, the learning process is performed. The  $\mathbf{W}_p$  and other parameters are updated to include the new data in the existing prototypes.

### 3.1.6 Learning Process

The learning process takes place only in the hidden layer. It adapts the synaptic weight,  $\mathbf{W}_p$ , and updates the parameters regarding the pattern clusters of the input space. In the learning process, each input vector  $\mathbf{p}$  from the input space is fuzzified to a membership value at each node of the hidden layer with respect to the distance measure between input vector  $\mathbf{p}$  and the synaptic weight matrix  $\mathbf{W}_p$ . The winning node of the hidden layer is determined by the defuzzification process using the fuzzy OR operation ( $\vee$ ) defined as:

$$winner \equiv m_1 \vee m_2 \vee \dots \vee m_L, \quad (3.2)$$

$$J \equiv winner\ index = \arg \max_i(m_i), \quad (3.3)$$

where  $m_1 \vee m_2 = m_1$  if  $m_1 \geq m_2$ ;  $m_1 \vee m_2 = m_2$  if  $m_1 < m_2$ . The membership value  $m_i$ ,  $i = 1, \dots, L$ , is calculated by (3.1). Only the parameters of the winner node (i.e.,  $J$ th neuron) including the number of patterns, the mean, and the standard deviation are updated, while other losing nodes remain the same, as follows:

$$cnt_{J,new} = cnt_{J,old} + 1, \quad (3.4)$$

$$\mathbf{w}_{PJ,new} = \frac{\mathbf{w}_{PJ,old}(cnt_{J,new} - 1) + \mathbf{p}}{cnt_{J,new}}, \quad (3.5)$$

$$\mathbf{s}_{J,\text{new}} = \begin{cases} \sqrt{\left(1 - \frac{1}{\text{cnt}_{J,\text{new}}}\right) \mathbf{s}_{J,\text{old}}^2 + \frac{(\mathbf{s}_{J,\text{old}} - \mathbf{p})^2}{\text{cnt}_{J,\text{new}}}} & \text{if } \text{cnt}_{J,\text{new}} > 1, \\ \mathbf{s}_0 & \text{otherwise;} \end{cases} \quad (3.6)$$

where a parameter with the subscript “old” represents that parameter before updating and a parameter with the subscript “new” represents that parameter after updating.  $\text{cnt}_J$  (abbreviation from “count”) represents the number of patterns that have been counted into the  $J$ th prototype. The mean  $\mathbf{w}_{pJ}$ , the center of the  $J$ th prototype, is a row in the synaptic weight  $\mathbf{W}_p$ . The standard deviation,  $\mathbf{s}_J = [\sigma_{J1}, \sigma_{J2}, \dots, \sigma_{JM}]$ , will be used to indicate the spread of the data in the  $J$ th prototype.  $\sigma_{Ji}$ ,  $i = 1, \dots, M$ , represents the standard deviation of the  $J$ th prototype in the  $i$ th dimension and  $M$  is the dimension of the pattern space.  $\mathbf{s}_0 = [\sigma_{01}, \sigma_{02}, \dots, \sigma_{0M}]$  is the initial standard deviation representing the isotropic spread in pattern space of a new category for the first sample. Initial standard deviation,  $\sigma_{0i}$ ,  $i = 1, \dots, M$ , is usually chosen small enough (e.g., a value between 0.001 and 0.05) to include only the pattern that is setup for the new prototype. After the patterns near the prototype are included in the same prototype, the standard deviation  $\mathbf{s}_J$  is updated accordingly.

Equations (3.4), (3.5), and (3.6) are learning rules used to update the prototype variables in the input subsystem. The number of patterns belonging to each cluster is updated by Equation (3.4). By knowing the previous centers and the number of patterns that belong to a cluster, new centers can be calculated by Equation (3.5). The estimated standard deviations can be calculated if the previous standard deviation and the number of the patterns belonging to a cluster are known. Estimated standard deviations, which are the spread of the Gaussian membership functions, are determined by Equation (3.6).

### 3.1.7 Decision Boundaries

The purpose of pattern classification is to determine to what class a given sample belongs. Through an observation or measurement process, a set of numbers which make up the observation vector is obtained. The observation vector serves as the input to a decision rule by which the sample to one of the given classes is assigned.

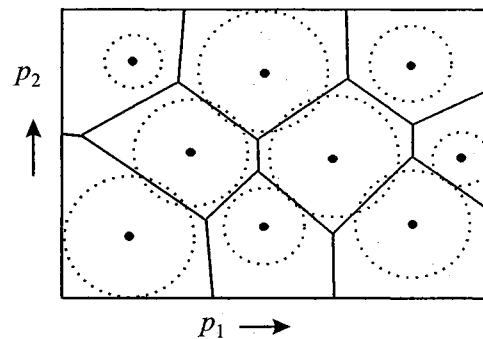


Figure 3.6: The Decision Boundaries Among Prototypes of the ILFN Classifier

The decision boundaries of the ILFN network distinguish among prototypes in the Voronoi tessellation [Kohonen97]. Each prototype has its own region separated by the decision boundaries. Since the ILFN classifier uses Gaussian type membership functions with different standard deviations, the soft decision boundaries of the ILFN classifier are quadratic. However, the hard decision boundary between the neighboring prototype vectors is a hyperplane containing the points that have the same degree of the membership value, as shown in Figure 3.6. Figure 3.6 shows the decision boundaries among prototypes of the ILFN network in which dotted circles indicate the spread of statistical data for each prototype.

### 3.1.8 ILFN Classification Algorithm

The ILFN network can learn in two different ways: 1) supervised learning which requires both input patterns and the corresponding targets; and 2) unsupervised learning which requires only input patterns without the corresponding targets and in which the target labeling module will assign appropriate class labels. The classification algorithm of the ILFN classifier is outlined as follows.

**Step 1:** Set the user-defined threshold parameter ( $\epsilon$ ), the initial standard deviation  $\sigma_0$ , and the maximum number of patterns allowed in each cluster. (The latest parameter is used in the unsupervised mode to force the system to stop updating weights.)

**Step 2:** Retrieve the first input pattern

- Use the first input pattern  $\mathbf{p}$  to set up the first prototype (or mean) to  $\mathbf{W}_P$ .
- Set the number of patterns for the first node to be 1.
- Set the standard deviation equal to the initial standard deviation,  $\sigma_0$ .
- Set a new neuron to  $\mathbf{W}_T$  using the first target  $\mathbf{t}$  to be the corresponding target of the prototype in  $\mathbf{W}_P$ .

**Step 3:** Retrieve the next training sample with an input and target.

**Step 4:** Measure the Euclidean distance between the input  $\mathbf{p}$  and the prototype  $\mathbf{W}_P$ .

**Step 5:** Calculate membership values for each node using the Gaussian type radial basis function.

**Step 6:** Assign membership values to each node. The current input pattern has different degrees of belongings for each node or prototype. For each class, select the maximum membership value from each prototype to represent the degree of similarity with respect to that class.

**Step 7:** Identify the largest membership (called *winner*) using the fuzzy OR operator.

**Step 8:** *If there is the corresponding target (i.e., supervised learning mode),*

- 1) If the value *winner* is larger than  $\epsilon$  and the target  $\mathbf{t}$  is the same value as  $\mathbf{W}_T$  at the winning node then update weight  $\mathbf{W}_P$ , the standard deviation, and the number of patterns belonging to this node.
- 2) If 1) is not satisfied, then:
  - Set a new node center for  $\mathbf{W}_P$  using the input pattern  $\mathbf{p}$ .
  - Set the number of patterns for the new node to be 1.
  - Set the initial standard deviation to the new node.
  - Add a new neuron to  $\mathbf{W}_T$  using the new target  $\mathbf{t}$  as the corresponding target of a new prototype in  $\mathbf{W}_P$ .

*If there is no corresponding target (i.e., unsupervised learning mode),*

- 1) If the value *winner* is larger than  $\epsilon$  and the number of patterns is less than the maximum number of allowed patterns, then update the weight  $\mathbf{W}_P$ , the standard deviation, and the number of patterns belonging to this node. Identify the class output which is stored in  $\mathbf{W}_T$  at the same index of the winning node of  $\mathbf{W}_P$ .
- 2) If the value *winner* is smaller than  $\epsilon$  then

- Set a new node center for  $\mathbf{W}_P$  using the input pattern  $\mathbf{p}$ .
- Set the number of patterns for the new node to be 1.
- Set the initial standard deviation to the new node.
- Add a new neuron to  $\mathbf{W}_T$  and assign a new target as the corresponding target of a new prototype in  $\mathbf{W}_P$ . (The assigned new target must be significantly different from the existing targets already stored in  $\mathbf{W}_T$ . For example, if there exist targets in  $\mathbf{W}_T = [1 \ 2 \ 3]^T$ , the new target should be “4,” that is  $\mathbf{W}_T$  becomes  $[1 \ 2 \ 3 \ 4]^T$ .)

**Step 9:** If there are no more input patterns, then stop. Otherwise, go to step 3.

Usually, if the user knows both input patterns and their targets, the network is trained in the supervised learning mode. After supervised training, the network is used in a pattern classification system. The ILFN network can detect new categories that have not been presented as training data. When the system detects new categories, it employs the unsupervised learning mode by using the target labeling module to assign the corresponding targets to the input patterns. The targets that are assigned to the novel prototypes are chosen significantly different from the existing targets in the target module.

ILFN network uses four weighting parameters:  $\mathbf{W}_P$ ,  $\mathbf{W}_T$ ,  $\mathbf{S}$ , **count**, as well as one threshold parameter,  $\epsilon$ .  $\mathbf{W}_P$  is the hidden weight of the input subsystem. Each row (i.e., a node in the hidden layer) of  $\mathbf{W}_P$  represents a mean or centroid of a cluster. Each node of  $\mathbf{W}_T$  stores the corresponding target of the input prototype patterns.  $\mathbf{S}$  is the standard deviation matrix and the **count** vector is the number of patterns that belong to each node.

$\epsilon$ , selected within the range  $[0, 1]$ , is the threshold parameter that controls the number of clusters. The system generates more clusters if  $\epsilon$  is large and fewer clusters if it is small. However, clusters that belong to the same class are grouped together via the pruning module. The details of learning algorithm can be found in [Meesad98], [Yen99], [Yen01].

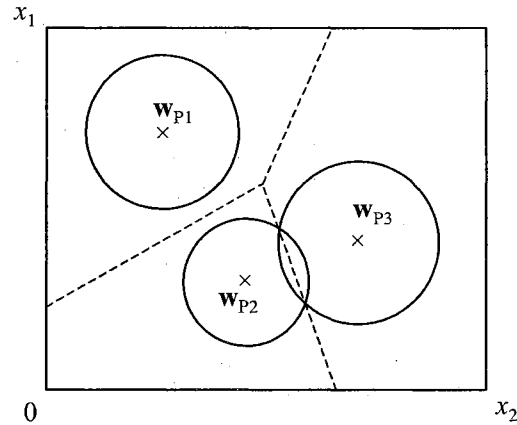


Figure 3.7: ILFN Decision Boundaries of a Three-Class, Two-Dimensional Pattern Space

Figure 3.7 shows the decision boundaries when the ILFN is used to classify a three-class, two-dimensional pattern space. Three circles indicate the locations of the three clusters of the three classes centered at  $w_{P1}$ ,  $w_{P2}$ , and  $w_{P3}$ . The membership values are highest when the patterns are located at the centers of the clusters. The membership values monotonically decrease when the distances between the patterns and the centers of the clusters increase. The size of the circle depends on the variances of the patterns that belong to the clusters. A pattern outside a circle indicates a near-zero membership degree of belonging to the cluster. The dashed line indicates the boundaries of each cluster.

A trained ILFN does not exhibit a clear meaning of knowledge embedded inside its structure. Linguistic knowledge is more preferable if an explanation about the decision

is needed. Thus it is desirable to transform the knowledge of the trained ILFN into a representation that is easier to comprehend in linguistic form used in a FES. A FES has a close relationship with an ILFN network in that it can be mapped from one to another. In order to employ both numerical calculation from an ILFN and linguistic processing from a FES, we will combine both a trained ILFN and the mapped FES into the same hybrid system. The output decision of the hybrid system is based on both the ILFN and the FES. The resulting hybrid system would provide complementary features from both the ILFN and the FES. The hybrid system seems to show the ability to deal with more complex problems that need an explanation capability.

The following sections describes the details of the FES used in this study, as well as how to map knowledge from a trained ILFN to a FES and vice versa.

### 3.2 Fuzzy Expert System (FES)

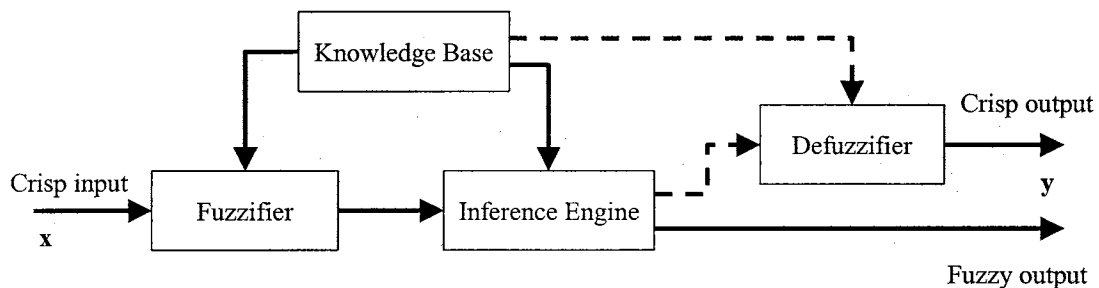


Figure 3.8: A Fuzzy Expert System (FES)

A FES can be thought of as a special kind of expert systems (ESs). In fact, a FES is an ES that is incorporated with fuzzy sets [Zadeh65]. Thus, a FES exhibits transparency to users. Users can easily understand the decision made by a FES due to the fact that the rule base is in “if-then” form used in natural languages. From a knowledge



representation viewpoint, a fuzzy if-then rule is a scheme for capturing knowledge that is imprecise by nature.

Figure 3.8 illustrates a schematic diagram of a FES. A FES is composed of four main modules: a fuzzifier, an inference engine, a defuzzifier, and a knowledge base. The function of the fuzzifier is to determine the degree of membership of a crisp input in a fuzzy set. The fuzzy knowledge base is used to represent the fuzzy relationships between input and output fuzzy variables. The output of the fuzzy knowledge base is determined by the degree of membership specified by the fuzzifier. The inference engine utilizes the information from the knowledge base as well as from the fuzzifier to infer additional information. The output of a FES can be fuzzy values from which the inference engine processes. The output in fuzzy value format is advantageous in pattern classification problems since the fuzzy values indicate the degree of belongings of a given pattern to class prototypes. Optionally, the defuzzifier is used to convert the fuzzy output of the system into crisp values.

### **3.2.1 Knowledge Base of the proposed FES**

In the proposed FES, a knowledge base is used for the system to generate an explanation as well as to make a decision. A knowledge structure used in the proposed FES comprises of 1) input features' names, 2) variables' ranges, 3) number of linguistic labels, 4) linguistic labels, 5) membership functions, 6) membership functions' parameters, and 7) fuzzy if-then rules. The information about the knowledge structure of the FES can be provided by experts or automatically generated from data. For an *M*-

dimensional pattern space, the components in the knowledge base used in the FES are detailed as follows.

*Knowledge Base (K)*

$$\mathbf{K} = \{ \mathbf{FN}, \mathbf{VR}, \mathbf{NL}, \mathbf{Ling}, \mathbf{MF}, \mathbf{MP}, \mathbf{R} \} \quad (3.7)$$

*Input Features' Names (FN)*

$$\mathbf{FN} = \{ fn_1, fn_2, \dots, fn_M \} \quad (3.8)$$

*Variables' Ranges (VR)*

$$\mathbf{VR} = \left\{ \begin{array}{c} Vmin_1, Vmax_1 \\ Vmin_2, Vmax_2 \\ \vdots \\ Vmin_M, Vmax_M \end{array} \right\} \quad (3.9)$$

*Number of Linguistic Labels (NL)*

$$\mathbf{NL} = \{ N_1, N_2, \dots, N_M \}; \quad N_j \in \{ 2, \dots, 9 \} \quad (3.10)$$

To maintain comprehensibility of the linguistic model, the number of the linguistic variables should be as small as possible. It is suggested that it should not be larger than nine [Jin00].

*Linguistic Labels (Ling)*

$$\mathbf{Ling} = \left\{ \begin{array}{c} \{ l_{11}, l_{12}, \dots, l_{1N_1} \} \\ \{ l_{21}, l_{22}, \dots, l_{2N_2} \} \\ \vdots \\ \{ l_{M1}, l_{M2}, \dots, l_{MN_M} \} \end{array} \right\} \quad (3.11)$$

where  $l_{jk}$  is a linguistic label in the  $j^{\text{th}}$  dimension and  $k$  is the index to it.

*Membership Functions (MF)*

$$\mathbf{MF} = \left\{ \begin{array}{c} \{mf_{11}, mf_{12}, \dots, mf_{1N_1}\} \\ \{mf_{21}, mf_{22}, \dots, mf_{2N_2}\} \\ \vdots \\ \{mf_{M1}, mf_{M2}, \dots, mf_{MN_M}\} \end{array} \right\} \quad (3.12)$$

*Membership Functions' Parameters (MP)*

$$\mathbf{MP} = \left\{ \begin{array}{c} \{\mathbf{mp}_{11}, \mathbf{mp}_{12}, \dots, \mathbf{mp}_{1N_1}\} \\ \{\mathbf{mp}_{21}, \mathbf{mp}_{22}, \dots, \mathbf{mp}_{2N_2}\} \\ \vdots \\ \{\mathbf{mp}_{M1}, \mathbf{mp}_{M2}, \dots, \mathbf{mp}_{MN_M}\} \end{array} \right\} \quad (3.13)$$

$$\mathbf{mp}_{jk} = \{mp_{jk1}, mp_{jk2}, \dots, mp_{jkn_{jk}}\}; j = 1, \dots, M; k = 1, \dots, N_j. \quad (3.14)$$

Assume that  $mf_{jk}$  is a Gaussian membership function,  $\mathbf{mp}_{jk}$  has two variables which are  $\sigma$  and  $\mu$ , a standard deviation and a mean of a Gaussian membership function, respectively. Thus, we have

$$\mathbf{mp}_{jk} = \{mp_{jk1}, mp_{jk2}\} = \{\sigma_{jk}, \mu_{jk}\}; j = 1, \dots, M; k = 1, \dots, N_j. \quad (3.15)$$

*Fuzzy If-Then Rules (R)*

$$\mathbf{R} = \{\mathbf{A}_{L \times M}, \mathbf{B}_{L \times 1}, \mathbf{CF}_{L \times 1}\} = \left\{ \begin{array}{cccccc} A_{11} & A_{12} & \dots & A_{1M} & B_1 & CF_1 \\ A_{21} & A_{22} & \dots & A_{2M} & B_2 & CF_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ A_{L1} & A_{L2} & \dots & A_{LM} & B_L & CF_L \end{array} \right\}. \quad (3.16)$$

$\mathbf{A}_{L \times M}$  represents the antecedent part of the if-then rules;  $\mathbf{B}_{L \times 1}$  and  $\mathbf{CF}_{L \times 1}$  constitute the consequent part of the if-then rules; where  $L$  and  $M$  is the number of fuzzy rules and the dimension of the pattern space, respectively.  $A_{ij}$ ,  $i = 1, \dots, L$ ,  $j = 1, \dots, M$ , is the

antecedent of the  $i$ th rule for the  $j$ th dimension.  $A_{ij} \in \{0, 1, \dots, N_j\}$  is the index of a linguistic label in the  $j$ th dimension of the linguistic labels (**Ling**) of the  $i$ th rule. If  $A_{ij}$  is “0” then the system uses a *don't care* label in which its activation function is always a unity membership grade.  $B_i$  is a constant value that is a class consequent part of the  $i$ th rule.  $CF_i$ , a value in  $[0, 1]$ , is a confident factor of the  $i$ th rule. In a FES, for a finite class pattern classification problem with an  $M$ -dimensional pattern space, linguistic knowledge can be written as a set of fuzzy if-then rule in a natural language as follow:

$$R_i: \quad \text{IF } x_1 \text{ is } A_{i1} \text{ AND } x_2 \text{ is } A_{i2} \text{ AND } \dots \text{ AND } x_M \text{ is } A_{iM},$$

$$\text{THEN } \mathbf{x} = \{x_1, x_2, \dots, x_M\} \text{ belongs to Class } B_i \text{ with confident factor } CF_i; \quad (3.17)$$

where  $R_i$ ,  $i = 1, \dots, L$ , is the label of the  $i$ th rule and  $A_{ij}$  indicates a linguistic label such as *small, medium, or large*.

Assume that Gaussian membership functions are employed in the FES. The rule firing strength or matching degree  $\phi_i$  can be computed by the following equation:

$$\phi_i = \min_j \left\{ \exp \left[ - \left( \frac{x_j - \mu_{ij}}{\sigma_{ij}} \right)^2 \right] \right\}, \text{ for } i = 1, \dots, L; j = 1, \dots, M; \quad (3.18)$$

where  $\mu_{ij}$  and  $\sigma_{ij}$  are the mean and the standard deviation, respectively, of the linguistic label indexed by  $A_{ij}$ ; and *min* is a  $T$ -norm operator which can be replaced by *product*.

After computing the firing strength from each rule, the class output,  $C_y$ , is calculated by using the inference mechanism as follow:

$$C_y = B_J; J = \arg \max_i (\phi_i \cdot CF_i). \quad (3.19)$$

### 3.2.2 Fuzzy If-Then Rule Generation

For the completeness of the rule structure in a FES, grid partition methods are widely used for partitioning input space into grid cells. Fuzzy if-then rules can be obtained by using fuzzy grid partitions [Wang92]. Despite its advantage of providing the completeness of rule structure, the grid partition method has a disadvantage in that the number of fuzzy rules increases exponentially as the dimension of the input space increases. Since each cell represents a fuzzy if-then rule, the number of fuzzy if-then rules is usually very large. The system becomes a black-box scheme that is not comprehensible to human users. Pattern classification problems in the real world often have large dimensions. It is undesirable to directly use the grid-type partitioning for constructing fuzzy if-then rules.

To obtain a smaller number of fuzzy rules, projection from clusters [Setnes98a], [Jin99] is called for. Clustering algorithms can be used for partitioning data points into a small number of clusters. Each cluster then represents a fuzzy relation and corresponds to a rule. The fuzzy sets in the antecedent parts of the rules are projected from the clusters onto the corresponding axis of the data space. The number of rules generated from projection method is smaller than the number of rules generated from grid-type partitioning. A more compact linguistic model is obtained. However, the fuzzy sets that are directly projected from clustering methods may not be transparent or crystal clear enough for human interpretation. The number of fuzzy sets from the projection may be very large and redundant since the projected fuzzy sets may be very similar resulting in a fuzzy system that is not optimal since some of the fuzzy sets can be discarded without losing the generalization. The problem mentioned above can be solved using rule

simplification methods [Jin99], [Setnes98b]. Alternatively, fuzzy rules can be generated by projection from trained ILFN parameters.

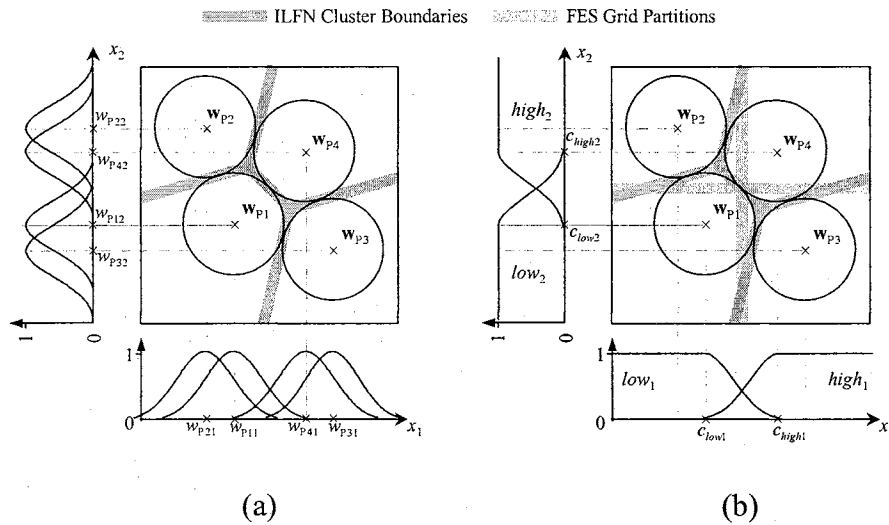


Figure 3.9: (a) Projection of ILFN to One-Dimensional Fuzzy Sets; (b) FES Grid

#### Partition with Its Parameter Projected from a Trained ILFN

The ILFN groups the patterns in the input space into a number of clusters. Based on grid partition methods, the clusters and its parameters of the trained ILFN can be mapped to fuzzy if-then rules. The number of fuzzy if-then rules is equal to the number of clusters in the trained ILFN. The number of fuzzy sets in each dimension depends on the number of grid partitions chosen. The parameters of fuzzy sets are projected from the cluster parameters of the trained ILFN. Figure 3.9 (a) shows the projection of ILFN to one-dimensional fuzzy sets. There are four fuzzy sets resulted from the projection from ILFN parameters to each axis. We can see that the fuzzy sets projected to  $x_1$  and  $x_2$  dimensions in Figure 3.9 (a) have some similarity. For example, in  $x_1$  axis, two Gaussian membership functions centered at  $w_{p21}$  and  $w_{p11}$  are highly overlapping. In the same fashion, two Gaussian membership functions centered at  $w_{p41}$  and  $w_{p31}$  are highly

overlapping. There are only two fuzzy sets in each axis when the fuzzy grid partitioning method is used. Combining grid partitioning method and projection method is shown in Figure 3.9 (b). The parameters of fuzzy sets in Figure 3.9 (b) are projected and adapted from the clusters of the trained ILFN.

Using a grid-based projection method, the fuzzy if-then rules of the FES are extracted from a trained ILFN. The hidden numerical weights of the ILFN are mapped into initial fuzzy if-then rules. A genetic algorithm is then used to select only discriminatory features resulting in a more compact rule set with highly transparent fuzzy sets or in other words easily understandable linguistic labels. (*Transparency* of the fuzzy system means that the rule structures and fuzzy sets of a fuzzy expert system can be easily understood by experts or experienced users in the problem domain.) Next section describes the method used to map the ILFN to the FES.

### 3.3 Network-To-Rule Module

Since the knowledge embedded in the ILFN is not in a linguistic form, the ILFN lacks of an explanation capability. ILFN weights can be extracted by using a rule extraction algorithm to obtain linguistic rules. A meaningful explanation in reasoning process can then be generated from the linguistic model i.e., the FES. The mechanism used for mapping a trained ILFN to a linguistic knowledge base operates inside the network-to-rule module. The mechanism is called *ilfn2rule* algorithm. (The name “ilfn2rule” comes from that the algorithm is used to transform “ilfn” parameters “to” fuzzy “rules”.)

### 3.3.1 ILFN2RULE Algorithm

Using a grid-based projection method, the `ilfn2rule` algorithm is used to map a trained ILFN to fuzzy if-then linguistic rules. The user specifies membership functions' types (**MF**). Any type of fuzzy membership functions can be used. In this study, Gaussian membership functions are used. The rule extraction algorithm is given in four steps described below.

**Step 1:** Retrieve trained ILFN parameters ( $\mathbf{W}_P$ ,  $\mathbf{W}_T$ , **count**) as well as the numbers of linguistic labels (**NL**). (The numbers of linguistic labels are determined during the genetic optimization process that will be discussed later.)

**Step 2:** Calculate membership functions' parameters (**MP**) that are a center and a standard deviation for each linguistic label in the case that Gaussian membership function is used. Centers of Gaussian functions can be determined from the variables' ranges (**VR**) that are minimum and maximum values of the numerical weight  $\mathbf{W}_P$  for the ILFN network.

$$Vmin_j = \min (w_{P1j}, w_{P2j}, \dots, w_{PLj}) = \min_i (w_{Pij}), \quad (3.20)$$

$$Vmax_j = \max (w_{P1j}, w_{P2j}, \dots, w_{PLj}) = \max_i (w_{Pij}), \quad (3.21)$$

$$res_j = \frac{Vmax_j - Vmin_j}{N_j - 1}, \quad (3.22)$$

where  $i = 1, \dots, L; j = 1, \dots, M$ ;  $L$  is the number of hidden nodes, i.e., prototypes created by the ILFN network;  $M$  is the dimension of the pattern space;  $res_j$  represents the numerical resolution between linguistic variables in the  $j$ th dimension;  $Vmax_j$  and  $Vmin_j$  are the maximum and the



minimum values of the weight  $\mathbf{W}_P$  in the  $j$ th dimension; and  $N_j$  is the number of linguistic variables in the  $j$ th dimension.

$$\mu_{jk} = \begin{cases} Vmin_j & \text{for } k = 1 \\ \mu_{j,k-1} + res_j & \text{for } k = 2, \dots, N_j \end{cases}, \quad (3.23)$$

$$\mu_j = [\mu_{j1}, \mu_{j2}, \dots, \mu_{jN_j}], \quad (3.24)$$

$$\sigma_j = \sqrt{-\left(\frac{res_j^2}{\sqrt{2} \times \ln \lambda}\right)}, \quad (3.25)$$

where  $\mu_{jk}$ ,  $k = 1, \dots, N_j$ , represents the mean of the  $k$ th Gaussian membership function in the  $j$ th dimension;  $\sigma_j$  represents the standard deviation of the Gaussian membership functions in the  $j$ th dimension; and  $\lambda$ , selected in  $[0, 1]$ , represents the overlap parameter between membership functions.

**Step 3:** Map the numerical weight  $\mathbf{W}_P$  into linguistic label form using the following equation:

$$A_{ij} = \arg \min_k \left( \left\| (w_{p_{ij}}, \mu_{jk}) \right\| \right), \quad (3.26)$$

where  $A_{ij}$  represents the index of the linguistic label mapped from  $w_{p_{ij}}$ ; and  $w_{p_{ij}}$ , for  $i = 1, \dots, L, j = 1, \dots, M, k = 1, \dots, N_j$ , is an element of the hidden weight  $\mathbf{W}_P$  of the ILFN network.  $M$  is the dimension of the pattern space and  $L$  is the number of prototypes created by the ILFN network.

**Step 4:** Generate if-then rule table: use linguistic antecedent parts obtained from  $\mathbf{W}_P$  and consequent parts from  $\mathbf{W}_T$ . The number of fuzzy if-then rules is equal to the number of hidden neurons of the trained ILFN. Calculate

confident factor  $CF_i$ ,  $i = 1, \dots, L$ , for each rule using count parameter **count** by the following equation:

$$\mathbf{count} = [cnt_1, cnt_2, \dots, cnt_L]^T, \quad (3.27)$$

$$CF_i = \frac{cnt_i}{\sum_{h \in \text{Class}(w_{Ti})} cnt_h}, \quad (3.28)$$

where  $cnt_i$ ,  $i = 1, \dots, L$ , is a count parameter of the  $i$ th rule (i.e., the  $i$ th prototype) obtained when a pattern is included into the  $i$ th prototype; and  $\text{Class}(w_{Ti}) = \{l \mid w_{Ti} = w_{Ti}, l = 1, \dots, L\}$ .

$$\mathbf{CF} = [CF_1, CF_2, \dots, CF_L]^T \quad (3.29)$$

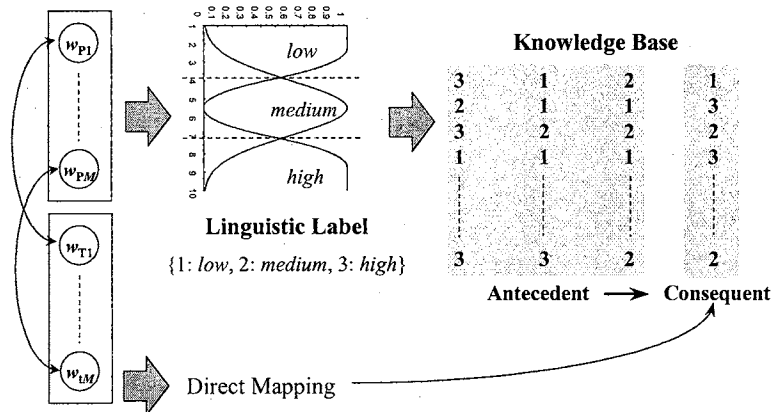


Figure 3.10: Mapping from ILFN to Linguistic Rules

The knowledge base from Figure 3.10 can be described by fuzzy linguistic form that is similar to natural language as follows:

Rule 1: If feature<sub>1</sub> is *high* and feature<sub>2</sub> is *low* and feature<sub>3</sub> is *medium*, then class is 1;

Rule 2: If feature<sub>1</sub> is *medium* and feature<sub>2</sub> is *low* and feature<sub>3</sub> is *low*, then class is 3;

Rule 3: If feature<sub>1</sub> is *high* and feature<sub>2</sub> is *medium* and feature<sub>3</sub> is *medium*, then class is 2;

...

After linguistic rules are extracted from the ILFN network, they can be used as a rule base for a fuzzy expert system. A fuzzy expert system is considered as a higher-level knowledge representation since it uses if-then rules similar to natural languages. Using linguistic form makes the system transparent, allowing human users to easily comprehend the rationale of how the decision was made. Explanations and answers can be provided if needed.

In pattern classification problems, the dimension of the pattern space may be very large. For a problem with a very large dimension, it is too cumbersome to use all the features available as a knowledge base. Though it is described in linguistic form, using all available features, it results in a system that is no longer transparent to users. It is possible to select only a feature subset that provides the most discriminatory power in classifying patterns. To do this, the genetic algorithm is very useful and suitable to select the important features. We will adapt the genetic algorithm (GA) [Holland75] to search for an optimal set of features used for each rule while maintaining a high percentage of correct classification. This will result in reducing the number of rules as well. Some rules will be redundant after many features have been eliminated, these duplicated rules can be pruned out.

### 3.3.2 Genetic Algorithm for Rule Optimization

The linguistic rule base extracted from the ILFN is clearly not *optimal*. (An *optimal* linguistic rule base implies that, for the same percentage of correctly classified training patterns, the number of rules and the number of fuzzy sets cannot be further reduced. An optimal linguistic rule base provides the most transparent fuzzy sets to the user.) In order to obtain a near optimal rule set, the GA is used to operate on initial fuzzy rules. An integer chromosome representation is used instead of a binary chromosomes representation, to reduce the size of chromosome and to improve the speed of the evolutionary operations. The fuzzy if-then rules are encoded into integer chromosomes to be evolved by the GA. After converging, the best chromosomes are decoded back into the FES with a compact rule set. (Please note that the GA optimization procedure here is not performed or related to ILFN network; it is used to change the rule structure of the FES to get near a optimal rule set.)

In order to apply the genetic optimization, the if-then rule base is encoded in a chromosome representation. Only the antecedent is coded and operated on by the evolutionary process. The original rule set is used as a reference rule set in decoding the most fitted individual to the final linguistic rule base.

### 3.3.3 Fuzzy If-Then Rule Encoding

In the proposed procedure, only the antecedents of the if-then rules are used in genetic encoding. If-then rules are encoded to an integer chromosome. A chromosome sometimes refers to an individual of the population. The elements of each chromosome

are called genes that are integer numbers. Each gene in a chromosome can be decoded to a fuzzy if-then rule. Let  $\mathbf{G}_R$  be a chromosome that is a set of genes  $g_i, i = 1, \dots, L$ , where

$$\mathbf{G}_R = \{ g_1, g_2, \dots, g_L \} \quad (3.30)$$

$$g_i = \sum_{j=1}^M 2^{j-1} a_{ij} \quad (3.31)$$

$$a_{ij} = \begin{cases} 0 & \text{if } A_{ij} = 0 \\ 1 & \text{otherwise} \end{cases}; i = 1, \dots, L, j = 1, \dots, M, \quad (3.32)$$

where  $L$  is the number of fuzzy if-then rules;  $M$  is the dimension of the pattern space. The antecedent  $A_{ij}$  is the linguistic label in the  $j$ th dimension of the  $i$ th rule.  $a_{ij}$  is equal to “1” meaning that the  $j$ th dimension of the  $i$ th rule is being used and  $a_{ij}$  is equal to “0” meaning that the  $j$ th dimension of the  $i$ th rule is not being used, i.e., *don't care*. Note that equation (3.31) is used to transform binary numbers to decimal numbers, which is the reason why the term  $2^{j-1}$  is used.

For example, a FES is used in a three-class, two-dimensional pattern space. The fuzzy expert system has two linguistic labels {1: *low*, 2: *high*} in each dimension. Suppose that there are four rules extracted from a trained ILFN, as follows.

Rule 1: if  $x_1$  is *low* and  $x_2$  is *high*, then class 1;

Rule 2: if  $x_1$  is *high* and  $x_2$  is *high*, then class 2;

Rule 3: if  $x_1$  is *low* and  $x_2$  is *low*, then class 3;

Rule 4: if  $x_1$  is *high* and  $x_2$  is *low*, then class 3.

That is we have  $\mathbf{R} = \{ \mathbf{A}, \mathbf{B} \}$ ,

$$\mathbf{A} = \begin{Bmatrix} 1 & 2 \\ 2 & 2 \\ 1 & 1 \\ 2 & 1 \end{Bmatrix}, \mathbf{B} = \begin{Bmatrix} 1 \\ 2 \\ 3 \\ 3 \end{Bmatrix}; \text{ where } \mathbf{A} \text{ is the antecedent set and } \mathbf{B} \text{ is the consequent}$$

set.

Let  $\mathbf{A}_{dc}$  be a set of antecedents when some features are composed of a *don't care* linguistic label. Let  $\mathbf{A}_{bi}$  be a binary set of 1's and 0's indicating whether or not an element of  $\mathbf{A}$  is used.

$$\text{Suppose the antecedent set } \mathbf{A} \text{ is reduced to } \mathbf{A}_{dc} = \begin{Bmatrix} 1 & 2 \\ 2 & 2 \\ 0 & 0 \\ 0 & 1 \end{Bmatrix}.$$

$$\text{That is we have a binary set } \mathbf{A}_{bi} = \begin{Bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 1 \end{Bmatrix}.$$

From the binary set  $\mathbf{A}_{bi}$ , we have an encoded chromosome

$$\begin{aligned} \mathbf{G}_R &= \{(2^{1-1} \times 1 + 2^{2-1} \times 1), (2^{1-1} \times 1 + 2^{2-1} \times 1), (2^{1-1} \times 0 + 2^{2-1} \times 0), (2^{1-1} \times 0 + 2^{2-1} \times 1)\} \\ &= \{(1+2), (1+2), (0+0), (0+2)\} \\ &= \{3, 3, 0, 2\}. \end{aligned}$$

The above numbers 3, 3, 0, and 2, are genes of a chromosome  $\mathbf{G}_R$ . Each gene represents an encoded fuzzy rule. With these encoded fuzzy rules, the GA can be operated on for rule optimization. After the GA optimization process, the final chromosomes, which are encoded fuzzy rules, can be decoded back to normal fuzzy rules.

### 3.3.4 Fuzzy If-Then Rule Decoding

Integer chromosomes are used in the genetic optimization process. After convergence of the solution, encoded integer chromosomes are decoded back to fuzzy if-then rule bases. Given an integer chromosome, each gene is decomposed into binary format. The decoding process is an inverse process of the encoding process mentioned above. The procedure for transforming from integer numbers to binary numbers can be performed as following steps:

**Step 1:** Retrieve an integer value and keep as  $g$ . Set index  $Q = 1$ .

**Step 2:** Divide 2 into  $g$ ; keep the remainder from the division as  $y_Q$ ; keep the answer from the division as  $g$ ;

**Step 3:** Check the answer if it is equal 0, go to Step 4; if it is not equal 0, set  $Q = Q + 1$  and repeat Step 2.

**Step 4:** Output the binary number  $b = \{y_1, y_2, \dots, y_Q\}$ .

For example, transforming integer number  $g = 5$  to binary form can be proceeded as follows

Step 1:  $g = 5$ ;  $Q = 1$ .

Step 2: Divide 2 into  $g$ ; the answer is 2 with the remainder 1; set  $g = 2$ ; set  $y_1 = 1$ .

Step 3:  $g$  is not 0;  $Q = 2$ ; repeat Step 2:

Divide 2 into  $g$ ; the answer is 1 and the remainder is 0; set  $g = 1$ ; set  $y_2 =$

0.  $g$  is not 0;  $Q = 3$ ; repeat Step 2:

Divide 2 into  $g$ ; the answer is 0 and the remainder is 1; set  $g = 0$ ; set  $y_3 =$

1.  $g$  is 0 go Step 4:

Step 4: Output the binary number  $b = \{y_1, y_2, y_3\} = \{1\ 0\ 1\}$

In a short way, we can transform an integer number  $g$  to a binary number  $b$  by expressing  $g$  into the form

$$\begin{aligned} g &= \{1 + 2 + 4 + \dots + 2^{M-1}\} = \{2^0 \times a_1 + 2^1 \times a_2 + 2^2 \times a_3 + \dots + 2^{M-1} \times a_M\} \\ &= \{2^{1-1} \times a_1 + 2^{2-1} \times a_2 + 2^{3-1} \times a_3 + \dots + 2^{M-1} \times a_M\} \\ &= \sum_{j=1}^M 2^{j-1} a_j; a_i \in \{0, 1\}. \end{aligned}$$

Finally, the binary number  $b$  can be written as  $b = \{a_1, a_2, \dots, a_M\}$ . For example, an integer number  $g = 5$  can be expressed in the form  $g = \{1 + 0 + 4\} = \{2^0 \times 1 + 2^1 \times 0 + 2^2 \times 1\}$ . Thus we have  $b = \{1\ 0\ 1\}$ .

Suppose that we have a solution chromosome  $\mathbf{G}_R = \{3, 3, 0, 2\}$ . Using the procedure for transforming from integer numbers to binary number,  $\mathbf{G}_R$  can be decomposed to binary format as follows:

$$\begin{aligned} \mathbf{G}_R &= \{3, 3, 0, 2\} \\ &= \{(1+2), (1+2), (0+0), (0+2)\} \\ &= \{(2^{1-1} \times 1 + 2^{2-1} \times 1), (2^{1-1} \times 1 + 2^{2-1} \times 1), (2^{1-1} \times 0 + 2^{2-1} \times 0), (2^{1-1} \times 0 + 2^{2-1} \times 1)\}. \end{aligned}$$

$$\text{So, we have } \mathbf{A}_{bi} = \begin{Bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 1 \end{Bmatrix}. \text{ Knowing the origin antecedent set } \mathbf{A} = \begin{Bmatrix} 1 & 2 \\ 2 & 2 \\ 1 & 1 \\ 2 & 1 \end{Bmatrix}, \text{ we}$$

$$\text{have the reduced antecedent set } \mathbf{A}_{dc} = \begin{Bmatrix} 1 & 2 \\ 2 & 2 \\ 0 & 0 \\ 0 & 1 \end{Bmatrix};$$



$$\mathbf{R} = \{\mathbf{A}_{dc}, \mathbf{B}\} = \left\{ \begin{array}{ccc} 1 & 2 & 1 \\ 2 & 2 & 2 \\ 0 & 0 & 3 \\ 0 & 1 & 3 \end{array} \right\} = \left\{ \begin{array}{ccc} 1 & 2 & 1 \\ 2 & 2 & 2 \\ 0 & 1 & 3 \end{array} \right\}.$$

Note that if a rule comprises of all *don't care* linguistic labels in the antecedent part, then that rule can be eliminated.

### 3.3.5 Genetic Selection for the Number of Linguistic Variables

The number of linguistic variables can be varied depending on a given problem. Some problems may need more linguistic variables than others. Using more linguistic variables results in finer fuzzy partitions and better classification performance. However, to maintain the interpretability of the system, the number of linguistic variable should be kept as small as possible. Selecting the numbers of linguistic variables becomes a trade off between the accuracy of the system and the interpretability of the system. To obtain the optimal point that balances between the accuracy and the interpretability is not an easy task. To avoid the difficulty, the numbers of linguistic labels can be selected by using the genetic algorithm. The genetic selection for the linguistic numbers can be processed simultaneously with the rule optimization.

The chromosome for the genetic optimization of the number of linguistic variables ( $\mathbf{G}_{NL}$ ) can be written as

$$\mathbf{G}_{NL} = \{N_1, N_2, \dots, N_M\} \quad (3.33)$$

where  $N_j, j = 1, \dots, M$ , is the number of linguistic variables for the  $j$ th dimension. The chromosome for optimizing the number of linguistic variables ( $\mathbf{G}_{NL}$ ) can be combined

with the chromosome for optimizing the fuzzy if-then rules ( $\mathbf{G}_R$ ). The combined chromosome from Equations (3.30) and (3.33) can be written as

$$\mathbf{G} = \{\mathbf{G}_{NL}, \mathbf{G}_R\} = \{N_1, N_2, \dots, N_M, g_1, g_2, \dots, g_L\} \quad (3.34)$$

When the genetic algorithm is implemented, it usually proceeds in a manner that involves the following steps:

Step 1 Initialization of the population

Step 2 Fitness evaluation

Step 3 Mate selection

Step 4 Crossover

Step 5 Mutation

Step 6 Check stopping criteria; if the solution meets the criteria, stop the algorithm and obtain the final if-then rules; otherwise, repeat Steps 2-6.

*Initialization of the population:* A chromosome has two different groups of genes: the number of linguistic variables and the fuzzy if-then rules. The initial population of the chromosomes is randomly selected as integer numbers in both of the groups. These initial individuals will be reproduced to next generation via the evolutionary operations: fitness evaluation, mate selection, crossover, and mutation.

*Fitness evaluation:* The fitness function is based on the performance of resulting rules decoded from a chromosome and the compactness of the rule set. A fuzzy expert system with the decoded rules is used to evaluate the performance of the resulting rules. The fitness function of a chromosome  $\mathbf{G}$  can be determined from the following equations:

$$fitness(\mathbf{G}) = W_{PC} \times PC - W_F \times SC - W_{NL} \times NL, \quad (3.35)$$

$$PC = \frac{\text{Total Patterns} - \text{Wrongly Classified Patterns}}{\text{Total Patterns}} \times 100, \quad (3.36)$$

$$SC = \sum_{i,j} a_{ij}, \quad (3.37)$$

$$NL = \sum_j N_j \quad (3.38)$$

where  $W_{PC}$  is the weight of percent correct classification by a fuzzy expert system;  $PC$  is the percent correct classification;  $W_F$  is the weight of the number of features used for a rule set;  $a_{ij}$  is calculated from Equation (3.32);  $SC$  is the structure complexity of the fuzzy system i.e., number of features used for a rule set, i.e., number of 1's in  $\mathbf{A}_{bi}$ ;  $W_{NL}$  is the weight of the number of linguistic variables used for a rule set; and  $NL$  is the summation of the numbers of linguistic variables used. Preferring fewer linguistic variables, fewer rules, and fewer features with higher correct classification performance, the weight of percent correctly classified patterns ( $W_{PC}$ ), is usually set to be relatively larger than the weight of structure complexity ( $W_F$ ) and the weight of the linguistic variables ( $W_{NL}$ ).  $W_F$ ,  $W_{PC}$ , and  $W_{NL}$  are all positive numbers in  $\mathfrak{R}$ ; they are predefined by the user.

*Mate selection:* There are many ways of selecting individuals for mating. One of the well-known methods is *roulette wheel selection* [Goldberg89]. The fittest individuals usually have a higher chance to mate than the ill-fitted ones. In roulette wheel selection, the individuals are randomly selected based on the probability of fitness. The reproduction probability can be defined from the fitness function.

$$prob(\mathbf{G}_i) = \frac{fitness(\mathbf{G}_i)}{\sum_{j=1}^P fitness(\mathbf{G}_j)}, \quad i = 1, \dots, P, \quad (3.39)$$

where  $P$  is the number of individuals in the population.

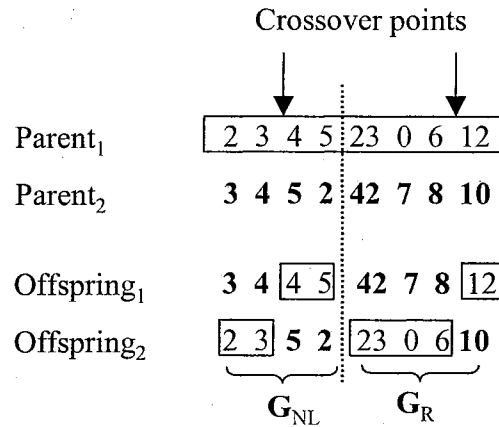


Figure 3.11: Crossover Operation

*Crossover:* After mate selection operation, crossover operation is performed. Crossover operation is a mechanism for changing information between two chromosomes called *parents* to reproduce two new individuals called *offspring*. A crossover point is selected randomly with probability  $p_c$ . In our problem, since a chromosome is separated into two groups, the crossover process is also separated into two parts: the crossover of the number of linguistic variables and the crossover of fuzzy if-then rules. The crossovers of the two parts are independent from each other. Figure 3.11 illustrates how two chromosomes crossover, yielding two offspring.

*Mutation:* Mutation is applied to offspring to prevent the solution from trapping at a local minimum area. The mutation operation allows the genetic algorithm to explore new possible solutions and increase a chance to get near global minima. Figure 3.12 illustrates how the mutation operation works.

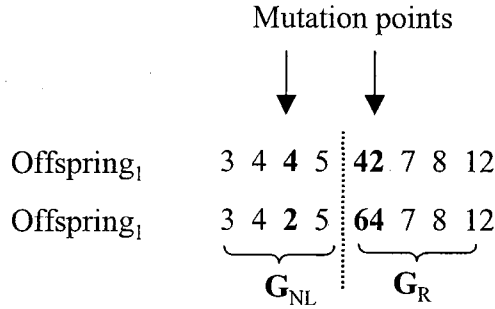


Figure 3.12: Mutation Operation

An offspring chromosome mutates with the mutation probability  $p_m$  on each gene. In the integer-coded genetic algorithm, the mutation process operates by the following equation

$$G_{new} = \text{round}(G_{old} + \gamma \times \text{randn}(1)), \quad (3.40)$$

where  $G_{old}$  is a gene selected for mutating;  $G_{new}$  is the resulted gene from mutating;  $\text{randn}(1)$  is a random number in  $[0, 1]$  produced by the Gaussian random number generator; and  $\gamma$  is the highest possible integer value a gene is allowed to be. The two parts of the chromosome  $\mathbf{G}$  have different values. The highest possible value of the number of linguistic variables is set to 9 or smaller. The highest integer value for the gene of the if-then rule is  $2^{M-1}$  for  $M$ -dimensional space.

### 3.4 Rule-To-Network Module

The rule-to-network module is used for transferring the linguistic knowledge into the ILFN structure. The rule-to-network module allows domain experts to incorporate their knowledge into the system. The rule-to-network consist of the *rule2ilfn* algorithm that is used for mapping the FES to the ILFN.

### 3.4.1 RULE2ILFN Algorithm

There are two phases in the rule2ilfn algorithm. Phase 1 is used when fuzzy rules are compact where they have *don't care* linguistic variables. The *don't care* linguistic variables need to be transformed into intermediate rules. In the transformed intermediate rules, every feature or component of the rules is composed of at least a linguistic variable attached; otherwise, we cannot map rules to an ILFN network. Phase 2 operates after phase 1 ended. In phase 2, the parameters of fuzzy rules are correspondingly mapped to the parameters of an ILFN network. The details of the two phases are shown as follows.

**Phase 1:** Mapping a compact rule set to an intermediate rule set.

- 1) Retrieve a rule  $R_i = \{A_{i1}, A_{i2}, \dots, A_{iM}, B_i, CF_i\}; i = 1, \dots, L$ .
- 2) Check for a feature that has a *don't care* linguistic label (i.e.,  $A_{ij} = 0$ ).  
Within the present rule, if there is a feature having a *don't care* linguistic label; expand every possible rule to cover the combinations of available linguistic labels.
- 3) Repeat 1) and 2), until there are no more rules.
- 4) Output the intermediate rule set.

**Phase 2:** Mapping an intermediate rule set to an initial ILFN network.

- 1) Set  $\mathbf{W}_P$ ,  $\mathbf{W}_T$ ,  $\mathbf{S}$ , and **count** to be empty sets.
- 2) For  $i$ th rule = 1 to  $L$  do,
  - For  $j$ th feature = 1 to  $M$  do,
    - Set  $w_{Pij} = \mu_{ij}$
    - Set  $std_{ij} = \sigma_{ij}$

- Set  $w_{Ti} = B_i$
  - Set  $cnt_i = 1$
- 3) Set  $\mathbf{W}_P = [\mathbf{w}_{P1}, \mathbf{w}_{P2}, \dots, \mathbf{w}_{PL}]^T$ ; where  $\mathbf{w}_{Pi} = [w_{Pi1}, w_{Pi2}, \dots, w_{PiM}]$
  - 4) Set  $\mathbf{W}_T = [w_{T1}, w_{T2}, \dots, w_{TL}]^T$
  - 5) Set  $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_L]^T$ ; where  $\mathbf{s}_i = [std_{i1}, std_{i2}, \dots, std_{iM}]$
  - 6) Set  $\mathbf{count} = [cnt_1, cnt_2, \dots, cnt_L]^T$

$L$  is the number of rules and  $M$  is the dimension of pattern space. The parameters  $\mu_{ij}$  and  $\sigma_{ij}$ ,  $i = 1, \dots, L, j = 1, \dots, M$ , are a mean and a standard deviation of the linguistic label indexed by  $A_{ij}$ . More specifically,  $\mu_{ij}$  and  $\sigma_{ij}$  are taken from  $\mathbf{mp}_{jA_{ij}}$  that is in the membership functions' parameters (**MP**) from Equation (3.13).

After obtaining the initial ILFN network, available training data is used to refine the ILFN network. Network pruning is also needed to eliminate the hidden nodes that do not have any belonging pattern. This can be done by checking at the parameter **count**. If count of a node is equal to one, then eliminate the node.

### 3.5 The Decision-Explanation Module

The last module in the HIS is the decision-explanation module. The decision-explanation module performs two functions: making a decision and explaining the decision. For a first function, making a decision, the decision-explanation module receives two inputs from the outputs of low-level ILFN and higher-level FES. Another function of the decision-explanation module is to generate a natural language to explain and conclude the decision made by using the knowledge base (**K**) from the FES module.

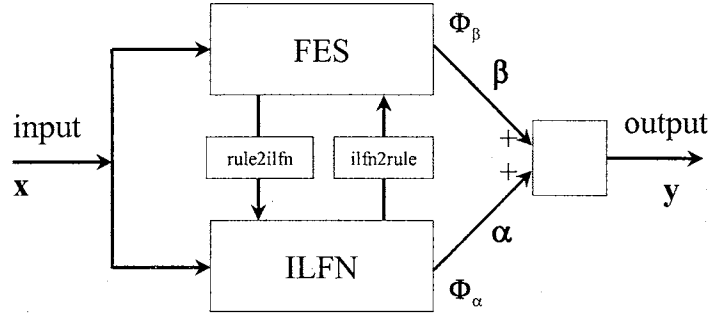


Figure 3.13: Hybrid System Combined from ILFN and FES

Figure 3.13 shows an equivalent of the HIS. The decision for the class output  $C_y$  can be calculated by the following equations:

$$\mathbf{C} = [C_1, C_2, \dots, C_Q] \quad (3.41)$$

$$\Phi_\alpha = [\phi_{\alpha 1}, \phi_{\alpha 2}, \dots, \phi_{\alpha Q}] \quad (3.42)$$

$$\Phi_\beta = [\phi_{\beta 1} \times CF_1, \phi_{\beta 2} \times CF_2, \dots, \phi_{\beta Q} \times CF_Q] \quad (3.43)$$

$$y_i = \frac{\alpha_i \phi_{\alpha i} + \beta_i \phi_{\beta i} \times CF_i}{\alpha_i + \beta_i}, \text{ for } i = 1, \dots, Q \quad (3.44)$$

$$\mathbf{y} = [y_1, y_2, \dots, y_Q] \quad (3.45)$$

$$C_y = C_J; J = \arg \max_i (y_i) \quad (3.46)$$

where  $\mathbf{C}$  is the class vector;  $\Phi_\alpha$  is the membership values from the ILFN with respect to  $\mathbf{C}$ ;  $\Phi_\beta$  is the membership values from the FES with respect to  $\mathbf{C}$ ;  $\mathbf{y}$  is the membership values from the HIS with respect to  $\mathbf{C}$ ;  $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_Q]$  and  $\beta = [\beta_1, \beta_2, \dots, \beta_Q]$  are the real-value weights linking from the ILFN and the FES to the decision-explanation module, respectively;  $Q$  is the number of classes; and  $C_y$  is the class decision output from the HIS. Please note that  $\alpha$  and  $\beta$  can be specified by the user or determined by an



optimization algorithm such as the GA. In our study we used a GA to search for possibly optimal values of  $\alpha$  and  $\beta$ .

For simplicity, we may set  $\alpha = \alpha_1 = \alpha_2 = \dots = \alpha_Q$  and  $\beta = \beta_1 = \beta_2 = \dots = \beta_Q$ . Then we have

$$y = \frac{\alpha \Phi_\alpha + \beta \Phi_\beta}{\alpha + \beta} \quad (3.47)$$

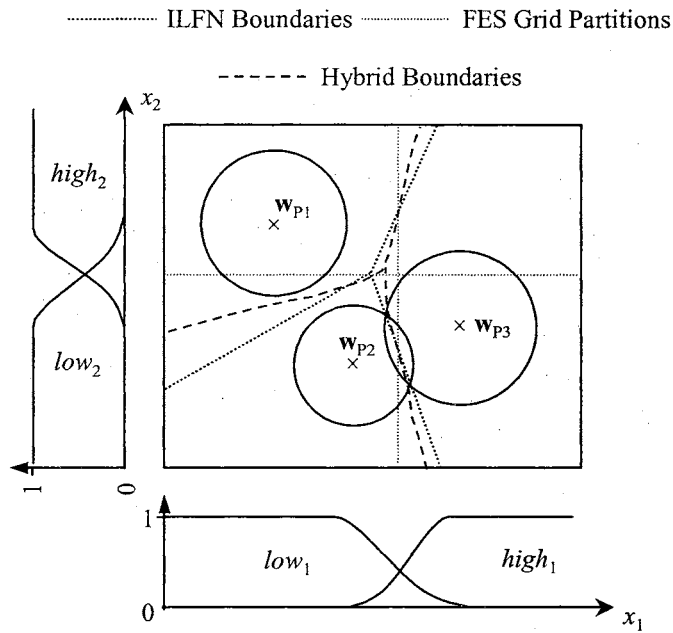


Figure 3.14: Hybrid Decision Boundaries

### 3.5.1 Decision Boundaries

The decision boundaries of the HIS come from the weighted average of the boundaries from the ILFN and FES. The decision boundaries of the ILFN and the decision boundaries of the FES contribute in different manners. The decision boundaries of the ILFN emphasize in local area to achieve better generalization while the decision boundaries of the FES preserve for human interpretability. Since numerical information

in the ILFN and linguistic information in the FES have complement benefits, it is preferable to incorporate both structures into the same system. The boundaries of the hybrid system provide both accuracy and interpretability. In the HIS, the ILFN serves as a low-level numerical computation, while the FES operates as a higher-level linguistic computation. Hybrid weights ( $\alpha$  and  $\beta$ ) play an important role in adjusting the hybrid decision boundaries. If  $\alpha_i$  is larger than  $\beta_i$ , the hybrid boundaries tend toward the ILFN boundaries. If  $\alpha_i$  is smaller than  $\beta_i$ , the hybrid boundaries tend toward the FES boundaries. Figure 3.14 shows the hybrid decision boundaries of the combined ILFN and FES.

### **3.5.2 Conflict and Conflict Resolution Between Low Level and Higher Level**

Ideally, there should be no conflict between low level and higher level decisions in the HIS, if it is a one-to-one mapping between them. Since a combination of grid based partition and projection is used in the mapping process, decisions from ILFN and FES may conflict with each other. The diagram in Figure 3.15 shows the possible conflict decisions between the ILFN and the FES. The system is in conflict, if the decision from the ILFN is correct but the decision from the FES is wrong or if the decision from the ILFN is wrong but the decision from the FES is correct. The system is not in conflict, if the two systems make the same decision. It is preferable that the two systems are not conflict and both make correct decisions.

It is feasible to resolve the conflict between the two systems by forcing their decision boundaries to be as close to the HIS decision boundaries as possible. Conflict resolution for the HIS is then the determination of the elements for  $\alpha$  and  $\beta$ . As a matter

of fact, this becomes an ordinary optimization problem, which can be solved by using any optimization method. Due to an advantage of not requiring a derivative calculation and unlikely to be trapped at local minima, GA can be adopted for this purpose. The GA searches for weights  $\alpha$  and  $\beta$  that adapt the decisions boundaries of the two modules to be closer together. The hybrid decision finally will be forced to the diagonal path indicated as a dashed line in Figure 3.15, which ideally shows that the ILFN and the FES are not conflicting and both make consistent decisions.

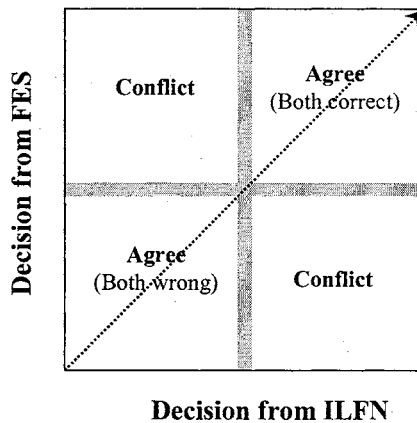


Figure 3.15: Conflict Decision between the ILFN and the FES

### 3.6 Increment Learning Characteristic of the Proposed HIS

An incremental learning system updates its new knowledge without training old data. Only new data is needed in the learning process. This concept has been studied by many researchers (see [Fu96], [Carpenter92], [Yen99], [Yen01].) In the hybrid structure between a low level and a higher level, such as in numerical and symbolic or numerical and linguistic systems, it is preferable to incorporate the incremental feature to the system. It is important in application such as controls and monitoring process, as well as in medical diagnosis, to employ an incremental learning aspect. New knowledge needs to

be captured in real time without spending tremendous time to learn all the old data along with the new ones.

Since the proposed HIS incorporated the ILFN which is equipped with an incremental learning architecture, it is easy to employ its incremental learning capability. The ILFN can learn all patterns within only one pass. While operating, the ILFN detects new unseen class prototypes. If new knowledge is found, the new knowledge is added in the hidden unit without forgetting the old knowledge. Incorporating the incremental learning feature to the higher-level linguistic model is straightforward. New linguistic rules can be directly extracted from the new hidden nodes of the ILFN by using the `ilfn2rule` algorithm in the network-to-rule module. An algorithm for checking conflict is operated to maintain consistency between the two levels. Similarly, if the higher level has a new knowledge, i.e., linguistic rules that maybe come from an expert or experienced users, the new knowledge needed to be mapped to the ILFN structures as well. This can be done by using the `rule2ilfn` algorithm in the rule-to-network module.

**CHAPTER IV**  
**QUANTITATIVE MEASURES ON THE ACCURACY,**  
**COMPREHENSIBILITY, AND COMPLETENESS**  
**OF A FUZZY KNOWLEDGE BASE**

Quantitative measures are essential and form the basis for making reliable decisions in software engineering including computational intelligence such as fuzzy expert systems (FESs). Quantitative assessment helps us to understand quality of FESs that are not accessible to our intuitive ability. Generally, quantitative assessments concerned when a FES is constructed are accuracy measures. Accuracy measures help us to judge how good a FES can perform in prediction of unseen data. In addition, since FESs provide a knowledge representation of the problems dealing with; accuracy alone may not be sufficient to guarantee the goodness of FESs [Setnes98], [Jin00], [Roubos01]. Comprehensibility measures are additional quantitative assessment that can assure that a FES is understandable. Moreover, a completeness measure is an indicator to check a fuzzy system whether its linguistic variables and rule structure cover the entire possible data domain [Jin99], [Stamou99], [Valente99]. The accuracy and the comprehensibility of a FES are discussed as follows.

**4.1 Accuracy Measures**

In a binary classification model there are two possible prediction errors: false positives (FP) and false negatives (FN). The performance of a binary classifier model is normally summarized in a confusion or contingency matrix that cross-tabulates the true and predicted classes. Contingency tables provide an easy method to determine

relationships between two variables. Based on the contingency table, we can interpret many performance measures such as accuracy, false alarm, sensitivity, and specificity.

TABLE 4.1:  
Binary-class Contingency Table

		Predicted Class		Total
		Yes	No	
True Class	Yes	True positive (TP)	False negative (FN)	TP + FN
	No	False positive (FP)	True negative (TN)	FP + TN
Total		TP + FP	FN + TN	TP + FN + FP + TN

The most common assessment of the performance of a classifier system is to test its *accuracy*. Accuracy is a measure of a predictive model that reflects the proportionate number of times that the model is making correct classification when applied to test data. It measures the probability that the system can correctly classify the data. In contrast to the accuracy measure, *false alarm* or *misclassification rate* measures the probability that the system wrongly classify the data. False alarm ( $F_A$ ) and accuracy ( $A_C$ ) may be used interchangeably.

$$A_C = \frac{TP + TN}{TP + FN + FP + TN} \quad (4.1)$$

$$F_A = \frac{FP + FN}{TP + FN + FP + TN} \quad (4.2)$$

*Sensitivity* ( $S_E$ ) and *specificity* ( $S_P$ ) are other two measures that are commonly used to assess the accuracy of a diagnostic test. Sensitivity is the proportion of all positive

classes that actually are correctly classified as positive in a test. For example in a medical diagnosis, sensitivity may measure the number of people who truly have the disease and who test positive. Specificity is the proportion of all negative classes that actually are correctly classified as negative in a test. In a medical diagnosis, specificity may measure the number of people who do not have the disease and who is tested negative.

$$S_E = \frac{TP}{TP + FN} \quad (4.3)$$

$$S_P = \frac{TN}{FP + TN} \quad (4.4)$$

TABLE 4.2:  
Multi-class Contingency Table

Class		Predicted Class					Total
		1	2	3	...	L	
True Class	1	T <sub>11</sub>	F <sub>12</sub>	F <sub>13</sub>	...	F <sub>1L</sub>	N <sub>row<sub>1</sub></sub>
	2	F <sub>21</sub>	T <sub>22</sub>	F <sub>23</sub>	...	F <sub>2L</sub>	N <sub>row<sub>2</sub></sub>
	3	F <sub>31</sub>	F <sub>32</sub>	T <sub>33</sub>	...	F <sub>3L</sub>	N <sub>row<sub>3</sub></sub>
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	L	F <sub>L1</sub>	F <sub>L2</sub>	F <sub>L3</sub>	...	T <sub>LL</sub>	N <sub>row<sub>L</sub></sub>
Total		N <sub>col<sub>1</sub></sub>	N <sub>col<sub>2</sub></sub>	N <sub>col<sub>3</sub></sub>	...	N <sub>col<sub>L</sub></sub>	N <sub>Total</sub>

For multi-class problems, the contingency or confusion matrix can be shown in Table 4.2. Table 4.2 shows a general contingency table of multi-class problems. Multi-class contingency tables provide a way to determine relationships among multiple variables. The rows of the contingency table indicate the true classes. The columns show the predicted classes. In the diagonal of the contingency table, T<sub>ii</sub>, i = 1, ..., L, means that true class i is identified as class i. For the off diagonal of the table, F<sub>ij</sub>, i = 1, ..., L, j = 1,

...,  $L$ , is a false alarm where true class  $i$  is identified as class  $j$ .  $N_{row_i}$ ,  $i = 1, \dots, L$ , is the summation of patterns in the  $i$ th row and it is the number of all true class  $i$ .  $N_{col_j}$ ,  $j = 1, \dots, L$ , is the summation of the patterns in the  $j$ th column and it is the total number of predicted class  $j$ .  $N_{Total}$  is the total number of the patterns. From Table 4.2, the accuracy ( $A_C$ ) measure can be determined from the following equation:

$$\begin{aligned}
 A_C &= \frac{\text{Number of Correctly Classified Patterns}}{\text{Total Number of Patterns}} \\
 &= \frac{\sum_{i=1}^L T_{ii}}{N_{Total}} \\
 &= \frac{\sum_{i=1}^L T_{ii}}{\sum_{i=1}^L (N_{row_i})} \\
 &= \frac{\sum_{i=1}^L T_{ii}}{\sum_{i=1}^L (N_{col_i})} \tag{4.5}
 \end{aligned}$$

where  $T_{ii}$  is the number of patterns of true class  $i$  that are identified as class  $i$ .

## 4.2 Comprehensibility Measures

Comprehensibility is an important issue for a fuzzy system. It is one indication for the goodness of a fuzzy system. It can tell whether a fuzzy system is understandable. Practically, it is preferable to design a fuzzy system which is highly comprehensible in term of knowledge representation. A highly comprehensible fuzzy system can be easy to



draw reasoning from. Comprehensibility of fuzzy systems involves the compactness of fuzzy systems, the similarity between linguistic terms, and the consistency of fuzzy rules.

#### 4.2.1 Compactness

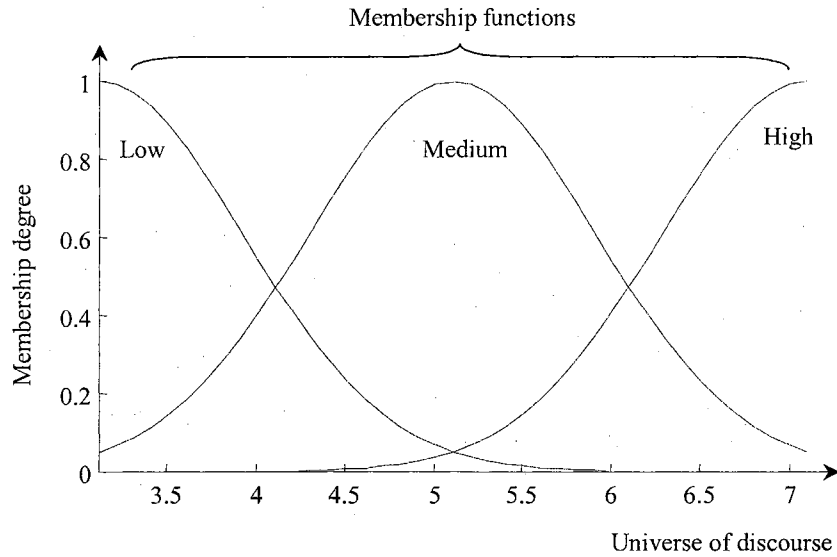


Figure 4.1: Higher Comprehensible Fuzzy System with Few Linguistic Terms

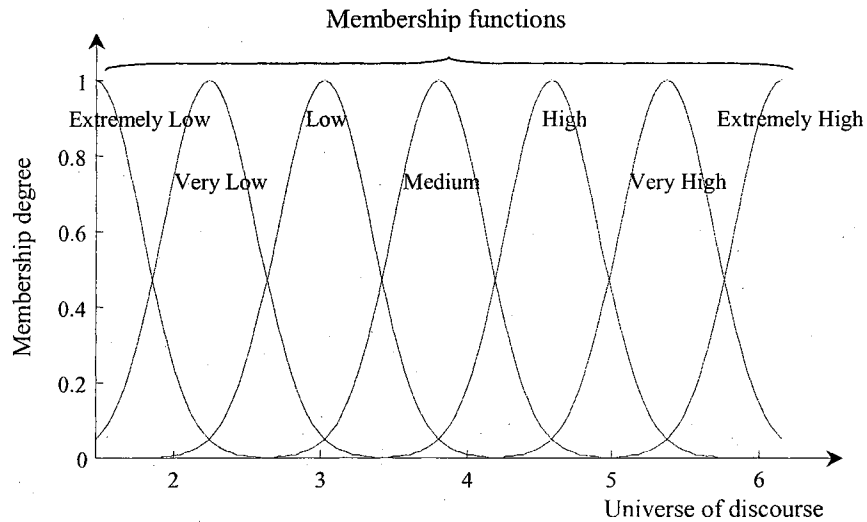


Figure 4.2: Low Comprehensible System with Too Many Linguistic Terms

Compactness of fuzzy systems is associated with the comprehensibility of fuzzy systems. A compact fuzzy system implies that the fuzzy systems are easy to comprehend.

Compactness of fuzzy systems relates to three aspects: a small number of linguistic terms in each dimension, a small number of fuzzy rules in the rule base, and a small number of conditions in the rule premise or antecedent part [Jin99], [Roubos2001].

For the first aspect of compactness regarding the number of linguistic terms, Figures 4.1 and 4.2 illustrate the compactness concept of fuzzy systems. Figures 4.1 and 4.2 differ in the number of linguistic terms. Comparing between Figures 4.1 and 4.2, Figure 4.1 has fewer linguistic terms. In general, it is relatively easier for the user to discern a fuzzy variable with three rather than seven linguistic labels.

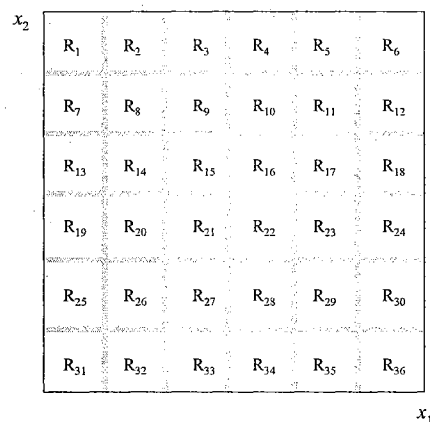


Figure 4.3: A Structure of Two-Dimensional Fuzzy System with Too Many Fuzzy Rules

The second aspect of compactness is the number of fuzzy rules. The number of fuzzy rules needed to represent a physical system depends on the structure of the fuzzy rules. In a standard structure of a fuzzy system with  $M$  dimensions and each dimension partitioned into  $N$  subspaces, there exist up to  $N^M$  rules in the fuzzy system. For example, in Figure 4.3, for a two-dimensional fuzzy system partitioned into 6 subspaces, the number of fuzzy rules is 36. If all the possible rules are used then the system is not compact. For the same fuzzy system, a more compact fuzzy system is shown in Figure 4.4. A compact rule set is easier to comprehend and recognize. Compactness of fuzzy

rules increases the degree of importance when the system increases the number of dimensions or the number of input features [Jin99].

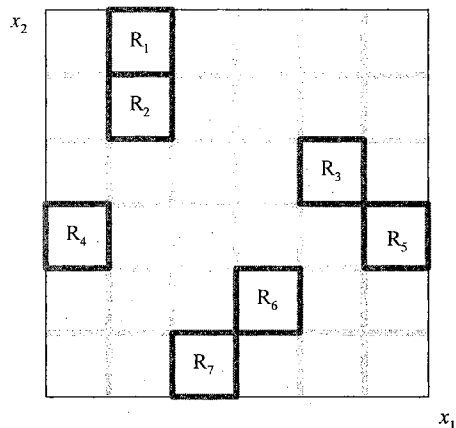


Figure 4.4: A Structure of Two-Dimensional Fuzzy System with Fewer Fuzzy Rules

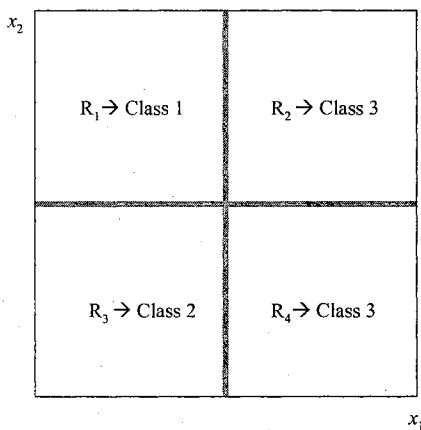


Figure 4.5: A Two-dimensional Fuzzy System with Two Conditions Per Rule

The third aspect of compactness is the number of conditions in the antecedent part of fuzzy rules or the number of features used per rule. If some of the features are not used then the system becomes more compact. The system structure can be easier to comprehend. Figure 4.5 shows a two-dimensional fuzzy system with four rules. In Figure 4.5, the number of conditions is two. Each rule uses both inputs as conditions in the

antecedent part. Figure 4.6 illustrates the same fuzzy system with three rules, but the number of conditions per rule is 1.67. Rules 1 and 3 use both inputs  $x_1$  and  $x_2$  in the antecedent part while Rule 2 uses only input  $x_1$ . Figure 4.6 has a structure that is easier to comprehend and recognize.

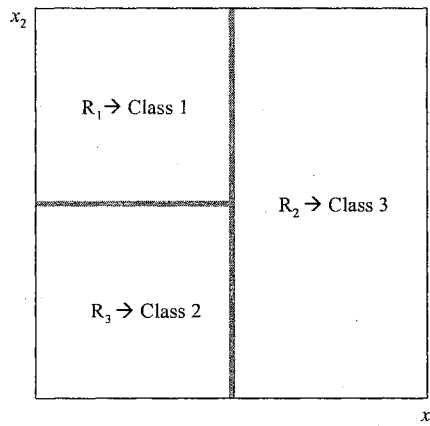


Figure 4.6: A More Compact Fuzzy System with 1.67 Conditions Per Rule

The compactness of a fuzzy system can be quantified into numerical values. The comprehensibility can be measured in several terms: the number of rules ( $L$ ), the number of antecedents per rule ( $N_A$ ), the number of labels per dimension ( $N_L$ ), the degree of linguistic similarity ( $LS$ ), and the degree of inconsistency or rule similarity ( $RS$ ).

$$L = \text{counts of all the rules in the rule set} \quad (4.6)$$

$$N_A = \frac{\text{counts of all the antecedents in the rule set}}{L} \quad (4.7)$$

$$N_L = \frac{\text{counts of all the linguistic labels}}{M} ; \quad (4.8)$$

where  $M$  is the number of dimensions.

## 4.2.2 Linguistic Similarity

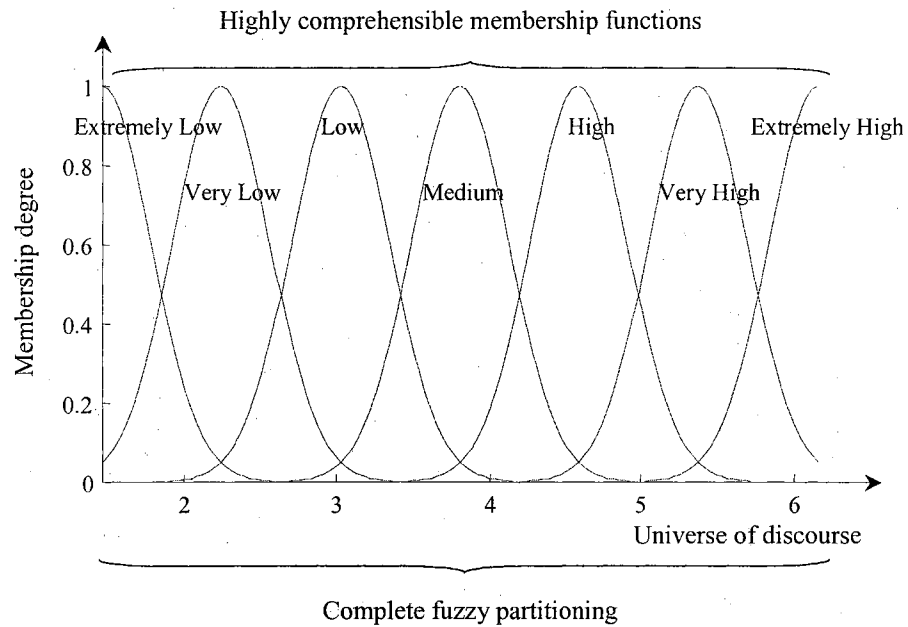


Figure 4.7: Comprehensible Linguistic Terms and Complete Fuzzy Partitions

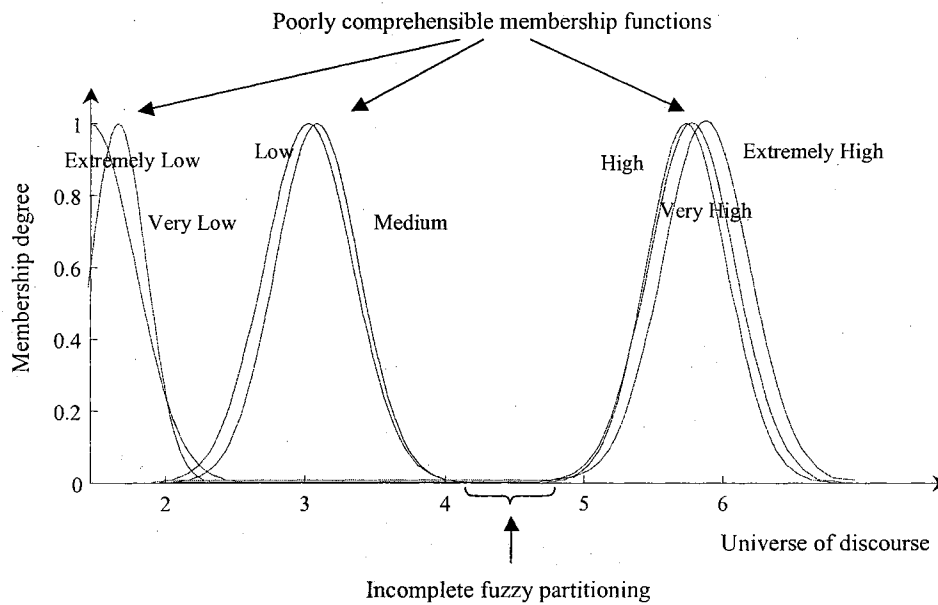


Figure 4.8: Poorly Comprehensible Linguistic Terms and Incomplete Fuzzy Partitions

Linguistic similarity of fuzzy variables is an important factor for the comprehensibility of fuzzy systems. Figures 4.7 and 4.8 illustrate two different sets of fuzzy linguistic terms, i.e., two different sets of membership functions. Both figures have the same number of linguistic terms that are *extremely low*, *very low*, *low*, *medium*, *high*, *very high*, and *extremely high*. One can notice that the membership functions in Figure 4.7 are more comprehensible than the membership functions in Figure 4.8. The membership functions in Figure 4.7 are equally distributed in the universe of discourse. Each membership function in Figure 4.7 is easily distinguished from the others. However, in Figure 4.8, the membership functions are not well comprehensible because there are some membership functions that are very similar to each other. They cannot be easily discriminated among others. The similar membership function should be eliminated or merged together [Setnes98], [Jin99], [Jin2000], [Roubos2001].

A similarity measure [Setnes98], [Jin2000] for fuzzy sets can be used to quantify the comprehensibility of a fuzzy knowledge base. The degree of linguistic similarity is considered the highest when two fuzzy sets are equal. When there are no overlapping fuzzy sets, the degree of linguistic similarity is zero. The degree of linguistic similarity falls in  $[0, 1]$ , if there are overlapping fuzzy sets. Based on the set-theoretic operations of intersection and union, we can determine the degree of linguistic similarity ( $LS$ ) of fuzzy sets by the following equation:

$$LS_j(k_1, k_2) = \frac{|l_{jk_1} \cap l_{jk_2}|}{|l_{jk_1} \cup l_{jk_2}|} \quad (4.9)$$

where  $|l_{jk}| = \sum_q m_{l_{jk}}(x_q)$  is the summation of membership value of linguistic label  $l_{jk}$ ; all  $x_q$  are crisp members of fuzzy set  $l_{jk}$ ;  $m$  defines the membership degree;  $j$  represents the

$j$ th dimension;  $k_1$  and  $k_2$  are the indexes to linguistic labels; and the  $\cap$  and  $\cup$  operators represent the intersection and union, respectively. Using the operator  $\cap$  as *min* and the operator  $\cup$  as *max*, we have the following degree of linguistic similarity ( $LS$ ):

$$LS_j(k_1, k_2) = \frac{\sum_q \min[m_{l_{jk_1}}(x_q), m_{l_{jk_2}}(x_q)]}{\sum_q \max[m_{l_{jk_1}}(x_q), m_{l_{jk_2}}(x_q)]} \quad (4.10)$$

$$LS_j = \frac{1}{(N_j - 1) + (N_j - 2) + \dots + 1} \sum_{k_1=1}^{N_j-1} \sum_{k_2=2}^{N_j} LS_j(k_1, k_2);$$

for  $k_1 \neq k_2; j = 1, \dots, M.$  (4.11)

$$LS = \frac{1}{M} \sum_{j=1}^M LS_j, \quad (4.12)$$

where  $LS_j(k_1, k_2) \in [0, 1]$  is the degree of linguistic similarity between linguistic labels  $l_{jk_1}$  and  $l_{jk_2}$ ;  $N_j$  is the number of linguistic labels in the  $j$ th dimension;  $k_1$  and  $k_2$  are the indexes to linguistic labels;  $m_{l_{jk}}$  denotes the membership function of linguistic labels  $l_{jk}$ ;  $x_q, j = 1, \dots, M, q = 1, \dots, Q$ , is the  $q$ th input sample in the  $j$ th dimension;  $M$  is the number of dimension;  $Q$  is the total number of input samples in the universe of discourse; and  $LS_j \in [0, 1]$  is the average of the degree of linguistic similarity in the  $j$ th dimension.

Figures 4.9 and 4.10 illustrate the intersection and the union, respectively, between two fuzzy sets.

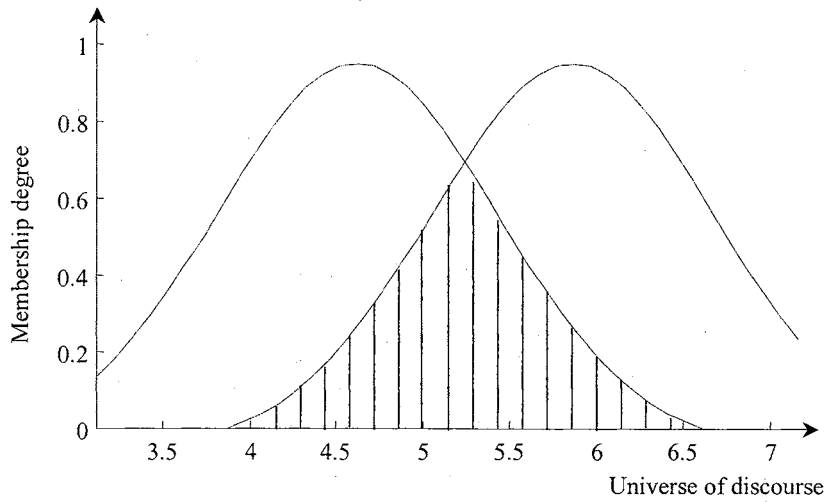


Figure 4.9: Intersection of  $I_{jk_1}$  and  $I_{jk_2}$

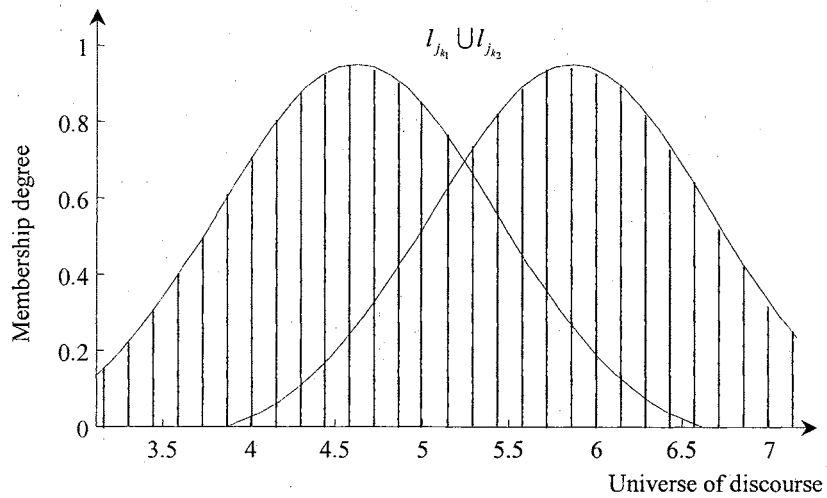


Figure 4.10: Union of  $I_{jk_1}$  and  $I_{jk_2}$

### 4.2.3 Consistency of Fuzzy Rules

Fuzzy rules should be consistency with each other, i.e., they are not conflicting with each others [Jin99]. Inconsistency of fuzzy rules can directly effect to the overall decision-making of the system. It can degrade the overall performance of the system. Inconsistency of fuzzy rules should be avoided. Inconsistency of fuzzy rules occurs when there are two or more rules are conflicting. Fuzzy rules are conflicting if they have



similar antecedents but rather different consequents. Measuring rule inconsistency is equivalent to measuring rule similarity. Degree of fuzzy rule similarity can be measured by using fuzzy similarity measure. Fuzzy rule similarity ( $RS$ ) is divided into two parts: the similarity of the antecedents ( $SA$ ) and the similarity of the consequents ( $SC$ ). The similarity of the antecedents ( $SA$ ) can be determined from the following equation:

$$SA_j(R_i, R_k) = \frac{\sum_q \min[m_{l_{jA_{ij}}}(x_q), m_{l_{jA_{kj}}}(x_q)]}{\sum_q \max[m_{l_{jA_{ij}}}(x_q), m_{l_{jA_{kj}}}(x_q)]} \quad (4.13)$$

where  $A_{ij} \in \{0, 1, 2, \dots, N_j\}$  is the index to the linguistic terms of the  $i$ th rule in the  $j$ th dimension;  $A_{kj} \in \{0, 1, 2, \dots, N_j\}$  is the index to the linguistic terms of the  $k$ th rule in the  $j$ th dimension; and  $N_j$  is the total number of linguistic labels in the  $j$ th dimension.

Using constant numbers as consequents, the similarity of the consequents ( $SC$ ) can be determined from the following equation:

$$SC(R_i, R_k) = \begin{cases} 1 & \text{if the consequents are the same;} \\ 0 & \text{otherwise.} \end{cases} \quad (4.14)$$

$$RS(R_i, R_k) = \frac{1}{M+1} \left( \sum_j SA_j(R_i, R_k) + SC(R_i, R_k) \right);$$

for  $i \neq k$ ;  $i = 1, \dots, L-1$ ;  $k = 2, \dots, L$ ;  $j = 1, \dots, M$ . (4.15)

$$RS = \frac{1}{(L-1) + (L-2) + \dots + 1} \sum_{i=1}^{L-1} \sum_{k=2}^L RS(R_i, R_k); \text{ for } i \neq k, \quad (4.16)$$

where  $RS(R_i, R_k) \in [0, 1]$  is the degree of linguistic similarity between rules  $R_i$  and  $R_k$ ;

and  $RS \in [0, 1]$  is the average of the degree of rule similarity.

### 4.3 Completeness Measure

Completeness is a property of deductive systems that has been used in the context of artificial intelligence to indicate that the knowledge representation scheme can represent every entity within the intended domain. In a fuzzy system, completeness is a fundamental issue since complete fuzzy systems can respond to any given input. A complete fuzzy system can achieve a proper operation avoiding undesirable situations [Stamou99], [Valente99]. The completeness of fuzzy systems consists of two main factors: completeness of fuzzy partitions and completeness of fuzzy rule structure [Jin99]. Examples of complete and incomplete fuzzy partitions are shown in Figures 4.7 and 4.8, respectively. Examples of complete and incomplete fuzzy rule structures are shown in Figures 4.11a and 4.11b, respectively.

To measure the completeness and incompleteness of fuzzy rule structure, suppose input variable  $x$  in the universe of discourse  $X$  is divided into  $N$  fuzzy partitions represented by membership functions  $m_i(x)$ , for  $i = 1, \dots, N$ . The completeness of the system is satisfied if

$$\forall x \in X, \exists i: 1 \leq i \leq N \text{ such that } m_i(x) > 0. \quad (4.17)$$

More generally, one may defined a certain *level of completeness*,  $\delta$ , given rise to the concept of *strong completeness*, as follows:

$$\forall x \in X, \exists i: 1 \leq i \leq N \text{ such that } m_i(x) > \delta. \quad (4.18)$$

Figures 4.11a and 4.11b, respectively, illustrate complete and incomplete fuzzy rule structures. In Figure 4.11a, the rule structure is complete because every partition in

each dimension is incorporated. Though the fuzzy partitions are complete, the rule structure in Figure 4.11b is incomplete because some partitions are not incorporated. The input in which its partitions are not used may cause a *no-response* or zero output.

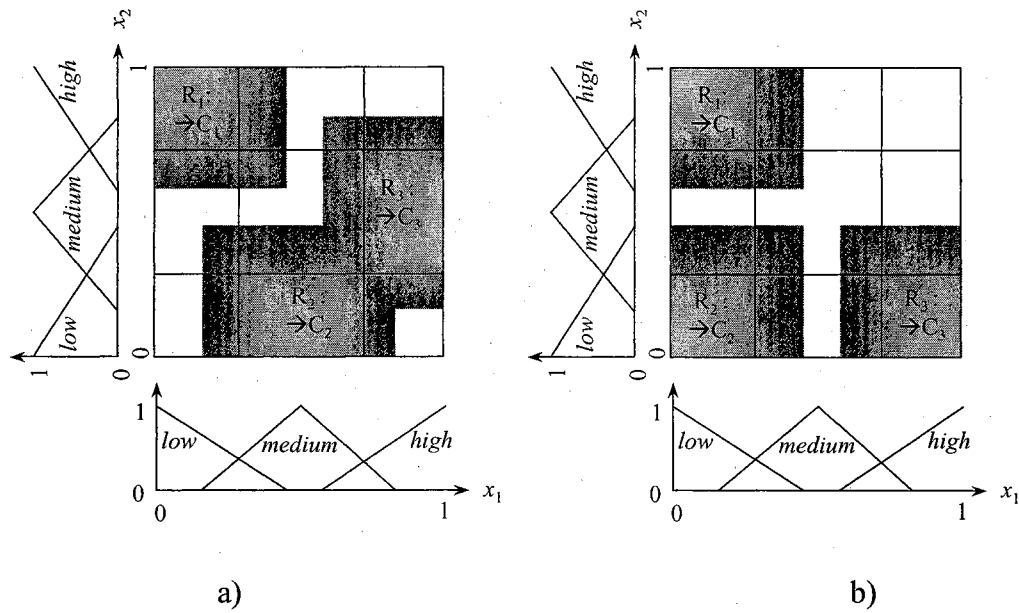


Figure 4.11: a) A Complete Rule Structure; b) An Incomplete Rule Structure

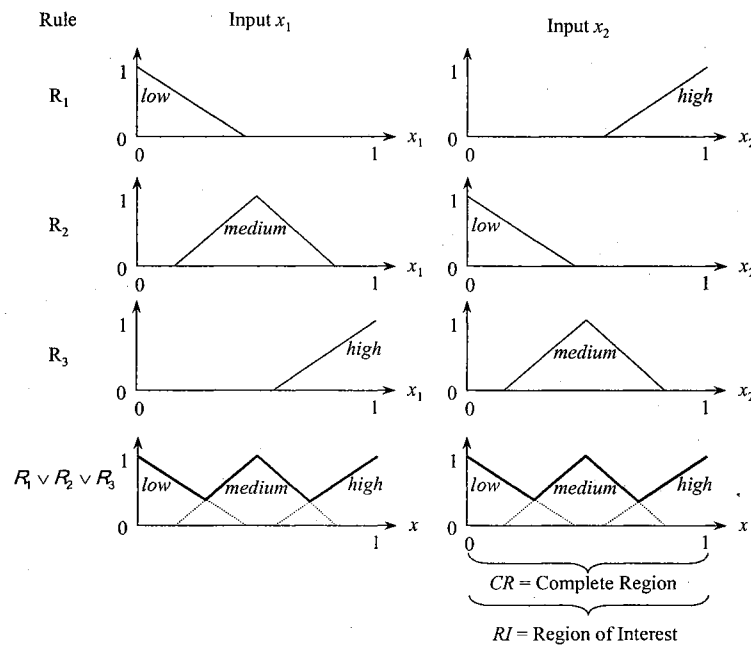


Figure 4.12: Antecedent Structure of a Complete Fuzzy System

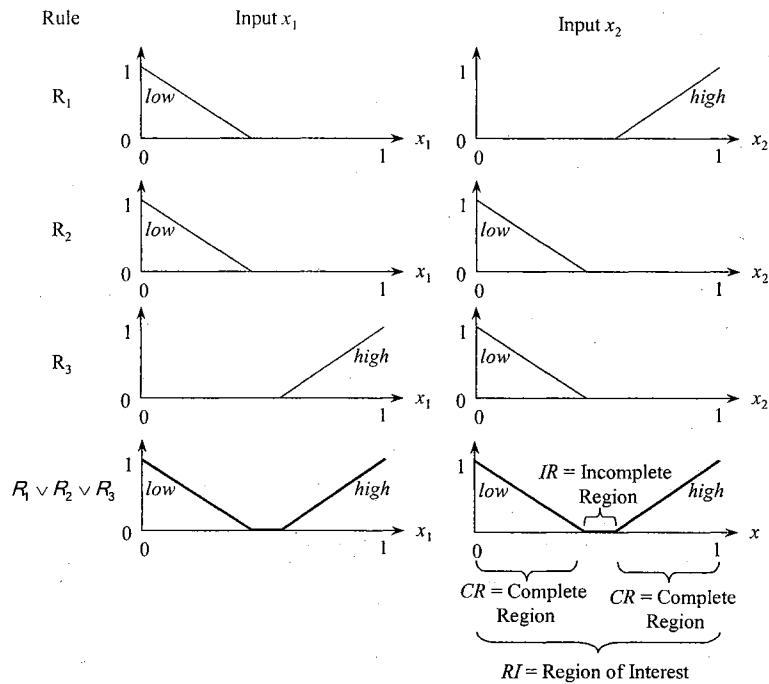


Figure 4.13: Antecedent Structure of an Incomplete Fuzzy System

To check whether or not the rule structure is complete, each dimension of fuzzy sets from all rules are mapped onto the same axis by OR operator ( $\vee$ ), as shown in Figures 4.12 and 4.13. Figure 4.12 shows the antecedent structure of a complete fuzzy system drawn from Figure 4.11a, while Figure 4.13 illustrates the antecedent structure of an incomplete fuzzy system from Figure 4.11b.

A completeness measure is defined as the proportion of the complete region and the region of interest. Similarly, an incompleteness measure is defined as the proportion of the incomplete region and the region of interest. Completeness degree in the  $j$ th dimension ( $CD_j$ ) and incompleteness degree in the  $j$ th dimension ( $ID_j$ ) are calculated from the following equations:

$$CD_j = \frac{CR_j}{RI_j} = \frac{N_{m(x) \geq \delta, x \in X}}{N_{x \in X}} \quad (4.19)$$

$$ID_j = \frac{IR_j}{RI_j} = \frac{N_{m(x) < \delta, x \in X}}{N_{x \in X}} = 1 - CD_j \quad (4.20)$$

$$I_D = \frac{\sum_{j=1}^M ID_j}{M} \quad (4.21)$$

where  $I_D$  is the overall incompleteness degree which is the average values of all the incompleteness degrees from each dimension.  $M$  is the number of the dimensions.  $CD_j$  and  $ID_j \in [0, 1]$  are completeness degree and incompleteness degree, respectively, in the  $j$ th dimension;  $CR_j$  is the length of the complete region in the  $j$ th dimension;  $IR_j$  is the length of the incomplete region in the  $j$ th dimension; and  $RI_j$  is length of the region of interest in the  $j$ th dimension or the universe of discourse  $X$ .  $x \in X$  is the input elements.  $m(x)$  the membership degrees of  $x$ .  $\delta \in [0, 1]$  is the *level of completeness*.  $N_{m(x) \geq \delta}$  is the number of element  $x$  that has membership degree larger than  $\delta$ .  $N_{m(x) < \delta}$  is the number of element  $x$  that has membership degree smaller than  $\delta$ .  $N_{x \in X}$  is the total number of element  $x$  in the universe of discourse  $X$ .

It is an N-P hard problem in constructing a fuzzy rule-based system that is to preserve the performance accuracy and the comprehensibility in term of knowledge representation. There are many ways to optimize fuzzy rules. One of the popular techniques is to use evolutionary computation such as genetic algorithms (GAs). In this study, we apply a GA to perform an optimization process of fuzzy rules by searching both for good accuracy and the comprehensibility based on the quantity measures discussed above.

#### 4.4 Fitness Functions Implemented for the Genetic Algorithm

GAs have been widely used for helping in the generation of if-then rule bases of FESs. When a FES is constructed, accuracy and comprehensibility should be concerned during the optimization process using the GAs. A fitness function is used to guide the evolutionary process to a satisfactory goal. The fitness function used is based on the accuracy performance of resulting rules, the comprehensibility of the rule set, and the completeness of the fuzzy rule structure. The fitness function can be determined from the following equations:

$$F_N = (W_A)(A_C + S_E + S_P) - (W_B) [(L)(N_A) + (M)(N_L) + (LS) + RS + I_D] \quad (4.22)$$

where  $F_N$  represents an overall fitness function;  $A_C$ ,  $S_E$ , and  $S_P$  are the accuracy, the sensitivity, and the specificity, respectively;  $L$  and  $N_A$  are the numbers of rules and antecedents per rule, respectively;  $M$  and  $N_L$  are the numbers of dimensions and linguistic terms per dimension;  $LS$  is linguistic similarity;  $RS$  is the rules similarity or the inconsistency; and  $I_D$  is the degree of incompleteness; and  $W_A$  is the weight for the reinforcement part and  $W_B$  is the weight for penalty part. Usually  $W_A$  is selected to be larger than  $W_B$ , since the accuracy of the system is paid more attention.

## CHAPTER V

### BENCHMARK SIMULATION RESULTS

To demonstrate the performance of the HIS, computer simulations were used in our study. Simulations and analysis of the HIS are performed using two well-known benchmark data sets: Iris data [Fisher36] and Wisconsin breast cancer data [Wolberg90]. The Iris data is used because it is a simple and widely used benchmark data set. Wisconsin breast cancer database (WBCD) [Wolberg90] is used as an example for application in medical diagnosis. The WBCD is a real application that has been exploited by many researchers [Peña99], [Setiono96], [Taha99], [Setiono00].

#### 5.1 Iris Data Set

The Fisher's Iris flower data set consists of 150 patterns with four features: sepal length, sepal width, petal length, and petal width. These four features describe the shape and size of the Iris flowers. Each pattern in the data set falls into one of three classes: Setosa, Versicolour and Virginica, with a total of 50 patterns per class. For the purpose of this experiment, we will call them Class 1, Class 2, and Class 3, respectively. Class 1 is linearly separable from the other two. However, Classes 2 and 3 are not linearly separable from each other.

Figure 5.1 shows the scatter plot of the Iris data for sepal width and length features. It is worth noting from the plot that Class 1 can be easily separated from Classes 2 and 3. However, Class 2 and Class 3 seem very difficult to separate since there is an overlap between them. Moreover, in Figure 5.2, the petal width and length features

are plotted, showing that Class 1 is very well separated from Classes 2 and 3. However, Class 2 and Class 3 remain overlapped [Fisher36].

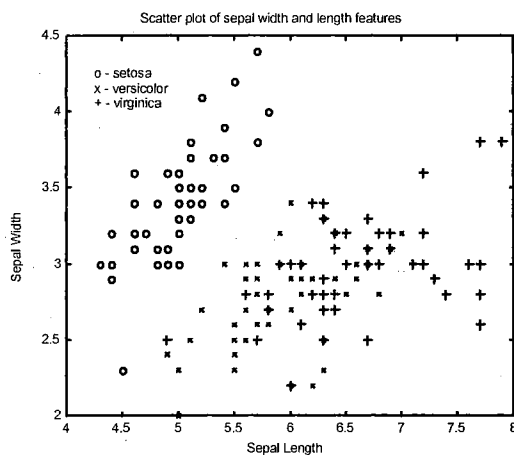


Figure 5.1: Scatter Plot of Sepal Width and Length Features of the Fisher's Iris Data

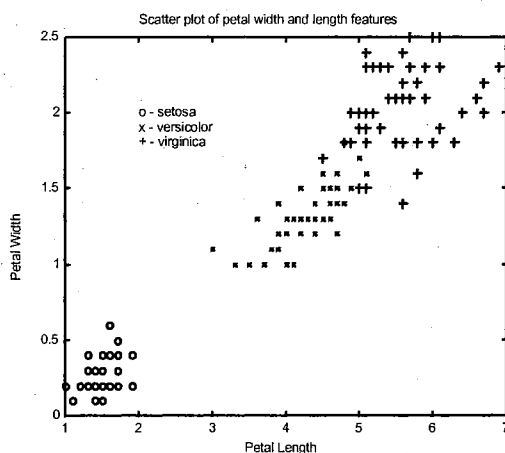


Figure 5.2: Scatter Plot of Petal Width and Length Features of the Fisher's Iris Data

### 5.1.1 Simulation Results for the Iris Data Set

We used 75 patterns with the first 25 patterns from each class for training. The remaining 75 patterns were used for testing. Setting the threshold  $\epsilon = 0$  and the standard deviation  $\sigma_0 = 1$ , the ILFN network one-pass incrementally learned the training set in



which 9 hidden neurons are constructed. The resulting numerical parameters of the ILFN network are shown in Tables 5.1 and 5.2. With the numerical values in Tables 5.1 and 5.2, ILFN classifier achieved 96% for training set and 98.67% for test set.

TABLE 5.1:  
 $W_P$  and  $W_T$  from the ILFN Classifier for the Iris Data

$W_P$				$W_T$
5.028	3.48	1.46	0.248	1
6.1059	2.8118	4.4529	1.3882	2
7.1	3.125	6.1583	2.1583	3
5.56	2.66	4.92	2	3
5.26	2.5	3.64	1.16	2
6.5167	2.9167	5.4333	1.9833	3
6	2.2	5	1.5	3
6.3	2.7	4.9	1.8	3
6.7333	3.0333	4.6333	1.4	2

TABLE 5.2:  
The Standard Deviation,  $S$ , and the Number of Patterns, **count**,  
from the ILFN Classifier for the Iris Data

$S$				<b>count</b>
1.3497	1.2501	0.68582	0.39022	25
0.91305	1.078	0.98196	0.70182	17
1.13	0.76985	1.1016	0.63668	12
0.65772	0.48898	0.55937	0.61237	5
0.69121	0.64083	0.58204	0.50869	5
0.51473	0.63733	0.55909	0.49749	6
1	1	1	1	1
1	1	1	1	1
0.59815	0.59255	0.61328	0.58737	3

The rule extraction algorithm, “ilfn2rule” discussed in Section 3.3, was used to map the numerical parameters of the ILFN network from Table 5.1 into fuzzy linguistic form shown in Table 5.4. Table 5.3 indicates the extracted linguistic variables and their

parameters. Table 5.4 shows the original fuzzy rule set that was the result from the rule extraction “ilfn2rule.”

TABLE 5.3:  
Resulted Linguistic Labels and Their Parameters for the Iris Data

Features	Linguistic Labels and Parameters		
F1: Sepal Length	1: Short	2: Long	
	(Gaussian, 5.28, 0.6827)	(Gaussian, 7.1, 0.6827)	
F2: Sepal Width	1: Narrow	2: Wide	
	(Gaussian, 2.2, 0.4218)	(Gaussian, 3.48, 0.4218)	
F3: Petal Length	1: Short	2: Medium	3: Long
	(Gaussian, 1.46, 0.7741)	(Gaussian, 3.8092, 0.7741)	(Gaussian, 6.1583, 0.7741)
F4: Petal Width	1: Narrow	2: Medium	3: Wide
	(Gaussian, 0.248, 0.3147)	(Gaussian, 1.2032, 0.3147)	(Gaussian, 2.1583, 0.3147)

TABLE 5.4:  
Original Fuzzy If-Then Rules for the Iris Data

Antecedent				Consequent	
F1	F2	F3	F4	Class	CF
1	2	1	1	1	1
2	1	2	2	2	0.68
2	2	3	3	3	0.48
1	1	2	3	3	0.2
1	1	2	2	2	0.2
2	2	3	3	3	0.24
1	1	3	2	3	0.04
2	1	2	3	3	0.04
2	2	2	2	2	0.12

A fuzzy expert system with the original rule classified the training pattern with 96% correct classification. The generalization for the test set was 96% correct classification. Although the originally extracted rule is functional, it may be further simplified by reducing the number of discriminatory features. This simplification greatly enhances human understandability of the rule without sacrificing performance. In other words, it is possible to use fewer features which have high discriminatory power. In this

simulation, the genetic algorithm was used to select only important features that contribute a high discriminatory power. The genetic parameters are chosen as follows: population size = 100, the probability for mutation,  $p_m = 0.8$ , and the probability for crossover,  $p_c = 0.01$ . After running for 60 generations, the genetic algorithm yielded results shown in Table 5.5.

TABLE 5.5:  
Final Fuzzy If-Then Rules for the Iris Data

Antecedents				Consequent	
F1	F2	F3	F4	Class	CF
0	0	0	1	1	1
2	0	2	2	2	0.68
2	0	3	0	3	0.48
0	0	0	3	3	0.2
0	0	2	2	2	0.2

From Table 5.5, the linguistic rules can be interpreted in a natural language form as follow:

- Rule 1: If petal width is *narrow*  
Then class is Iris Setosa with CF = 1;
- Rule 2: If sepal length is *long* and  
petal length is *medium* and  
petal width is *medium*  
Then class is Iris Versicolor with CF = 0.68;
- Rule 3: If sepal length *long* and  
petal length is *long*  
Then class is Iris Virginica with CF = 0.48;
- Rule 4: If petal width is *wide*

Then class is Iris Virginica with  $CF = 0.2$ ;

Rule 5: If petal length is *medium* and  
petal width is *medium*

Then class is Iris Versicolor with  $CF = 0.2$ .

Using 75 patterns for training and 75 patterns for testing, the fuzzy expert system with this rule set achieved 98.67% and 96% correct classification for training and testing, respectively.

We combine the trained ILFN and the fuzzy expert system resulting in a hybrid intelligent system. The weights between the decision output from the ILFN and the FES are  $\alpha = 0.42857$  and  $\beta = 0.57143$  (where  $\alpha$  and  $\beta$  were determined by using GA). The resulting system achieve 100% for classification rate in training set and 100% for classification rate in the test set.

## 5.2 Wisconsin Breast Cancer Data (WBCD)

The WBCD contains a collection of 699 patterns each described by 9 features. Each feature is a real number in the interval 1 to 10 based on a fine needle aspirate taken directly from human breasts: clump thickness, size uniformity, shape uniformity, marginal adhesion, cell size, bare nuclei, bland chromatin, normal nucleoli and mitosis. The larger the values of these attributes yield the greater the likelihood of malignancy. There are 458 patterns for benign (labeling as "2" in the data base) and 241 patterns for malignant (labeling as "4"). There are 16 patterns with incomplete feature descriptions marked as "?" [Wolberg90], [Blake98]. We replaced the missing values with "0."

### 5.2.1 Simulation Results for the WBCD

Ten simulations were performed to evaluate the proposed method. In every simulation, the ILFN learning parameters were set to defaults as follows: the threshold,  $\varepsilon = 0$  and the standard deviation,  $\sigma_0 = 0.5$ . The numerical weights of the ILFN network were extracted to fuzzy initial linguistic rules. In order to optimize the linguistic rules, the GA with the integer chromosome representation was used by setting its learning parameters heuristically as follows: population size = 100, the number of generations = 100, the mutation probability,  $p_m = 0.8$ , and the crossover probability,  $p_c = 0.01$ . The weights in the fitness evaluation are set as follows:  $W_{PC} = 50$ ,  $W_F = 5$ , and  $W_{NL} = 1$ . The number of linguistic labels was constrained to within 3 for each dimension. The GA with a real chromosome representation also was used to find the weighting parameters,  $\alpha$  and  $\beta$ . The parameters for the GA were as follows: population size = 60, the number of generations = 20, the mutation probability,  $p_m = 0.8$ , and the crossover probability,  $p_c = 0.01$ . The results from the ten simulations are shown in Table 5.6.

From Table 5.6, the ILFN achieved an average correct classification of 96.17% on training set and 97.37% on test set. The fuzzy rules extracted from the trained ILFN achieved an average correct classification of 97.43% on training set and 96.72% on test set. It is worth noting that the fuzzy rules extracted from the trained ILFN achieved a higher correct classification rate for the training set. However, the fuzzy rules achieved lower percentage of correctly classified patterns from the test set. When we combined the ILFN and fuzzy rules extracted to construct a HIS, the results show that the HIS achieved a better performance than both the ILFN and the extracted fuzzy rules alone. The

proposed HIS had an average of 97.61% and 97.48% correct classification on the training set and the test set, respectively.

TABLE 5.6:  
Simulation Results for the WBCD

Run no.	Methods	Structure complexity		Number of patterns		% Correctly classified patterns		
		#Nodes and/or #Rules	# conditions*	# Training	# Test	Training set	% Test set	% Overall patterns
1	Numerical (ILFN)	3 nodes	9 F/N	100	599	98%	96.83%	97.00%
	Fuzzy Rules	3 rules	2.3 F/R	100	599	98%	96.49%	96.71%
	Hybrid ILFN and Fuzzy Rules	3 nodes & 3 rules	9 F/N & 2.3 F/R	100	599	98%	96.83%	97.00%
2	Numerical (ILFN)	3 nodes	9 F/N	100	599	98%	96.83%	97.00%
	Fuzzy Rules	3 rules	4 F/R	100	599	98%	94.74%	96.25%
	Hybrid ILFN and Fuzzy Rules	3 nodes & 3 rules	9 F/N & 4 F/R	100	599	98%	97.14%	97.57%
3	Numerical (ILFN)	3 nodes	9 F/N	100	342	98%	97.95%	97.23%
	Fuzzy Rules	3 rules	2.7 F/R	341	342	97.95%	96.49%	96.71%
	Hybrid ILFN and Fuzzy Rules	3 nodes & 3 rules	9 F/N & 2.7 F/R	341	342	98%	98.25%	98.13%
4	Numerical (ILFN)	4 nodes	9 F/N	120	358	95.83%	97.49%	96.57%
	Fuzzy Rules	4 rules	3.75 F/R	341	358	97.36%	96.65%	97.00%
	Hybrid ILFN and Fuzzy Rules	4 nodes & 4 rules	9 F/N & 3.75 F/R	341	358	97.07%	97.50%	97.43%
5	Numerical (ILFN)	4 nodes	9 F/N	120	342	95.83%	98.25%	96.93%
	Fuzzy Rules	3 rules	2.67 F/R	341	342	96.77%	96.78%	96.79%
	Hybrid ILFN and Fuzzy Rules	4 nodes & 3 rules	9 F/N, 2.67 F/R	341	342	96.48%	98.25%	97.36%
6	Numerical (ILFN)	5 nodes	9 F/N	150	342	94.67%	97.19%	96.63%
	Fuzzy Rules	5 rules	2.2 F/R	341	342	97.07%	97.37%	97.22%
	Hybrid ILFN and Fuzzy Rules	5 nodes, 5 rules	9 F/N & 2.2 F/R	341	342	97.07%	97.08%	97.07%
7	Numerical (ILFN)	5 nodes	9 F/N	150	342	94.67%	97.19%	96.63%
	Fuzzy Rules	4 rules	2.5 F/R	683	683	97.07%	97.07%	97.07%
	Hybrid ILFN and Fuzzy Rules	5 nodes & 4 rules	9 F/N & 2.5 F/R	683	683	97.22%	97.22%	97.22%
8	Numerical (ILFN)	3 nodes	9 F/N	100	342	98%	97.95%	97.23%
	Fuzzy Rules	2 rules	3 F/R	683	683	97.23%	97.23%	97.23%
	Hybrid ILFN and Fuzzy Rules	3 nodes & 2 rules	9 F/N & 3 F/R	683	683	97.57%	97.57%	97.57%
9	Numerical (ILFN)	5 nodes	9 F/N	150	549	94.67%	97.19%	96.42%
	Fuzzy Rules	4 rules	2.5 F/R	699	699	97.57%	97.57%	97.57%
	Hybrid ILFN and Fuzzy Rules	5 nodes & 4 rules	9 F/N & 2.5 F/R	699	699	97.57%	97.57%	97.57%
10	Numerical (ILFN)	3 nodes	9 F/N	100	599	98%	96.83%	97.00%
	Fuzzy Rules	3 rules	2.33 F/R	699	699	96.85%	96.85%	96.85%
	Hybrid ILFN and Fuzzy Rules	3 nodes & 3 rules	9 F/N & 2.33 F/R	699	699	97.42%	97.42%	97.42%
Average	Numerical (ILFN)	3.8 nodes	9 F/N			96.17%	97.37%	96.77%
	Fuzzy Rules	3.4 rules	2.77 F/R			97.43%	96.72%	97.08%
	Hybrid ILFN and Fuzzy Rules	3.8 nodes & 3.4 rules	9 F/N & 2.77 F/R			97.61%	97.48%	97.55%

\* F/N = # features per node and F/R = # features per rule.

Due to the space limitation, we show the details of numerical weights of the ILFN and the extracted linguistic rules from one example (highlighted) based on the best classification performance of the HIS, i.e., from run number 3 in Table 5.6. Based on run

number 3, the details on ILFN and its linguistic rules extracted are shown in Tables 5.7, 5.8, and 5.9.

TABLE 5.7:  
ILFN Parameters for the WBCD

WP									WT
2.7818	1.3455	1.4182	1.2727	2.0545	1.5273	2.7818	1.1818	1.0909	2
7.3462	6.6538	6.6154	5.1923	6.7692	7.5	5.5385	6.9615	3.4615	4
6.6316	3.7895	4.3684	2.6842	3.8947	4	3.8947	4.4211	2.0526	4
Standard Deviation									count
8.0293	3.4577	4.1716	2.6234	2.3358	5.2169	7.9414	2.2247	1.6642	55
8.9208	9.4657	7.9145	12.538	9.2651	9.8717	7.2446	10.184	13.132	26
8.9898	4.0137	4.2122	4.8625	5.2584	7.6044	3.4663	8.9787	7.9811	19

TABLE 5.8:  
Resulted Linguistic Labels and Their Parameters for the WBCD

Features	Linguistic Labels and Parameters		
F <sub>1</sub> = Clump Thickness	1: <i>low</i> <sub>1</sub>	2: <i>high</i> <sub>1</sub>	
	(Gaussian: 2.7818, 1.504)*	(Gaussian: 7.3462, 1.504)	
F <sub>2</sub> = Size Uniformity	1: <i>low</i> <sub>2</sub>	2: <i>high</i> <sub>2</sub>	
	(Gaussian: 1.3455, 1.7491)	(Gaussian: 6.6538, 1.7491)	
F <sub>3</sub> = Shape Uniformity	1: <i>low</i> <sub>3</sub>	2: <i>medium</i> <sub>3</sub>	3: <i>high</i> <sub>3</sub>
	(Gaussian: 1.4182, 0.85625)	(Gaussian: 4.0168, 0.85625)	(Gaussian: 6.6154, 0.85625)
F <sub>4</sub> = Marginal Adhesion	1: <i>low</i> <sub>4</sub>	2: <i>high</i> <sub>4</sub>	
	(Gaussian: 1.2727, 1.2915)	(Gaussian: 5.1923, 1.2915)	
F <sub>5</sub> = Cell Size	1: <i>low</i> <sub>5</sub>	2: <i>medium</i> <sub>5</sub>	3: <i>high</i> <sub>5</sub>
	(Gaussian: 2.0545, 0.77676)	(Gaussian: 4.4119, 0.77676)	(Gaussian: 6.7692, 0.77676)
F <sub>6</sub> = Bare Nuclei	1: <i>low</i> <sub>6</sub>	2: <i>high</i> <sub>6</sub>	
	(Gaussian: 1.5273, 1.968)	(Gaussian: 7.5, 1.968)	
F <sub>7</sub> = Bland Chromatin	1: <i>low</i> <sub>7</sub>	2: <i>medium</i> <sub>7</sub>	3: <i>high</i> <sub>7</sub>
	(Gaussian: 2.7818, 0.45416)	(Gaussian: 4.1601, 0.45416)	(Gaussian: 5.5385, 0.45416)
F <sub>8</sub> = Normal Nucleoli	1: <i>low</i> <sub>8</sub>	2: <i>medium</i> <sub>8</sub>	3: <i>high</i> <sub>8</sub>
	(Gaussian: 1.1818, 0.95222)	(Gaussian: 4.0717, 0.95222)	(Gaussian: 6.9615, 0.95222)
F <sub>9</sub> = Mitosis	1: <i>low</i> <sub>9</sub>	2: <i>high</i> <sub>9</sub>	
	(Gaussian: 1.0909, 0.78113)	(Gaussian: 3.4615, 0.78113)	

\* Since Gaussian membership functions are used, the parameters of the linguistic labels are written as (Gaussian: mean, standard deviation).

From run number 3, we used 100 patterns for training the ILFN, 341 patterns for training the FES and HIS, and used 342 patterns for testing in all three systems. The ILFN constructed 3 hidden nodes with the parameters shown in Table 5.7. The ILFN

network achieved 98% and 97.95% correct classification for the training set and the test set, respectively.

From Table 5.7, the knowledge embedded in the trained ILFN is in numerical form. Linguistic rules are preferably extracted from the trained ILFN for a reasoning purpose. The fuzzy linguistic rules are mapped from the ILFN parameters and the GA is used to select only discriminatory features. This will be resulted in a more compact rule set.

TABLE 5.9:  
Fuzzy Expert Rules for the WBCD

Antecedent									Consequent	
F <sub>1</sub>	F <sub>2</sub>	F <sub>3</sub>	F <sub>4</sub>	F <sub>5</sub>	F <sub>6</sub>	F <sub>7</sub>	F <sub>8</sub>	F <sub>9</sub>	Class	CF
1	1	0	1	1	0	0	1	1	2	1
2	0	3	0	0	0	0	0	2	4	0.57778
2	0	0	0	0	0	2	0	1	4	0.42222

	F <sub>1</sub>	F <sub>2</sub>	F <sub>3</sub>	F <sub>4</sub>	F <sub>5</sub>	F <sub>6</sub>	F <sub>7</sub>	F <sub>8</sub>	F <sub>9</sub>	Class	CF
<b>Rule 1</b>										2	1
<b>Rule 2</b>										4	0.57778
<b>Rule 3</b>										4	0.42222

After running for 100 generations, the resulted fuzzy linguistic labels are shown in Table 5.8. The fuzzy linguistic rules are shown in Table 5.9. Using the linguistic knowledge from Tables 5.8 and 5.9 as the rule set for a fuzzy expert system, the final fuzzy linguistic rules achieved 97.95% and 96.49% correct classification for training and testing data, respectively. The hybrid intelligent system combining the decisions from both ILFN and FES achieved 98% correct classification rate for training set and 98.25% correct classification rate for the test set. The HIS achieved 98.13% in all 683 patterns of the WBCD. Fuzzy expert rules in natural language for the WBCD can be interpreted as



Rule1: If Clump Thickness is *low*<sub>1</sub> and Size Uniformity is *low*<sub>2</sub> and Marginal Adhesion is *low*<sub>4</sub> and Cell Size is *low*<sub>5</sub> and Normal Nucleoli is *low*<sub>8</sub> and Mitosis is *low*<sub>9</sub>, Then Malignant, with confidence = 1;

Rule2: If Clump Thickness is *high*<sub>1</sub> and Shape Uniformity is *high*<sub>3</sub> and Mitosis is *high*<sub>9</sub>, Then Benign, with confidence = 0.58;

Rule3: If Clump Thickness is *high*<sub>1</sub> and Bland Chromatin is *medium*<sub>7</sub> and Mitosis is *low*<sub>9</sub>, Then Benign, with confidence = 0.42;

### 5.2.2 Comparison Results for the WBCD

Several groups of researchers have studied and developed knowledge-based system for the WBCD. Peña-Reyes and Sipper used a fuzzy if-then system as a classifier. They developed a fuzzy-GA algorithm to extract rules from the WBCD. Fuzzy-GA algorithm uses the genetic algorithm (GA) to search for two parameters, P and d, of their fuzzy rules [Peña99]. The number of rules has to be predetermined in an *ad hoc* manner. In [Setiono96], Setiono developed a rule extraction called NeuroRule. NeuroRule uses a pruning procedure after the training phase to decrease the number of the network connections. The pruning process runs until network performance drops to 95% correct classification rate. In [Setiono96], 100-MLP networks were used in the training phase. The network with the best performance out of 100 pruned networks was used in rule-extraction phase. The NeuroRule extracts rules by clustering the hidden nodes activation values. Then, the input combinations are checked if any input makes the hidden nodes and output node active. An improvement of NeuroRule in the WBCD was studied by the same author in [Setiono00] by data pre-processing before the training step. Another

group was Taha and Ghosh [Taha99]. In [Taha99], three rule extraction algorithms were developed: BIO-RE, Partial-RE, and Ful-RE. BIO-RE is a black box rule extraction technique which does not require information regarding the internal network structure to generate rules. Partial-RE searches for a set of incoming connections that will cause a unit to be active. Full-RE decomposes the rule extraction process into two steps: rules between hidden and output units and rules between input units and hidden units. This is similar to NeuroRule [Setiono00] but the difference is that Full-RE employs linear programming and an input discretization method to find a combination of the input values that will cause a hidden unit to be activated. Comparison results on the WBCD are shown in Table 5.10, which shows the comparison among several rule-based systems from [Peña99], [Setiono96], [Taha99], [Setiono00].

TABLE 5.10:  
Comparison Results for the WBCD Among Well-Known Methods

Methods	Representaion Type	Rule complexity		Number of patterns		Performance Evaluation		
		# Rules	# F/R*	# Training	# Test	% Training corr.	% Test corr.	% Overall corr.
NeuroRule [Setiono96]	Boolean Rules	1 + default	2	350	349	96.86%	93.98%	95.42%
NeuroRule [Setiono96]	Boolean Rules	2 + default	4	350	349	97.71%	96.56%	97.14%
NeuroRule [Setiono00]	Boolean Rules	1 + default	4	341	342	97.07%	97.66%	97.36%
NeuroRule [Setiono00]	Boolean Rules	3 + default	3.7	341	342	97.95%	98.25%	98.10%
NeuroRule [Setiono00]	Boolean Rules	4 + default	1	341	342	97.07%	97.66%	97.36%
NeuroRule [Setiono00]	Boolean Rules	5 + default	4.2	341	342	98.53%	97.95%	98.24%
NeuroRule [Setiono00]	Boolean Rules	6 + default	1.7	341	342	97.95%	98.25%	98.10%
Fuzzy-GA [Peña99]	Fuzzy Rules	1 + default	4	341	342			97.07%
Fuzzy-GA [Peña99]	Fuzzy Rules	2 + default	3	341	342			97.36%
Fuzzy-GA [Peña99]	Fuzzy Rules	3 + default	4.7	341	342			97.80%
Fuzzy-GA [Peña99]	Fuzzy Rules	4 + default	4.8	341	342			97.80%
Fuzzy-GA [Peña99]	Fuzzy Rules	5 + default	3.4	341	342			97.51%
BIO-RE [Taha99]	Boolean Rules	11 + default	2.7	341	342	97.07%	96.20%	96.63%
Partial-RE [Taha99]	Boolean Rules	9 + default	2.67	341	342	97.07%	95.91%	96.49%
Full-RE [Taha99]	Boolean Rules	5	1.8	341	342	96.77%	95.61%	96.19%
This study	Numerical (ILFN)	(3 nodes)	(9 F/N)	100	342	98%	97.95%	97.23%
	Fuzzy Rules	3	2.7	341	342	97.95%	96.49%	96.71%
	Hybrid ILFN and Fuzzy Rules	(3 nodes & 3 rules)	(9 features & 2.7 F/R)	341	342	98%	98.25%	98.13%

\* F/R = # features per rule and F/N = # features per node.

From Table 5.10, the best performance was from NeuroRule [Setiono00] with 5 rules plus a default rule extracted from one of the 100 pruned networks with 2 hidden

units and 9 connections. The accuracy rate was 98.24% in 683 patterns. The rule set extracted in [Setiono00] is as follows:

If  $F_2 \leq 4$  and  $F_6 \leq 2$  and  $F_8 \leq 2$ , then benign,

Else if  $F_2 \leq 4$  and  $F_6 \leq 2$  and  $F_8 \leq 8$  and  $F_1 \leq 6$ , then benign,

Else if  $F_1 \leq 5$  and  $F_4 \leq 4$  and  $F_6 \leq 5$  and  $F_8 \leq 2$ , then benign,

Else if  $F_1 \leq 6$  and  $F_2 \leq 4$  and  $F_6 \leq 6$  and  $F_8 \leq 8$ , then benign,

Else if  $F_2 \leq 4$  and  $F_4 \leq 5$  and  $F_6 \leq 5$  and  $3 \leq F_6 \leq 5$  and  $F_8 \leq 8$ , then benign,

Else malignant

NeuroRule does not produce any rule for malignancy. It needs a default rule for malignancy. Fuzzy-GA [Peña99] extracted rules based on the predetermined number of rules in the range of 1 to 5. A total of 120 evolutionary runs were performed. The highest performance system was 97.80% correct classification rate using 3 fuzzy if-then rules with 4.7 conditions per rule, and a default rule. In [Taha99], using Bio-RE algorithm, the best performance system was 96.96% using 11 Boolean rules with 2.7 conditions per rule. Using Full-RE algorithm, the best performance was 96.19% with 5 rules and 1.8 conditions per rule (no default rule). NeuroRule [Setiono96], [Setiono00], fuzzy-GA [12 Peña99], Bio-RE [Taha99], and Partial-RE [Taha99] have a default rule that seems to imply they lack of completeness. Default rules do not provide a symbolic interpretation of the decision other than that “because none of the above occurred” [Taha99].

Based on ten runs, our proposed HIS achieved 98.13% correct classification for all 683 patterns. The HIS used ILFN with 3 hidden nodes and FES with 3 fuzzy if-then rules and 2.7 conditions per rules. An advantage of the proposed HIS is that it incorporates an incremental learning characteristic in the system. Since data can be made

available on a daily basis, using the proposed HIS, the novel data can be added into the system quickly without spending too much time on retraining all the old information.

### **5.3 Additional Medical Diagnosis Data Sets**

Three real medical data domains which are provided by the Institute of Oncology, University Medical Center, Ljubljana, Yugoslavia were used for evaluation of the proposed method. The three data domains include Breast Cancer Data, Lymphograph Domain, and Primary Tumor Domain. The three data sets were archived and publicly accessible at the UCI repository of machine learning databases and domain theories, <http://www.ics.uci.edu/~mlearn/MLRepository.html> [Murphy95].

#### **5.3.1 Breast Cancer data**

This data set has 286 instances: 201 instances of nonrecurrence and 85 instances of recurrence. The instances are described by nine attributes, some of which are linear and some are nominal, plus one class attribute, as shown in Table 5.11. This data set has the recurrence versus non-recurrence of breast cancer in patients and provides information for evaluating the prognosis of breast-cancer recurrence. The data has been used as a benchmark test for machine learning studies. This data set presents a challenging problem because of the fact that the best test accuracy reported in the literature on this domain is less than 80%. It reflects a high degree of uncertainty involved or insufficient discriminant information provided.

Since the data is not in a format that can be applied to the developed systems, it has been rearranged to the format that can be processed by the algorithms studied in this dissertation as follows.

For the class attribute, the original parameters are *no-recurrence-events*, *recurrence-events* are replaced by numbers 1 and 2, respectively. For age attribute, numbers 15, 25, 35, 45, 55, 65, 75, 85, and 95 replace the ranges of numbers *10-19*, *20-29*, *30-39*, *40-49*, *50-59*, *60-69*, *70-79*, *80-89*, and *90-99*, respectively. Numbers 10, 20, and 30 replace the parameters *lt40*, *ge40*, and *premeno*, respectively, in the attribute *menopause*. For tumor-size attribute, the ranges of number *0-4*, *5-9*, *10-14*, *15-19*, *20-24*, *25-29*, *30-34*, *35-39*, *40-44*, *45-49*, *50-54*, and *55-59* are replaced by numbers 2, 7, 12, 17, 22, 27, 32, 37, 42, 47, 52, and 57, respectively. The ranges of number *0-2*, *3-5*, *6-8*, *9-11*, *12-14*, *15-17*, *18-20*, *21-23*, *24-26*, *27-29*, *30-32*, *33-35*, and *36-39* in the *inv-nodes* attribute are substituted by numbers 1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34, and 37, respectively. The parameters of the node-caps *yes* and *no* are substituted by numbers 10 and 20, respectively. The parameters of the *deg-malig* attribute are kept the same. For the breast attribute, the parameters *left* and *right* are substituted by numbers 10 and 20, respectively. Numbers 10, 20, 30, 40, and 50 substitute the parameters *left-up*, *left-low*, *right-up*, *right-low*, and *central*, respectively, in the *breast-quad* attribute. For the *irradiat*, *yes* and *no* are replaced by numbers 10 and 20, respectively.

### 5.3.1.1 Simulation Results for the Breast Cancer Data

Table 5.12 shows the results on breast cancer data from other researchers in the literature and from this study. In this data set, Michalski *et al.* used their developed system called AQ15, and achieved 66-72% correct classification performance for the test data. Clark *et al.* achieved 65-72% correct classification performance using a simple

Bayes network. Assistant86 developed by Cestnik *et al.* achieved the best performance of 78% of correct classification.

In this study, 200 patterns were randomly selected for training and the remaining 86 patterns for testing to ILFN, FES, and HIS. As shown in Table 5.12, based on an average of five runs, the accuracy performance on the ILFN was 82.5% correct classification based on 200 pattern from training sets and 63.95% correct classification based on 86 patterns from the test set. The accuracy of the FES was 75.5% correct classification based on 200 training patterns, 73.26% correct classification based on 86 test patterns, and 74.83% classification overall. The HIS achieved 88% correct classification on the training patterns, 64.79% correct classification on the test patterns, and 80.41% correct classification overall. It is found that in this data set the HIS achieved the highest accuracy results compared to other methods.

Other than the accuracy performance, the sensibility and specificity are also used in a diagnostic test of binary class problems such as the breast cancer data. A high percentage of the sensitivity or the specificity implies that the systems are more accurate and reliable. The sensibility and specificity are shown in Table 5.12. The sensitivities of the ILFN on training set, test set, and overall were 80.99%, 74.57%, and 79.10%, respectively. The sensitivities of the FES on training set, test set, and overall were 98.59%, 94.92%, and 97.51%, respectively. The sensitivities of the HIS on training set, test set, and overall were 96.48%, 81.36%, and 92.04%, respectively. The specificities of the ILFN on training set, test set, and overall were 86.21%, 40.74%, and 71.76%, respectively. The specificities of the FES on training set, test set, and overall were

18.96%, 25.92%, and 21.17%, respectively. The specificities of the HIS on training set, test set, and overall were 67.24%, 22.22%, and 52.94%, respectively.

TABLE 5.11:  
Breast Cancer Data

Attribute	Attribute Information
1	Class: no-recurrence-events, recurrence-events
2	Age: 10-19, 20-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89, 90-99.
3	Menopause: lt40, ge40, premeno.
4	Tumor-size: 0-4, 5-9, 10-14, 15-19, 20-24, 25-29, 30-34, 35-39, 40-44, 45-49, 50-54, 55-59.
5	Inv-nodes: 0-2, 3-5, 6-8, 9-11, 12-14, 15-17, 18-20, 21-23, 24-26, 27-29, 30-32, 33-35, 36-39.
6	Node-caps: yes, no.
7	Deg-malig: 1, 2, 3.
8	Breast: left, right.
9	Breast-quad: left-up, left-low, right-up, right-low, central.
10	Irradiate: yes, no.

TABLE 5.12:  
Simulation Results for the Breast Cancer Data

Methods	Reference	Accuracy (%)			Sensibility (%)			Specificity (%)		
		Train	Test	Overall	Train	Test	Overall	Train	Test	Overall
AQ15	Michalski	-	66-72	-	-	-	-	-	-	-
Simple Bayes	Clark	-	65-72	-	-	-	-	-	-	-
Weighted Network	Tan	-	60-73.5	-	-	-	-	-	-	-
Assistance86	Cestnik	-	78	-	-	-	-	-	-	-
CLILP2	Liu	-	76	-	-	-	-	-	-	-
ILFN	Meesad	82.5	63.95	76.92	80.99	74.57	79.1	86.21	40.74	71.76
FES	Meesad	75.5	73.26	74.83	98.59	94.92	97.51	18.96	25.92	21.17
HIS	Meesad	88	64.79	80.41	96.48	81.36	92.04	67.24	22.22	52.94

The fuzzy if-then rules of the FES are as follows:

If (*age is Old*) and (*menopause is Premeno*) and (*deg-malig is High*), Then Class is *no-recurrence-event* with confidence 0.99.

IF (*age is Young*) and (*menopause is LT40*) and (*inv-nodes is Small*) and (*node-caps is Yes*) and (*breast-quad is left-up*) and (*irradiate is Yes*), Then Class is *no-recurrence-events* with confidence 1.

If (*age is Old*) and (*menopause is Premeno*) and (*inv-nodes is Big*) and (*deg-malig is High*) and (*irradiate is Yes*), Then Class is *recurrence-events* with confidence 1.

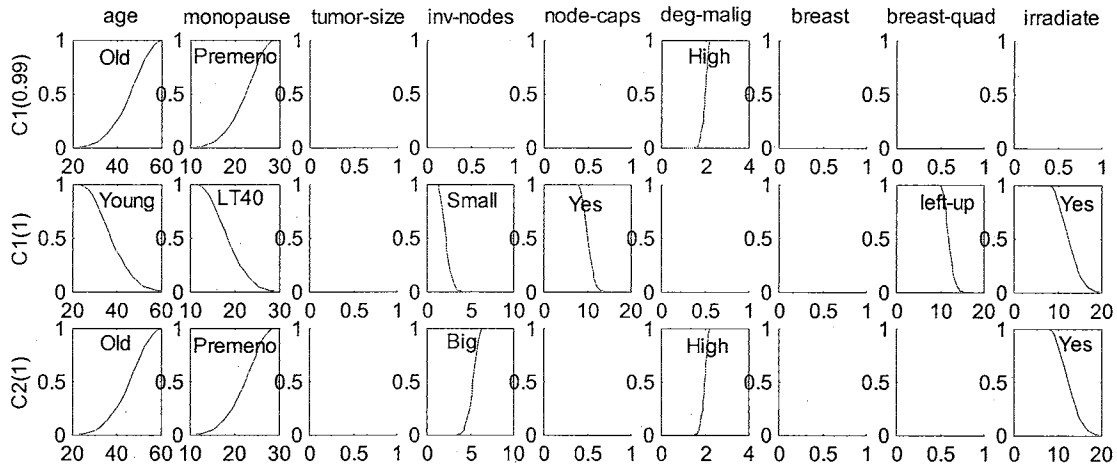


Figure 5.3: Linguistic Rules for the Breast Cancer Data

TABLE 5.13:  
The Compressibility of Fuzzy Rules for the Breast Cancer Data

Data	Comprehensibility					$I_D$
	$L$	$N_A$	$N_L$	$LS$	$RS$	
Breast Cancer	3	4.7	2.4	0.084	0.53	0.02

The comprehensibility of the fuzzy rules can be quantified as shown in Table 5.13. It is found that the knowledge base obtained from the proposed method is compact and highly comprehensible. The number of rules ( $L$ ) is 3, which is very small. The number of antecedents per rule ( $N_A$ ) is 4.7. Please note that this is a nine-attribute problem.  $N_A = 4.7$  is considered that a system is very comprehensible. In addition, the number of labels per dimension ( $N_L$ ) is incredibly small. There are only 2.4 labels per dimension in the fuzzy knowledge base. The degree of linguistic similarity ( $LS$ ) of the fuzzy rules is 0.084. This implies that the fuzzy variables are easy to be discerned from



each other. The degree of fuzzy rule similarity ( $RS$ ) is 0.53. This shows that the fuzzy rule is not conflicting each other greatly. The fuzzy rule structure has an incompleteness degree of 0.02. This implies that the rule structure of the system is nearly complete.

### 5.3.2 Lymphography Domain

The aim is to determine the results of the lymphographic investigation. This data is described by four subsets of eight features. There are 148 patterns in the data set. The training data comprised of 103 learning patterns and 45 test patterns. The set of features for this domain was complete i.e., always sufficient to differentiate between different cases. Actual testing of physicians was not performed and diagnoses in this domain were not verified. Table 5.14 shows the information details of the attributes of the Lymphography domain.

TABLE 5.14:  
Lymphography Domain

Attribute	Attribute Information
1	class: normal find, metastases, malign lymph, fibrosis
2	lymphatics: normal, arched, deformed, displaced
3	block of affer: no, yes
4	bl. of lymph. c: no, yes
5	bl. of lymph. s: no, yes
6	by pass: no, yes
7	extravasates: no, yes
8	regeneration of: no, yes
9	early uptake in: no, yes
10	lym.nodes dimin: 0-3
11	lym.nodes enlar: 1-4
12	changes in lym.: bean, oval, round
13	defect in node: no, lacunar, lac. marginal, lac. central
14	changes in node: no, lacunar, lac. margin, lac. central
15	changes in stru: no, grainy, drop-like, coarse, diluted, reticular, stripped, faint
16	special forms: no, chalices, vesicles
17	dislocation of: no, yes
18	exclusion of no: no, yes
19	no. of nodes in: 0-9, 10-19, 20-29, 30-39, 40-49, 50-59, 60-69, >=70

TABLE 5.15:  
Simulation Results for the Lymphography Domain

Methods	Reference	Accuracy (%)		
		Train	Test	Overall
AQ15	Michalski	-	85	-
Expert	Michalski	-	80-82	-
Simple Bayes	Clark	-	83	-
CN2 (99%threshold)	Clark	-	82	-
Assistance86	Cestnik	-	76	-
CLILP2	Liu	-	85	-
ILFN	Meesad	91.26	88.89	90.5
FES	Meesad	85.43	84.44	85.14
HIS	Meesad	92.23	91.11	91.89

TABLE 5.16:  
The Compressibility of Fuzzy Rules for the Lymphography Domain

Data	Comprehensibility					$I_D$
	$L$	$N_A$	$N_L$	$LS$	$RS$	
Lymphography	8	7.9	2.5	0.15	0.56	0

### 5.3.2.1 Simulation Results for the Lymphography Domain

The simulation results from other researchers and this study are shown in Table 5.15. Michalski *et al.* used their AQ15 to obtain 85% correct overall classification performance. In addition, they used an expert system to experiment on the data set and achieved a correct classification range of 80-82%. Clark *et al.* achieved 83% correct classification using a simple Bayes method and achieved 82% correct classification using CN2 system with 99% threshold.

In this study, based on an average of five runs, the ILFN achieved 91.26%, 88.89%, and 90.5% correct classification on training set, test set, and overall, respectively. The FES achieved 85.43%, 84.44%, and 85.14% correct classification. The HIS achieved 92.23%, 91.11%, and 91.89% correct classification. The three methods

performed better than those from the literature. The quantitative measure on the comprehensibility of the fuzzy if-then rules is shown in Table 5.16. The comprehensibility measures of the resulted rules were 8, 7.9, 2.5, 0.15, and 0.56 for  $L$ ,  $N_A$ ,  $N_L$ ,  $LS$ , and  $RS$ , respectively. From the quantitative measures on the comprehensibility, it implied that the resulting fuzzy rules were comprehensible. The incompleteness degree ( $I_D$ ) was low as 0 showing that the rule structure was complete. The resulting linguistic rules for Lymphography data are as follows.

1. IF (*by pass is yes*) and (*early uptake in is yes*) and (*changes in node is lac. central*) and (*changes in stru is reticular or stripped or faint*) and (*special forms is vesicles*) and (*dislocation of is yes*) and (*exclusion of no is yes*) and (*no. of nodes in is High*), Then Class is *malign lymph* (with confidence = 1)
2. IF (*lymphatics is deformed or displaced*) and (*block of affere is yes*) and (*bl. of lymph. c is yes*) and (*lym.nodes dimin is Medium*) and (*lym.nodes enlar is High*) and (*changes in lym. is round*) and (*defect in node is lac. marginal or lac. central*) and (*special forms is chalices*), Then Class is *metastases* (with confidence = 0.99)
3. IF (*bl. of lymph. s is yes*) and (*extravasates is yes*) and (*lym.nodes dimin is High*) and (*lym.nodes enlar is Low*) and (*defect in node is lac. marginal or lac. central*) and (*changes in node is lacunar or lac. margin*) and (*no. of nodes in is High*), Then Class is *fibrosis* (with confidence = 0.96)
4. IF (*block of affere is no*) and (*bl. of lymph. s is no*) and (*by pass is no*) and (*regeneration of is no*) and (*early uptake in is neutral*) and (*lym.nodes dimin is Low*) and (*lym.nodes enlar is Low*) and (*defect in node is no or lacunar*) and (*changes in stru is coarse or diluted*) and (*special forms is no*) and (*dislocation of*

- is *no*) and (*exclusion of no* is *no*) and (*no. of nodes in* is *Low*), Then Class is *malign lymph* (with confidence = 1)
5. IF (*bl. of lymph. c* is *neutral*) and (*regeneration of* is *no*) and (*lym.nodes dimin* is *Medium*) and (*defect in node* is *lac. marginal* or *lac. central*) and (*changes in node* is *lacunar* or *lac. margin*) and (*changes in stru* is *coarse* or *diluted*) and (*no. of nodes in* is *Low*), Then Class is *metastases* (with confidence = 0.82)
  6. IF (*block of affere* is *no*) and (*bl. of lymph. c* is *neutral*) and (*defect in node* is *lac. marginal* or *lac. central*) and (*changes in node* is *lacunar* or *lac. margin*) and (*changes in stru* is *reticular* or *stripped* or *faint*) and (*exclusion of no* is *yes*) and (*no. of nodes in* is *Low*), Then Class is *malign lymph* (with confidence = 0.79)
  7. IF (*block of affere* is *no*) and (*bl. of lymph. s* is *yes*) and (*regeneration of* is *no*) and (*changes in lym.* is *bean*) and (*defect in node* is *no* or *lacunar*) and (*changes in node* is *no*) and (*dislocation of* is *no*) and (*exclusion of no* is *no*), Then Class is *normal find* (with confidence = 1)
  8. IF (*lymphatics* is *deformed* or *displaced*) and (*block of affere* is *no*) and (*by pass* is *no*) and (*special forms* is *vesicles*) and (*no. of nodes in* is *Medium*), Then Class is *malign lymph* (with confidence = 1)

### 5.3.3 Primary Tumor Domain

This is one of three medical domains provided by the Oncology Institute. Primary tumor domain consists of 339 instances with 18 attributes including one class attribute. In this study, 173 patterns and 166 patterns were used for training and testing, respectively. There are 22 classes in this data set. All attribute values in the database have been entered

as numerical values corresponding to their index in the list of attribute values for that attribute domain as given in Table 5.17.

TABLE 5.17:  
Primary Tumor Domain

Attribute	Attribute Information
1	class: lung, head & neck, esophagus, thyroid, stomach, duoden & sm.int, colon, rectum, anus, salivary glands, pancreas, gallbladder, liver, kidney, bladder, testis, prostate, ovary, corpus uteri, cervix uteri, vagina, breast
2	age: <30, 30-59, >=60
3	sex: male, female
4	histologic-type: epidermoid, adeno, anaplastic
5	degree-of-diffe: well, fairly, poorly
6	bone: yes, no
7	bone-marrow: yes, no
8	lung: yes, no
9	pleura: yes, no
10	peritoneum: yes, no
11	liver: yes, no
12	brain: yes, no
13	skin: yes, no
14	neck: yes, no
15	supraclavicular: yes, no
16	axillar: yes, no
17	mediastinum: yes, no
18	abdominal: yes, no

### 5.3.3.1 Simulation Results for the Primary Tumor Domain

The simulation results on the primary tumor domain are shown in Table 5.18. Michalski *et al.* used AQ15 system achieved the correct classification performance in the range of 29-41%. An expert system also was used and it achieved 42% correct classification. Using a simple Bayes network and CN2 with 95% threshold, Clark *et al.* achieved 48% and 45%, respectively. Cestnik *et al.* achieved 44% correct classification using their Assistant86 system.

In this study, we used three systems: ILFN, FES, and HIS. Based on an average of five runs, ILFN constructed 56 hidden nodes for the problem. It achieved 37.99%,

33.33%, and 37.17% correct classification for training set, test set, and overall, respectively. The FES achieved 41.94%, 38.33%, and 41.29% correct classification for training set, test set, and overall, respectively. The HIS achieved 41.67%, 38.35%, and 40.1% correct classification, for training set, test set, and overall, respectively. The comprehensibility of knowledge base is shown in Table 5.19. The comprehensibility measures based on  $L$ ,  $N_A$ ,  $N_L$ ,  $LS$ , and  $RS$  were 55, 8.5, 2.2, 0.2, and 0.46, respectively. The incompleteness degree ( $I_D$ ) was 0. The number of fuzzy rules for this data set was not too high since this data set has 22 classes overall. The average number of rules generated was only about 2.5 fuzzy rules per class. The primary tumor data were very difficult to classify compared to the other two data sets.

TABLE 5.18:  
Simulation Results for the Primary Tumor Domain

Methods	Reference	Accuracy (%)		
		Train	Test	Overall
AQ15	Michalski	-	29-41	-
Expert	Michalski	-	42	-
Simple Bayes	Clark	-	48	-
CN2 (95%threshold)	Clark	-	45	-
Assistance86	Cestnik	-	44	-
CLILP2	Liu	-	37	-
ILFN	Meesad	37.99	33.33	37.17
FES	Meesad	41.94	38.33	41.29
HIS	Meesad	41.67	38.35	40.1

TABLE 5.19:  
The Compressibility of Fuzzy Rules for the Primary Tumor Domain

Data	Comprehensibility					$I_D$
	$L$	$N_A$	$N_L$	$LS$	$RS$	
Primary Tumor	55	8.5	2.2	0.2	0.46	0

The result has shown that the fuzzy knowledge bases had the best accuracy on the Lymphography data set. The fuzzy knowledge base extracted for the primary tumor data was with the lowest accuracy. However, the fuzzy knowledge bases were easy to comprehend, since the number of rules, the number of antecedent per rule, the number of linguistic per dimension, the linguistic similarity degree, and the rule similarity degree were relatively low.

It is worth noting to see that  $LS$ ,  $RS$ , and  $I_D$  are the values in the ranges  $[0, 1]$ . A FES that has the values of  $LS$  and  $RS$  close to zero implies that the FES is highly comprehensible. On the other hand, if  $LS$  and  $RS$  are close to one, the FES may be difficult to understand. If a FES has the value of  $I_D$  equals to zero, it implies that the rule structure of the FES is complete. However, the objectives of reducing  $RS$  and  $I_D$  are conflicting to each other. When using the GA to search for fuzzy rules with small  $RS$ , the incompleteness degree may be increased. The incompleteness degree is considered low if the  $I_D$  is less than 0.1. This may be acceptable for pattern classification problems. However, in function approximation or control system, the  $I_D$  which is larger than zero may not be acceptable. An incomplete system may result in an undesirable behavior of the system output.

For the comparisons among several methods, such as Assistant-86, Bayes, AQR, CN2, AQT-15, and CLILP2, it is found that the FES achieved competitive performance with others. Except for the last data set, the primary tumor FES achieved lower accuracy based on the test set. One reason is that the FES is not focused solely on the accuracy. While searching for fuzzy rules with acceptable accuracy performance, we maintain the comprehensibility of the fuzzy sets as well as the completeness of the fuzzy rule

structure. Obtaining a highly comprehensible and complete system results in losing the accuracy of the system. Another reason of having lower accuracy performance on the primary tumor may be because we replaced missing values with zeros while the others used some things else. With diverse choices of the numerical values to replace the missing ones, the decision of the system can be made differently. To do a good job for a 22-class of primary tumor, we should use more data in the training set to improve accuracy.



## CHAPTER VI

### DEVELOPMENT OF A HIS GRAPHICAL USER INTERFACE

To provide a user-friendly interface of the proposed HIS, a HIS graphical user interface (HIS-GUI) was developed under the Matlab programming environment. The HIS-GUI framework integrates the ILFN and the FES as well as a GA optimization technique. Some medical data sets used in the study will be included in the HIS-GUI framework. The main frame of the HIS-GUI is shown in Figure 6.1.

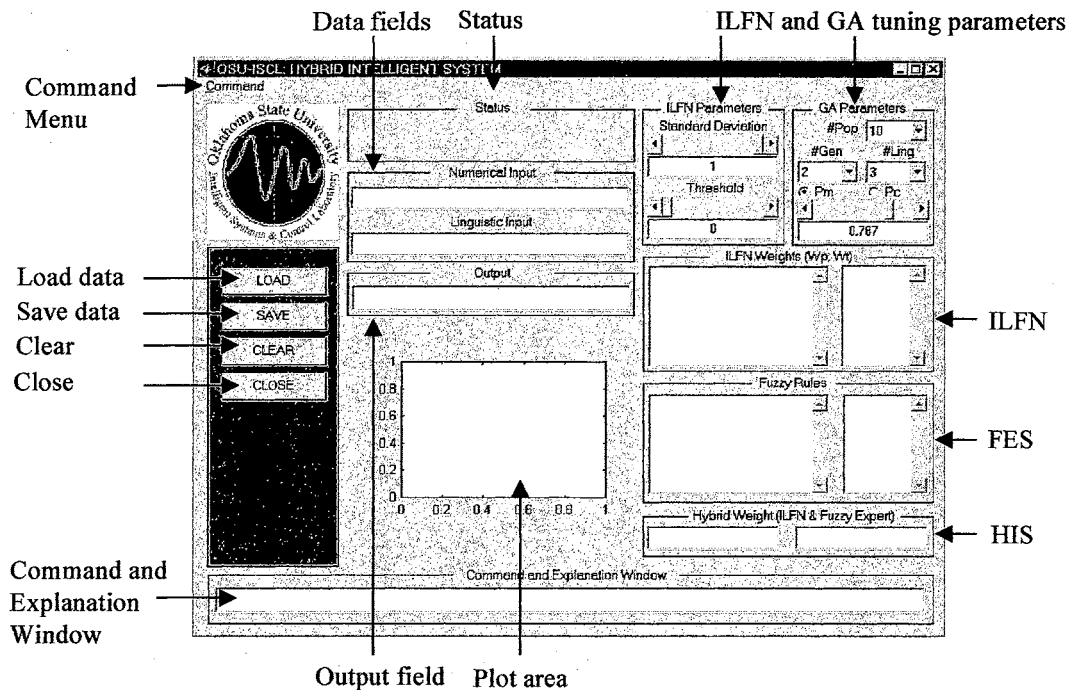


Figure 6.1: HIS Graphical User Interface

#### 6.1 The Main Window of HIS-GUI

The main window of HIS-GUI composes of several areas: 1) command menu, 2) data field, 3) status of the system, 4) ILFN and GA tuning parameters, 5) load button,



When the Load Data command is selected, the LOAD DATA dialog is appeared, as shown in Figure 6.4. The data files in the current directory will be listed in the dialog. The user can type in a file's name or select a file from the list to load data to the system. After a file is typed in or selected, the user can click the OK button to load the data file.

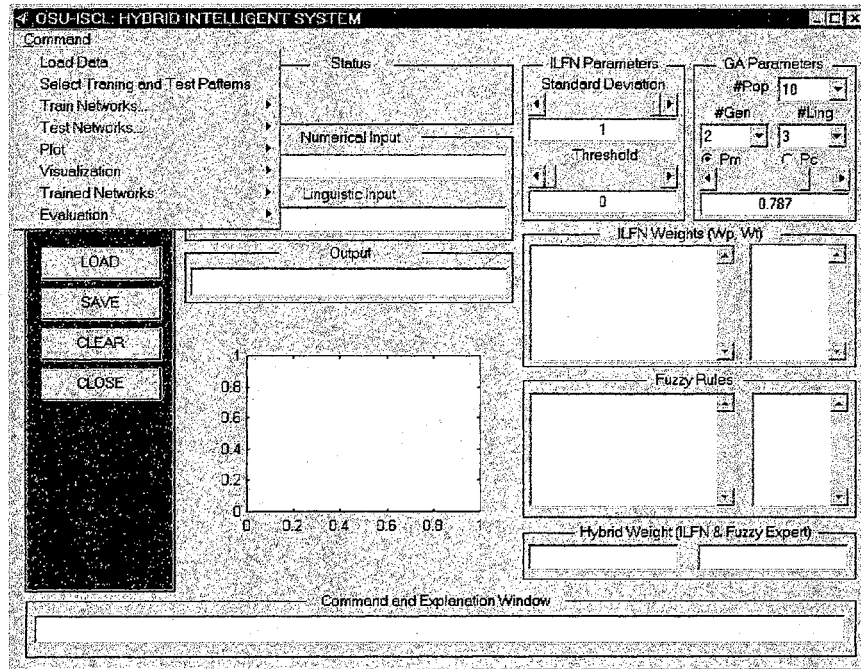


Figure 6.3: Main Menu of the GUI

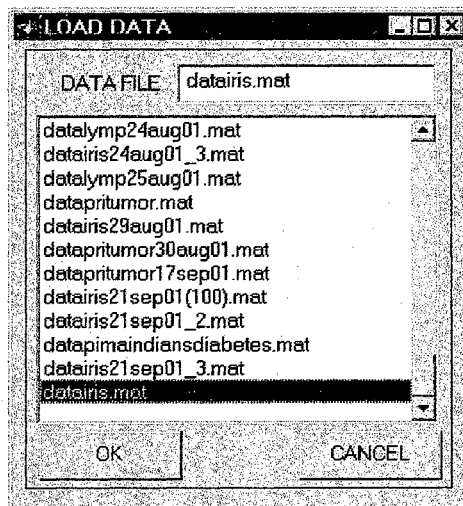


Figure 6.4: Dialog for Load Data

### 6.3 Setting the Number of Training and Test Patterns

After data is loaded, the user can choose the number of training patterns and the number of test patterns, by selecting a command “Select Training and Test Patterns” from the command menu. The “Set Number of Training and Test Patterns” dialog is shown in Figure 6.5. The total number of patterns will be shown in “#Patterns” text field. The number of training patterns can be specified in the “Training Patterns” text field. The number of the test patterns can be specified in the “Test Patterns” text field. For example, in Figure 6.5, there are 150 patterns in the data set loaded. The training patterns are from the patterns #1 to #75 and the test patterns are from patterns #76 to #150. The user can reorder the training data by click at the “Reorder Training Data” button.

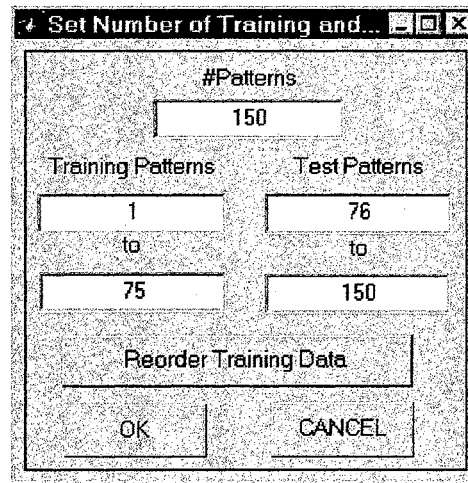


Figure 6.5: Dialog for Setting the Number of Training and Testing Data

### 6.4 Setting ILFN Tuning Parameters

The ILFN network needs two tuning parameters: initial standard deviation  $\sigma_0$  and threshold  $\epsilon$ . The initial standard deviation is set default to 1 and the threshold is set

default to 0. To change the values of the initial standard deviation and the threshold, click at the left or right arrow of the standard deviation and the threshold slider bars. An alternative way is to type in a number directly to the edit field under the standard deviation slider bar and the threshold slider bar.

### 6.5 Training ILFN Network

After the number of training data has been selected and the ILFN tuning parameters have been set, it is ready to train an ILFN network. To train the ILFN network select “Train Network” and “Train ILFN” from the command menu, as shown in Figure 6.6. The resulting network’s weights will appear in the ILFN weights area, as shown in Figure 6.7.

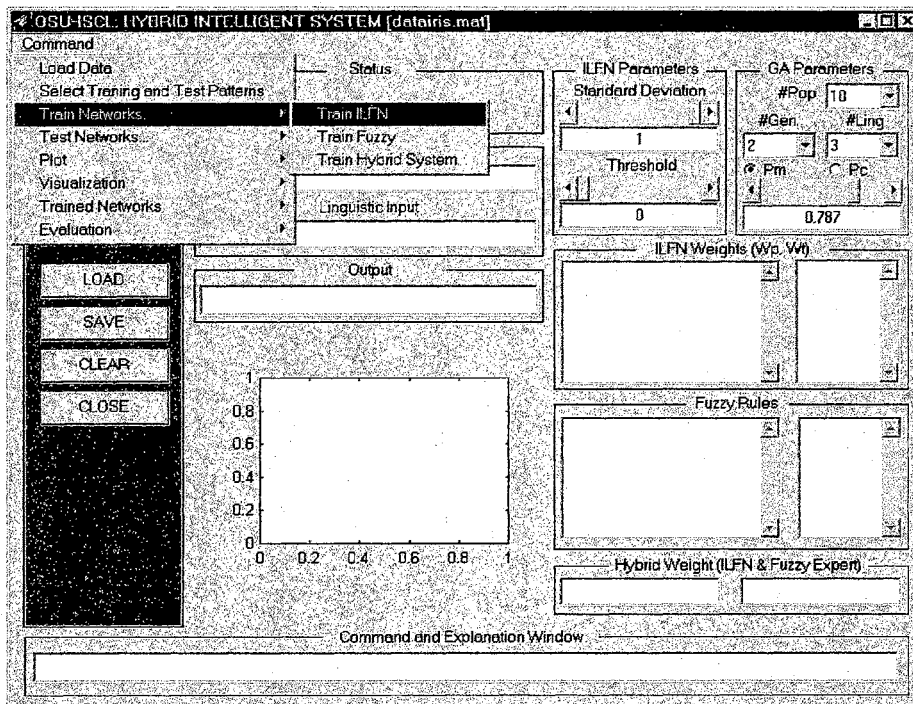


Figure 6.6: Training ILFN Network

Figure 6.7 illustrates the trained ILFN weights. The weights  $W_P$  and  $W_T$  are shown in matrix forms. Weight  $W_P$  is located on the left side and weight  $W_T$  is located on the right of the area. The number of columns of  $W_P$  matrix is the number of dimension of the input patterns. The number of rows of  $W_P$  matrix is the number of hidden nodes of the ILFN. Each corresponding target of each node is placed in each row of  $W_T$ .

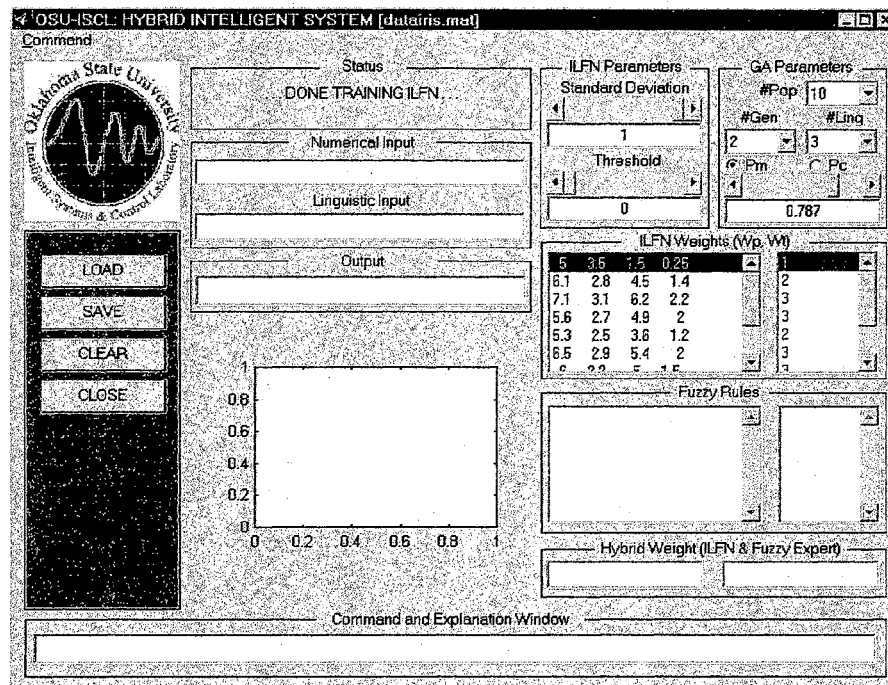


Figure 6.7: ILFN Weights Appeared in the Main Window

The user can add or delete the weights of the trained ILFN. To add or edit the weights of a trained ILFN, select "Trained Networks" and then "ILFN Network" from the command menu. The command will bring the ILFN window, as shown in Figure 6.8.

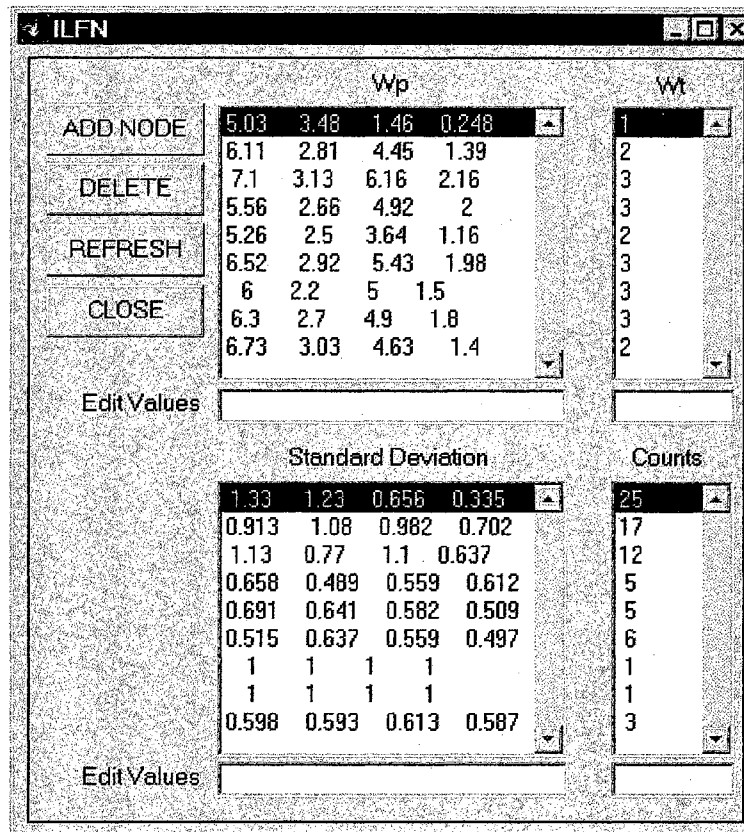


Figure 6.8: The Trained ILFN Weights and Parameters

### 6.6 ILFN Evaluation

To evaluate a trained ILFN, choose “Evaluation” and then “ILFN Evaluation” from the command menu, as shown in Figure 6.9. The ILFN will be evaluated using the test patterns specified in the “Set Number of Training and Test Patterns” dialog. The ILFN evaluation dialog will appear. The results from the evaluation will be shown in evaluation dialog, as shown in Figure 6.10.

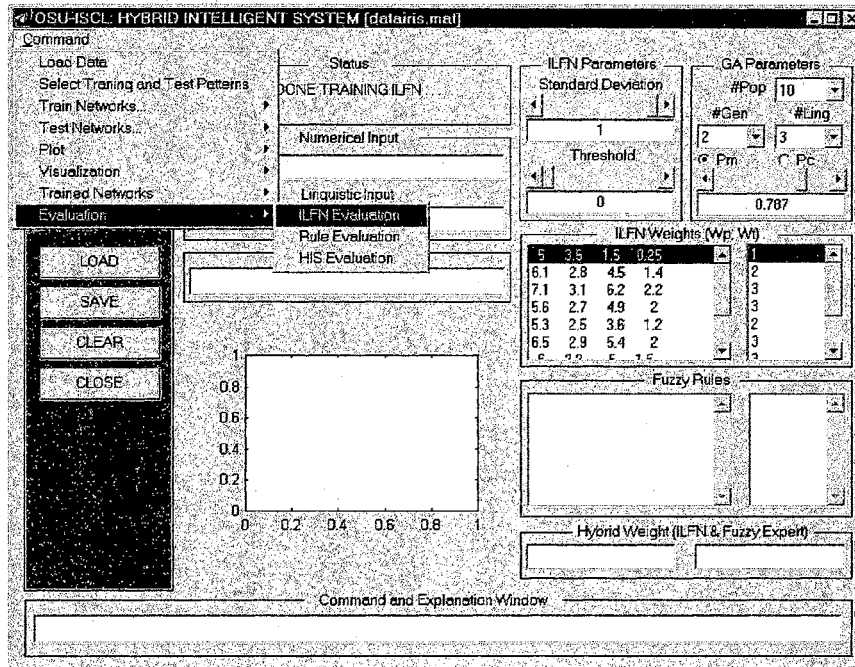


Figure 6.9: Evaluation of the ILFN by the Test Data

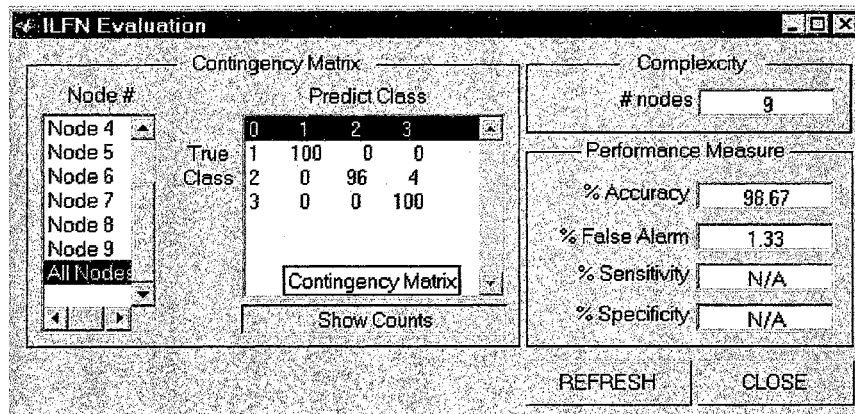


Figure 6.10: ILFN Evaluation Dialog

In the ILFN evaluation dialog, there are main areas of information: contingency matrix, complexity, and performance measure. The contingency matrix shows the details of the accuracy performance of the ILFN. The list of hidden nodes of the ILFN is shown in the left side of the contingency matrix. When the user click at a node of the ILFN, the accuracy performance of the selected node will appear in the contingency matrix. For



example in Figure 6.11, node #1 is selected. The accuracy performance of node #1 is displayed in the contingency matrix. The first column and the first row of the contingency matrix indicate the classes of the data. For example in Figure 6.11, there are three classes: classes 1, 2, and 3. The elements inside the contingency matrix are the accuracy in count or percentage depending on the option selected to show. In Figure 6.11, it shows that node #1 predicts class 1 as class 1 with 25 test patterns. If the user decides to show the accuracy in percentage, click at “Show Percentage” button under the contingency matrix the elements inside the contingency matrix will change to a percentage format, as shown in Figure 12. Percentage and count will be interchanged when the button under the contingency matrix is clicked.

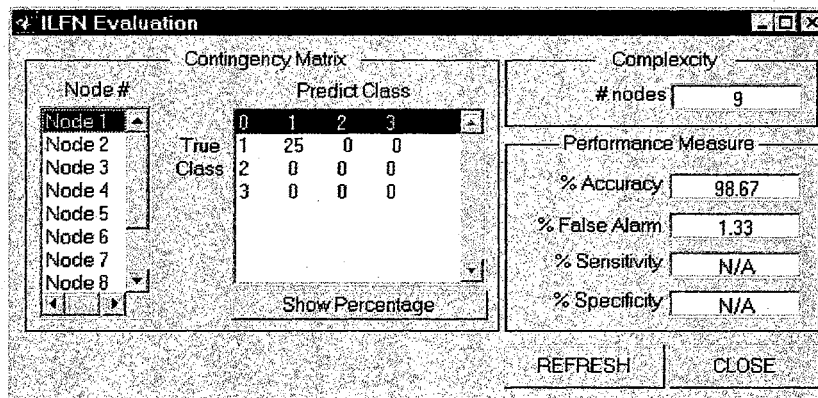


Figure 6.11: ILFN Performance Based on Individual Node in Count

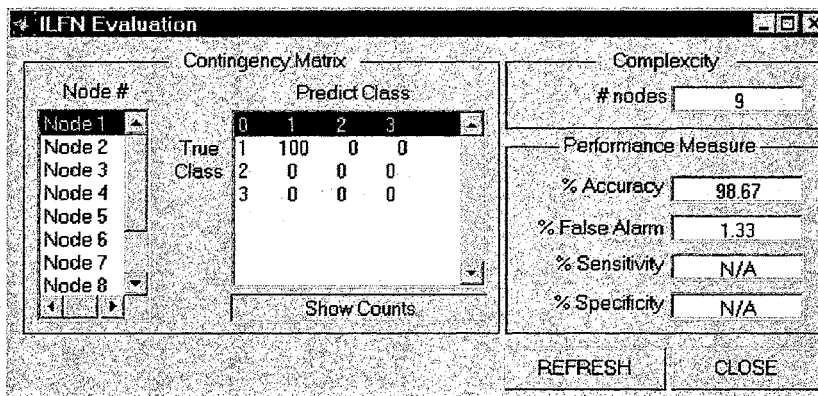


Figure 6.12: ILFN Performance Based on Individual Node in Percentage

From Figure 6.12, the complexity area shows the number of hidden nodes. The performance measure shows the overall accuracy and false alarm in percentage. Percentage of sensitivity and specificity are also shown in the performance measure area but they are available only in the binary classification problems.

### 6.7 Setting GA Parameters and Training FES

GA parameters need to be decided before training FES. As shown in Figure 6.13, the parameters for GA optimization include: the number of GA population, the number of generation to terminate the GA process, the constrained number of linguistic labels for FES, the mutation probability and the crossover probability. After the GA parameters are specified, the FES can be started to train by choosing “Train Networks” and then “Train Fuzzy” as shown in Figure 6.14.

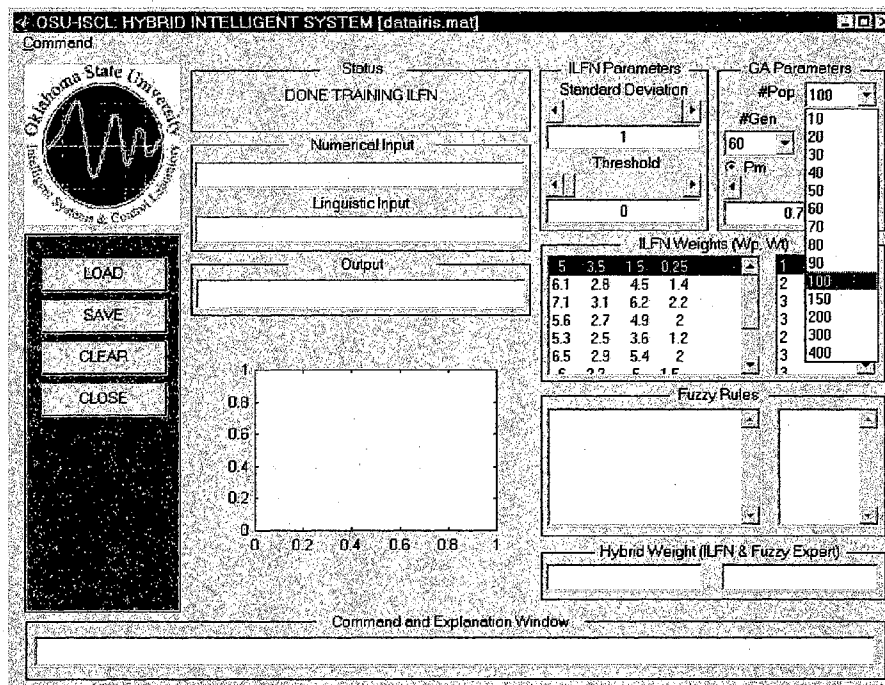


Figure 6.13: Select GA Parameters before Training to FES and HIS

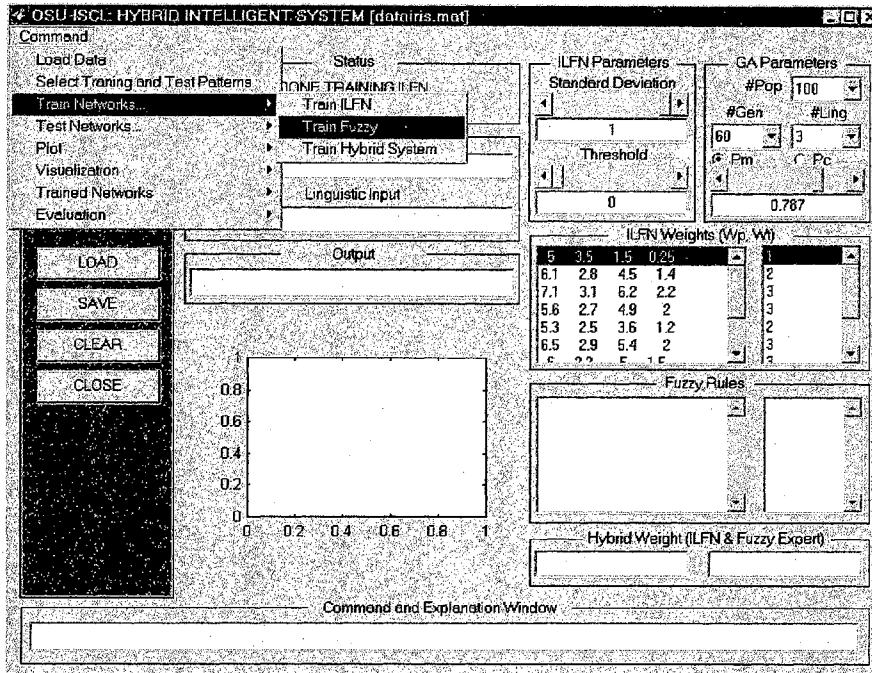


Figure 6.14: Select Train Fuzzy to Begin Training FES

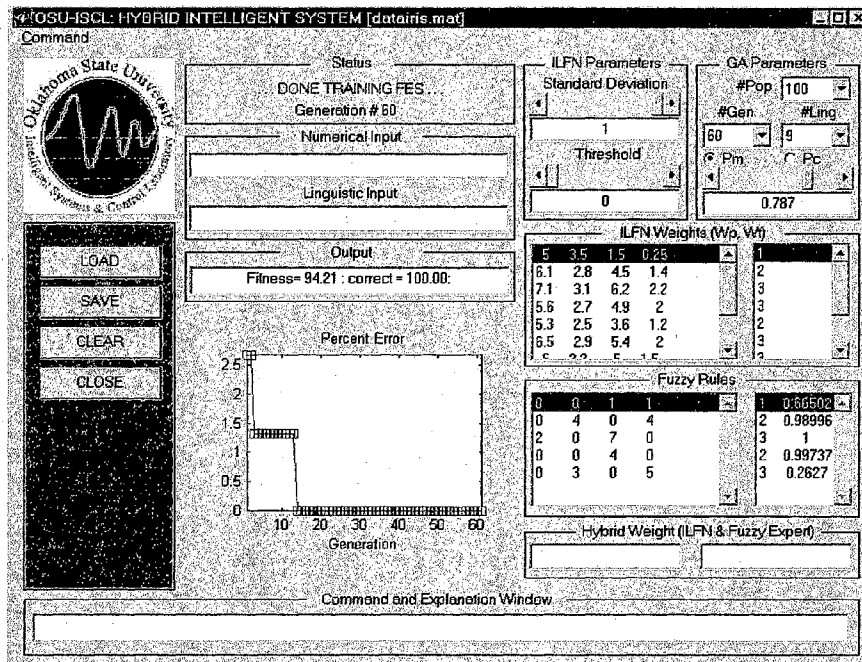


Figure 6.15: Result after Training FES

The FES will be trained based on the weights connections of the ILFN network by the GA optimization technique. (The details how the ILFN and GA are used to construct the knowledge of FES can be found in Chapter 3.) While FES is training, the current fuzzy rules are displayed in the “Fuzzy Rules” area. The current fitness value and the accuracy performance are displayed in the output field. The percent error will be displayed in the plot area. Figure 6.15 shows the result FES after trained.

### 6.8 FES Evaluation

After the FES has been trained, to evaluate the fuzzy if-then rules, choose “Evaluation” and then “Rule Evaluation” from the command menu, as shown in Figure 6.16. The results of the rule evaluation is will be appeared in the rule quality dialog, as shown in Figure 6.17. Figure 6.17 shows the rule evaluation dialog comprising of the contingency matrix, comprehensibility measure, and the performance measure.

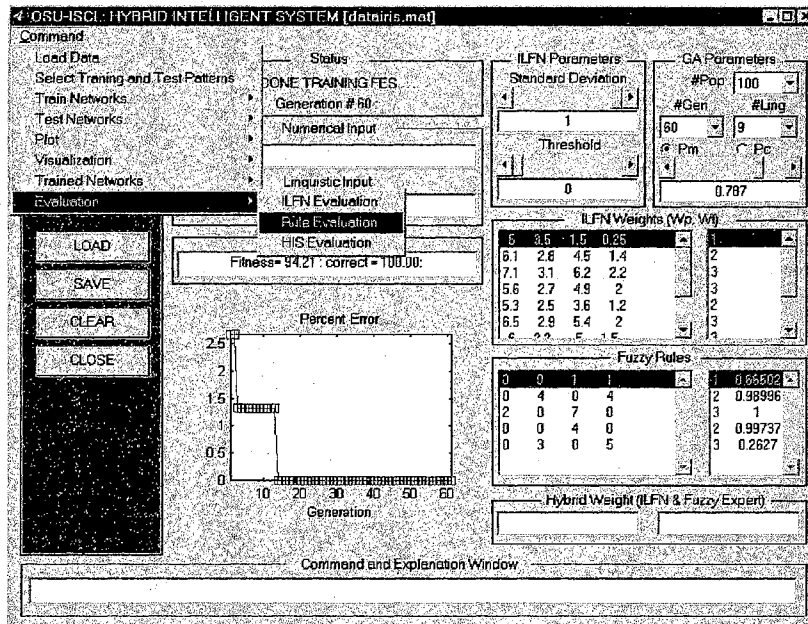


Figure 6.16: Evaluation of the FES by the Test Data

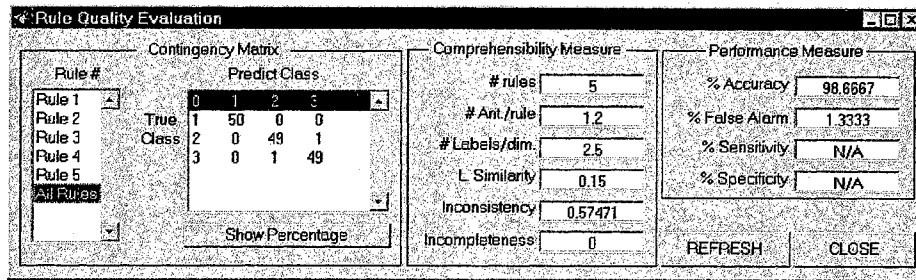


Figure 6.17: Rule Quality Evaluation Dialog

Rule quality evaluation dialog, as shown in Figure 6.17, contains three main parts: a contingency matrix, the comprehensibility measure, and the performance measure. The contingency matrix shows the details of the accuracy performance of the FES. The list of fuzzy rules of the FES is shown in the left side of the contingency matrix. When the user click at a rule of the FES, the accuracy performance of the selected rule will appear in the contingency matrix. For example in Figure 6.17, “All Rules” is selected. The accuracy performance of the whole fuzzy rules is displayed in the contingency matrix. The first column and the first row of the contingency matrix indicate the classes of the data. For example in Figure 6.17, there are three classes: classes 1, 2, and 3. The elements inside the contingency matrix are the accuracy in count or percentage depending on the option selected to show. In Figure 6.17, it shows that fuzzy rules predict class 1 as class 1 with 50 out of 50 test patterns. They predict class 2 as class 2 and class 3 as class 3 with 49 out of 50 patterns. If the user decides to show the accuracy in percentage, click at “Show Percentage” button under the contingency matrix the elements inside the contingency matrix will change to the percentage format. Percentage and count will be interchanged when the button under the contingency matrix is clicked.

## 6.9 Fuzzy Knowledge Base

The knowledge base of the FES can be shown in details by selecting “Trained Networks” and then “Fuzzy Knowledge Base” from the command menu, as shown in Figure 6.18. The fuzzy knowledge base dialog of the FES is shown in Figure 6.19. The fuzzy knowledge base dialog comprises of several parts: refresh button, edit values area, view rule surface button, and close button. The refresh button will be used to redisplay all the information of the fuzzy knowledge base. The edit values area contains many buttons that are used to update the following parameters: features, value’s ranges, number of linguistics, linguistic labels, membership functions, membership functions parameters, and fuzzy rules. The view fuzzy rule surface button is used to show the rule viewer dialog, discussed in Section 6.12. The close button is used to close the fuzzy knowledge base dialog.

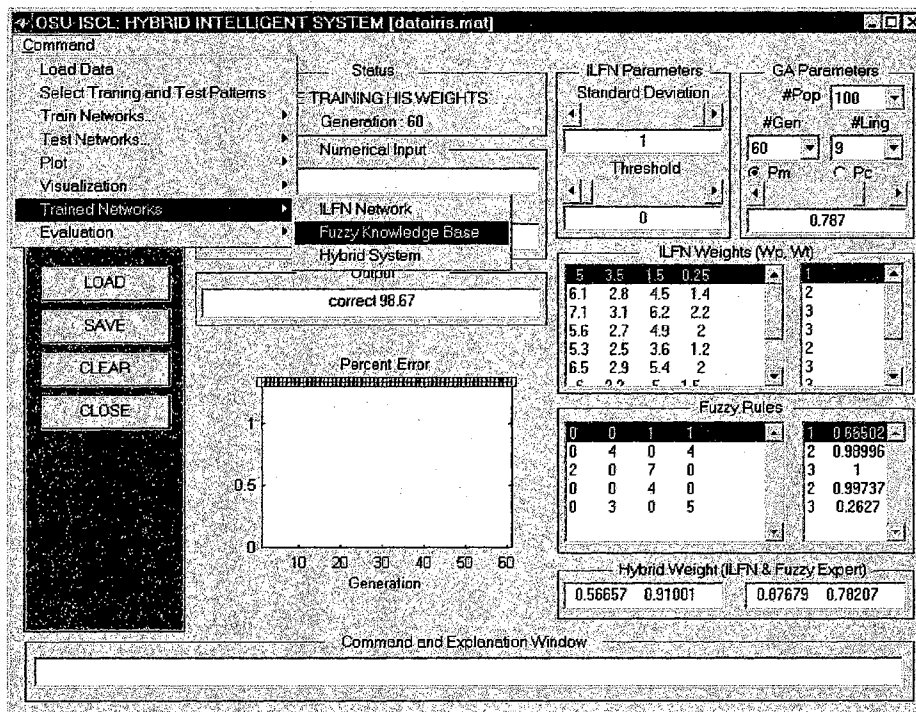


Figure 6.18: Selecting Fuzzy Knowledge Base from Command Menu

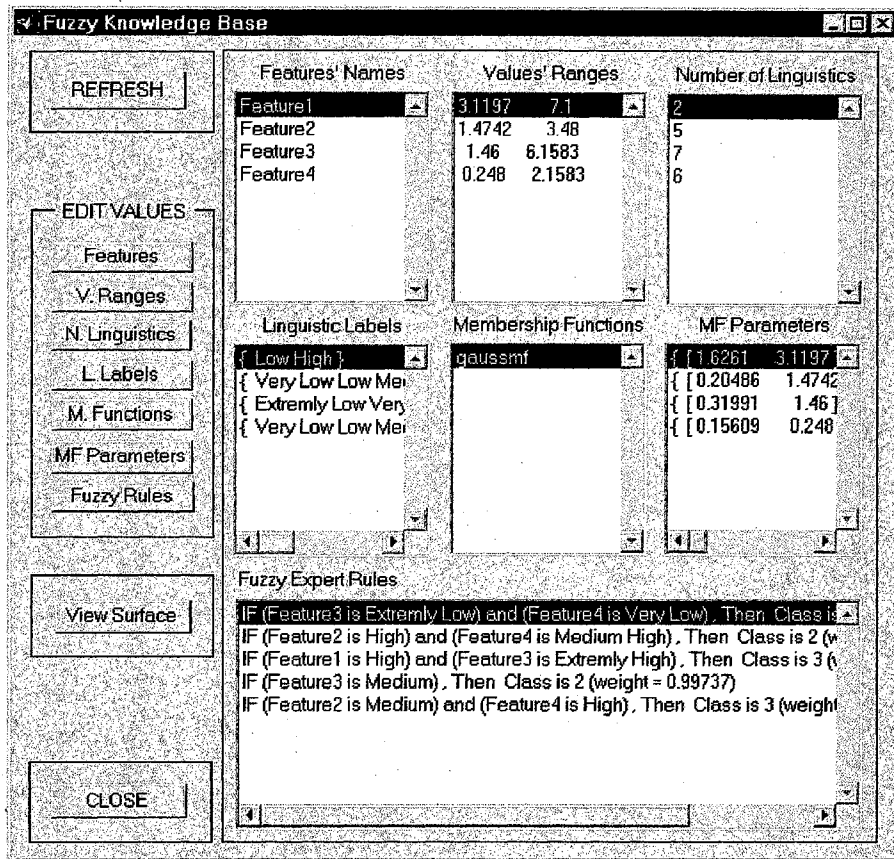


Figure 6.19: Fuzzy Knowledge Base Window

The right side of the fuzzy knowledge base dialog in Figure 6.19 is the information area of the fuzzy knowledge base. It comprises of feature's names, values' ranges, the number of linguistics, linguistic labels, membership functions, membership functions parameters, and fuzzy expert rules. These parameters are extracted from the ILFN network and optimized by the GA method.

If preferred, the user can edit the parameters by choosing a button from the edit values area. For example, if the user wants to change the feature's names that are appropriate for the problem under study, the user has to select "Features" button from the edit values areas. The "Edit Feature's Names" dialog will appear, as shown in Figure 6.20. In the "Edit Feature's Names" dialog, the user can select, from the right side of the

“Feature Name” label, a feature or a dimension of the data and retype a new name in the edit area on the right side of the “Enter New Name” label. Click at “REFRESH” will update the new feature’s name. Repeat the procedure to change the other names. Click “CLOSE” button when finish will return to the main fuzzy knowledge base dialog.

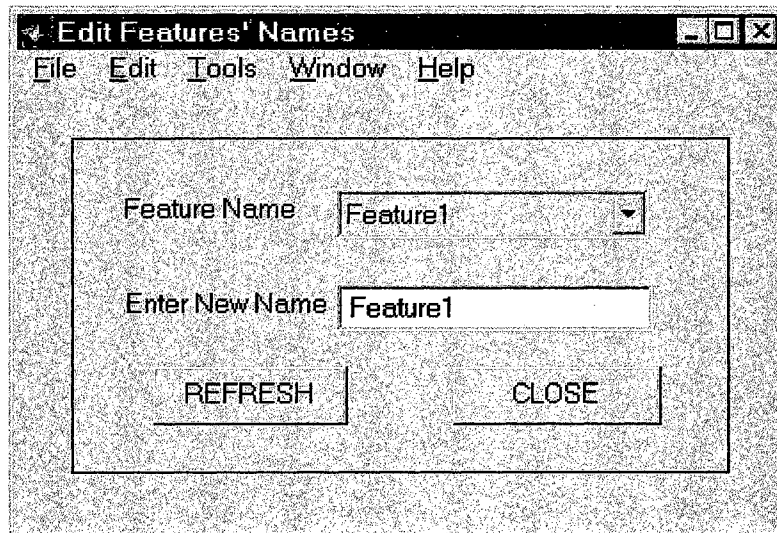


Figure 6.20: Edit Feature’s Names Dialog

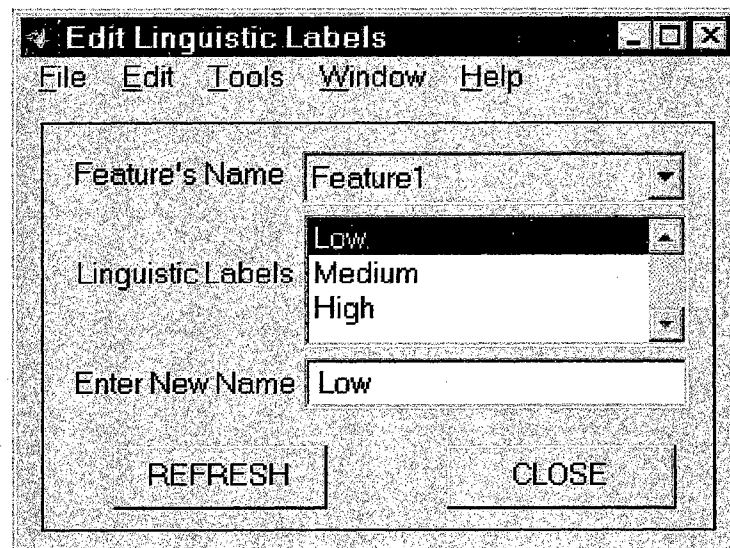


Figure 6.21: Edit Linguistic Labels Dialog



Figure 6.21 illustrates the “Edit Linguistic Labels” dialog. The user can change the names of the linguistic labels suitable for the problem. To change a linguistic label, the user click at the selected linguistic label and type a new name in the edit area on the right side of the “Enter New Name” label. Click “REFRESH” button when finish will update the new linguistic label. Repeat the update process for other linguistic labels. To close the dialog, click at “CLOSE” dialog.

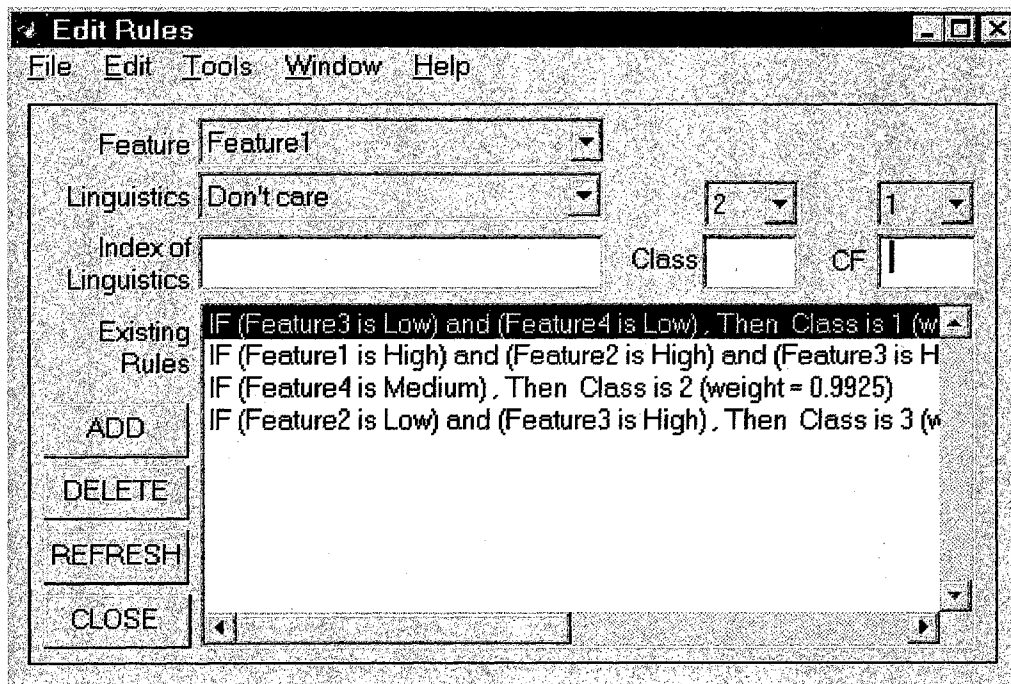


Figure 6.22: Edit Fuzzy Rules Dialog

The user can also edit fuzzy rules by using “Edit Rules” dialog. To edit fuzzy rules, click the “Fuzzy Rules” button in the “EDIT VALUES” area will open the “Edit Rules” dialog, as shown in Figure 6.22. From the “Edit Rules” dialog, the user can add and delete fuzzy rules. To add a rule, select feature and linguistics from the pop up menu on the right of the “Feature” and “Linguistics” labels. Then select or type the class consequent and the confident factor (CF) for the rule. Click the “ADD” button when

finish. The new rule will be added to the fuzzy rule base provided that the new fuzzy rule does not conflict with the existing rules. (Please note that when a new fuzzy rule is added to the fuzzy rule base system, several new hidden nodes may be added in the ILFN networks.) Figure 6.23 illustrates a fuzzy rule is added to the fuzzy rule base. To delete a fuzzy rule, the user can click at a rule to be deleted then click at “DELETE” button. The selected rule will be deleted from the fuzzy rule base. Figure 6.24 shows several fuzzy rules that have been deleted from the fuzz rule base.

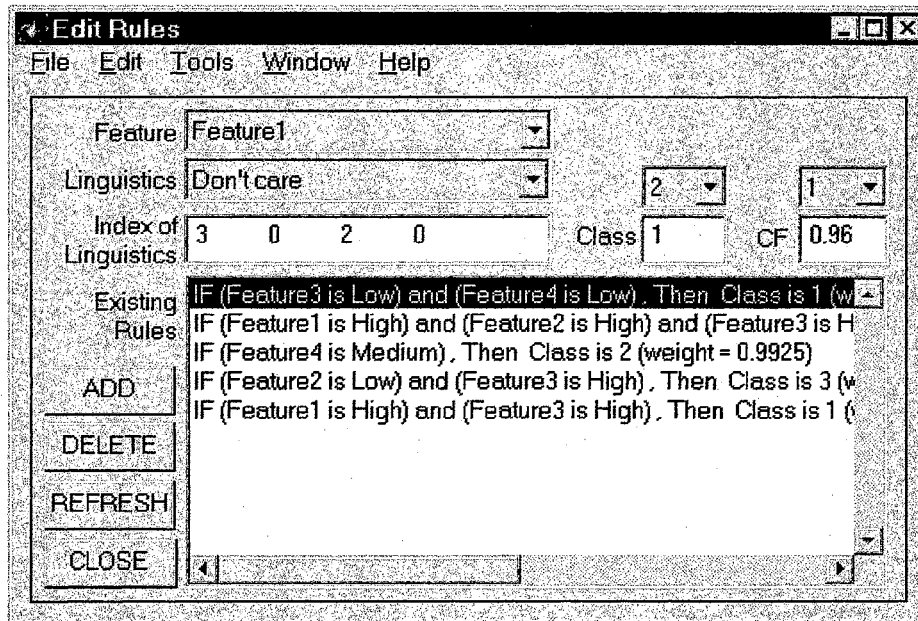


Figure 6.23: A New Rule is Added in the Fuzzy Rule Base

The current fuzzy rules can be viewed in the rule surface window by selecting the “Rule Surface” button in the “Fuzzy Knowledge Base” dialog. The fuzzy rule surface viewer is shown in Figure 6.25. The rule surface viewer can view two inputs at a time in a three dimensions plot. The user can select an input pair, a plot style, shading, and color map. Rule surface viewer helps the user to visualize the concept of the fuzzy rule structure better.

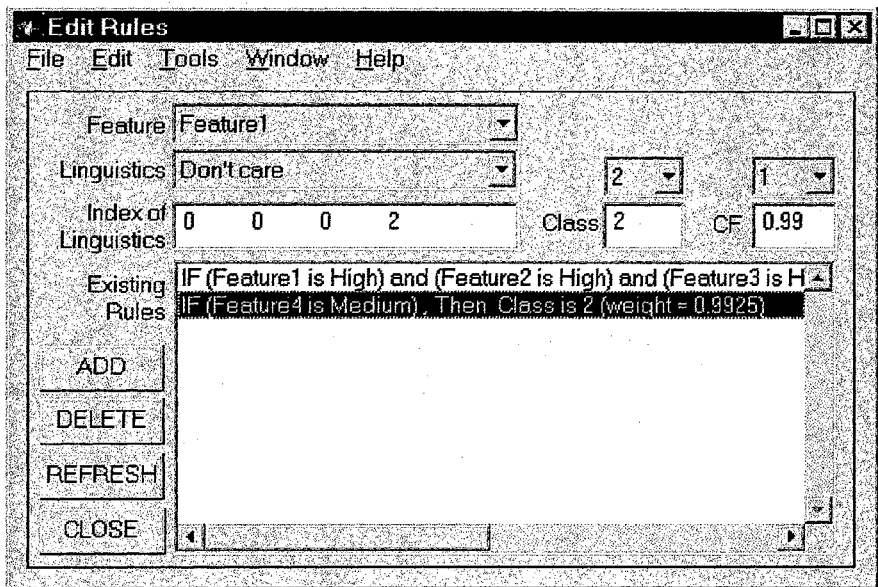


Figure 6.24: Many Rules are Deleted from the Fuzzy Rule Base

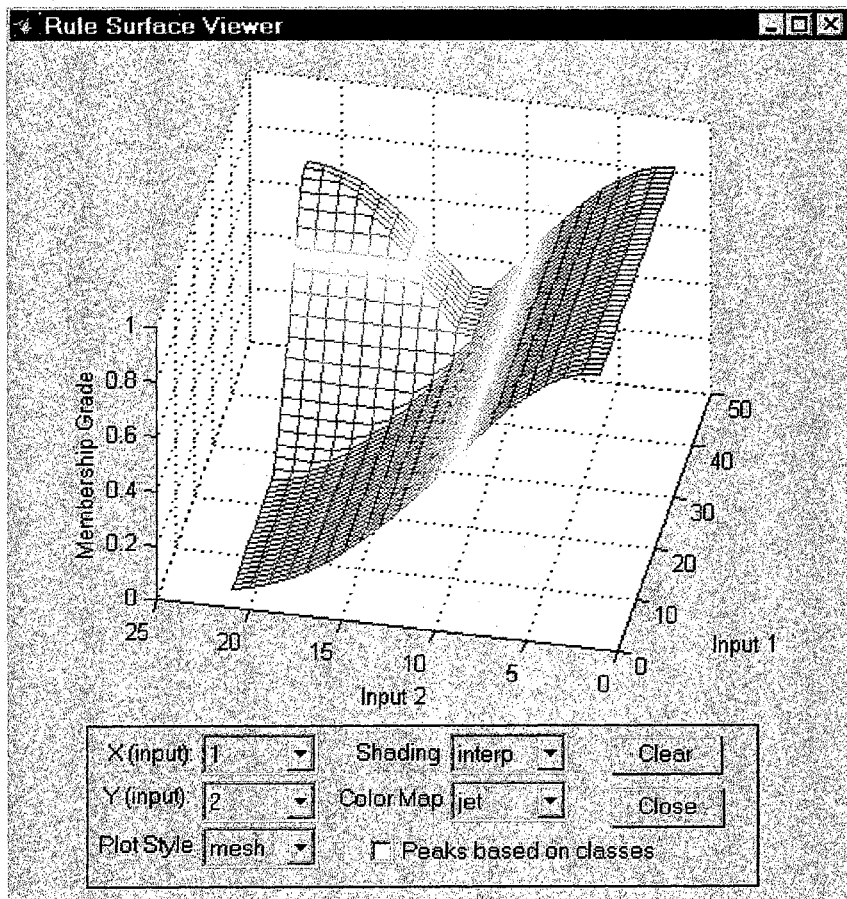


Figure 6.25: Fuzzy Rule Surface Viewer

## 6.10 Training HIS

After ILFN and FES have been trained. ILFN and FES will be combined together to become a HIS. The weighted link between the ILFN and the FES need to be trained. A GA is used to train the HIS weights. GA parameters are selected before training. To start the training session, select “Train Networks” and then “Train Hybrid System” from the command menu, as shown in Figure 6.26. When the training process is running, the current weights will appear in the “Hybrid Weight” area. The error will be plotted along the plot axis. The current fitness values and the percentage of correct classification of the training patterns are appeared in the “output” text field. The GA will terminate when the number of generations has been reached. The final generation of the trained HIS is shown in Figure 6.27.

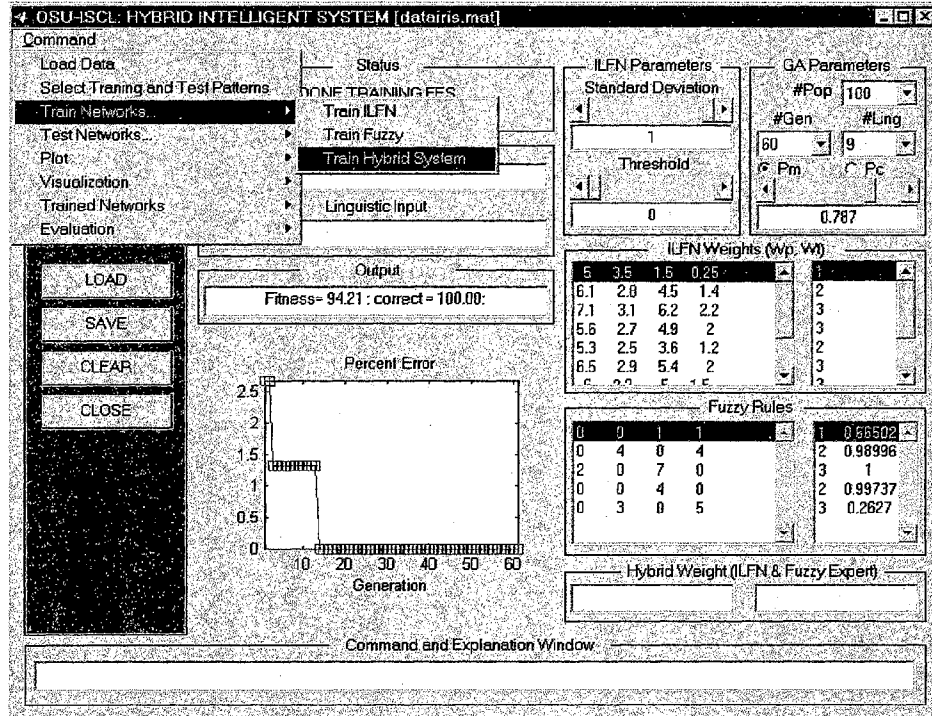


Figure 6.26: Select Train HIS to Begin Training HIS

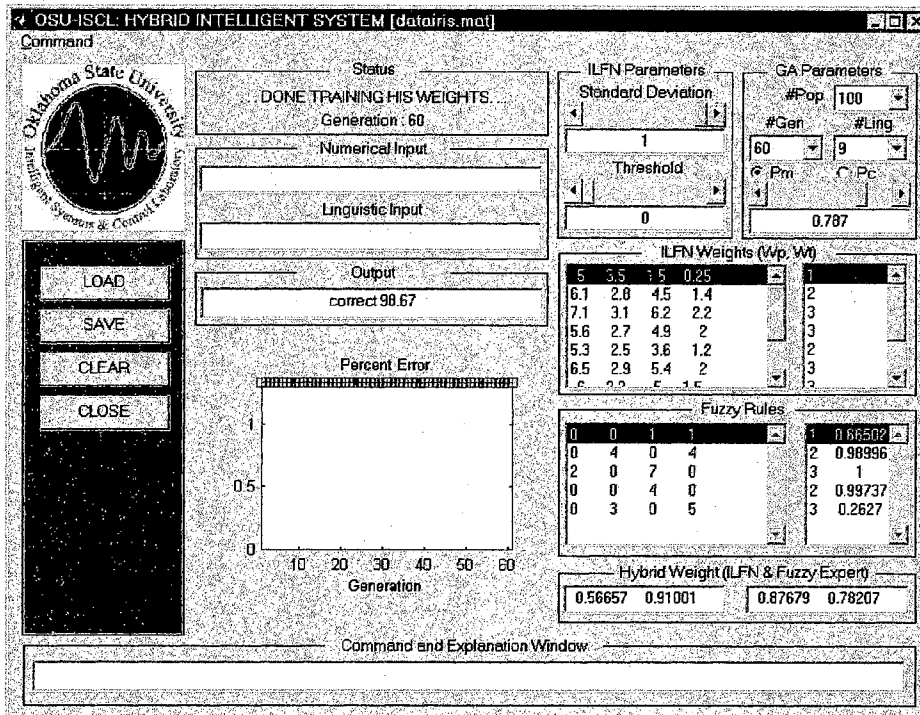


Figure 6.27: After Training HIS

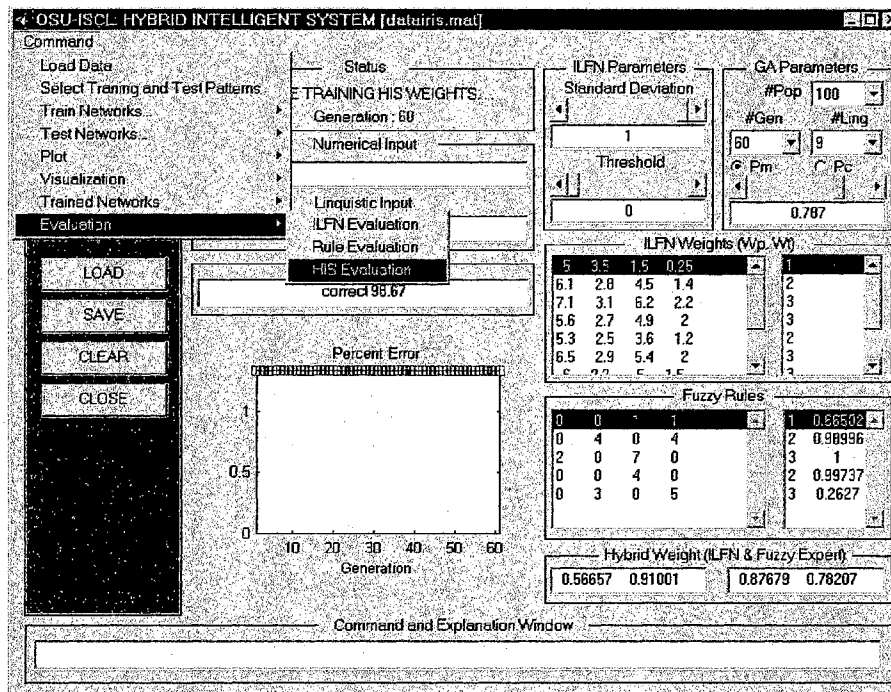


Figure 6.28: Evaluation of the FES by the Test Data

## 6.11 Evaluation of HIS

To evaluate a trained HIS, select “Evaluation” and then “HIS Evaluation” from the command menu, as shown in Figure 6.28. The results of the HIS evaluation will appear in the “HIS Evaluation” dialog, as shown in Figure 6.29. Figure 6.29 shows the HIS evaluation dialog comprising of the contingency matrix, conflicting matrix, complexity, and the performance measure.

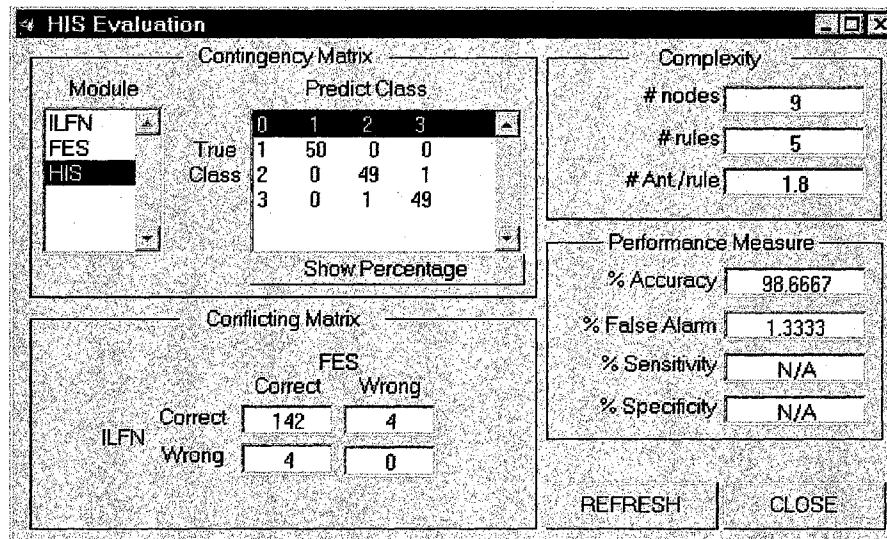


Figure 6.29: ILFN Evaluation Dialog

The HIS evaluation dialog, as shown in Figure 6.29, contains four main parts: a contingency matrix, the conflicting matrix, the complexity measure, and the performance measure. The contingency matrix shows the details of the accuracy performance of the HIS. The models, ILFN and FES, of the HIS are listed in the left side of the contingency matrix. When the user click at a model of the HIS, the accuracy performance of the selected model will appear in the contingency matrix. For example in Figure 6.29, HIS is selected. The accuracy performance of the combination of ILFN and FES is displayed in

the contingency matrix. The first column and the first row of the contingency matrix indicate the classes of the data. For example in Figure 6.29, there are three classes: classes 1, 2, and 3. The elements inside the contingency matrix are the accuracy in count or percentage depending on the option selected to show. In Figure 6.29, it shows that HIS predict class 1 as class 1 with 50 out of 50 test patterns. It predicts class 2 as class 2 and class 3 as class 3 with 49 out of 50 patterns. If the user decides to show the accuracy in percentage, click at “Show Percentage” button under the contingency matrix, the elements inside the contingency matrix will change to the percentage format. Percentage and count will be interchanged when the button under the contingency matrix is clicked.

## 6.12 Test Networks

Similar to the “Evaluation” command menu, the “Test Networks” command menu, as shown in Figure 6.30, provides a way to evaluate the trained networks. However, only the accuracy performance is evaluated. Unlike the “Evaluation” command menu, the “Test Networks” command menu will show the predicted result of the current pattern in the main window. The user can immediately know to which class the current pattern belongs. If the ILFN is tested, a highlight bar appears at the ILFN winning node to which the pattern is classified. Similarly, if the FES is tested, a highlight bar appears at the FES winning rule to which the current pattern is classified and an “IF-THEN” explanation message appears in the “Command and Explanation” window. When the HIS is tested, two highlight bars appear at the ILFN winning node and the FES winning rule which are the current pattern belongs to. In addition, an “IF-THEN” explanation message appears in the “Command and Explanation” window.

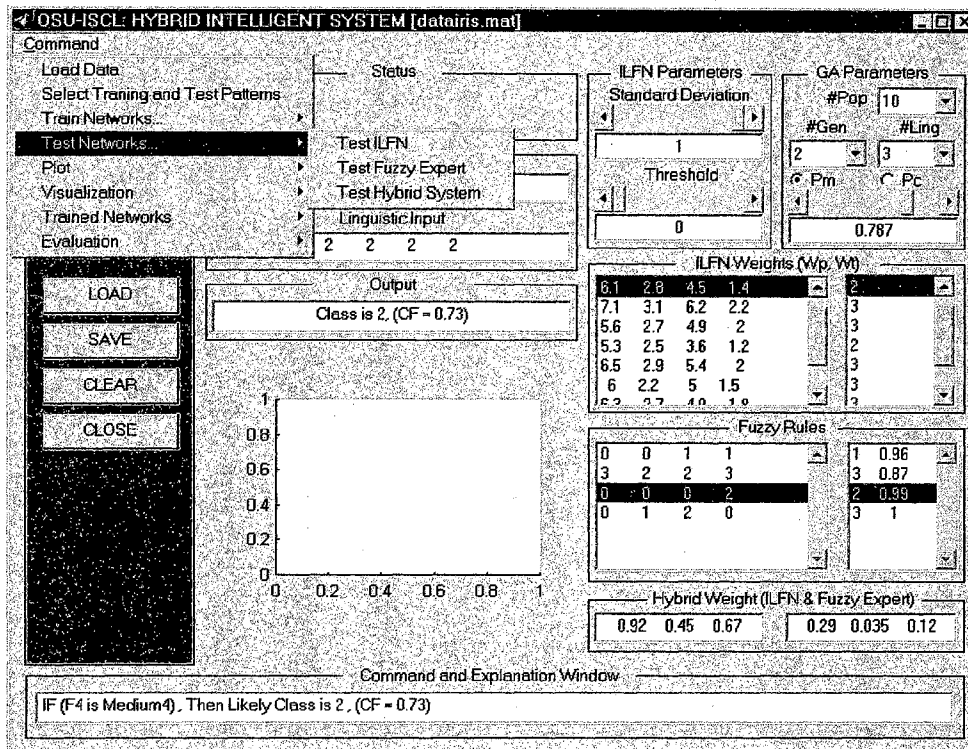


Figure 6.30: Test Networks Command Menu

### 6.13 Plot Command Menu

Plot command menu provides three commands: “Plot ILFN,” “Plot Fuzzy Rules (Initial),” and “Plot Fuzzy Rules (Optimized).” The “Plot ILFN” command will plot the hidden weight of the ILFN in the plot area, as shown in Figure 6.31. The “Plot Fuzzy Rules” commands will open a fuzzy rule window and plot linguistic terms of the fuzzy rules, as shown in Figures 6.32 and 6.33. Figure 6.32 shows initial fuzzy rules, which are directly extracted from a trained ILFN without the GA optimization. Figure 6.33 shows a final fuzzy rule set that has been optimized by GA optimization. From Figures 6.32 and 6.33, each row of the figures represents antecedents of a fuzzy rule and each column represents an attribute or a dimension of the rule. Consequents of the fuzzy rule appear on the left of each rule, for example, C1(1) means “class 1 with confident factor 1.”



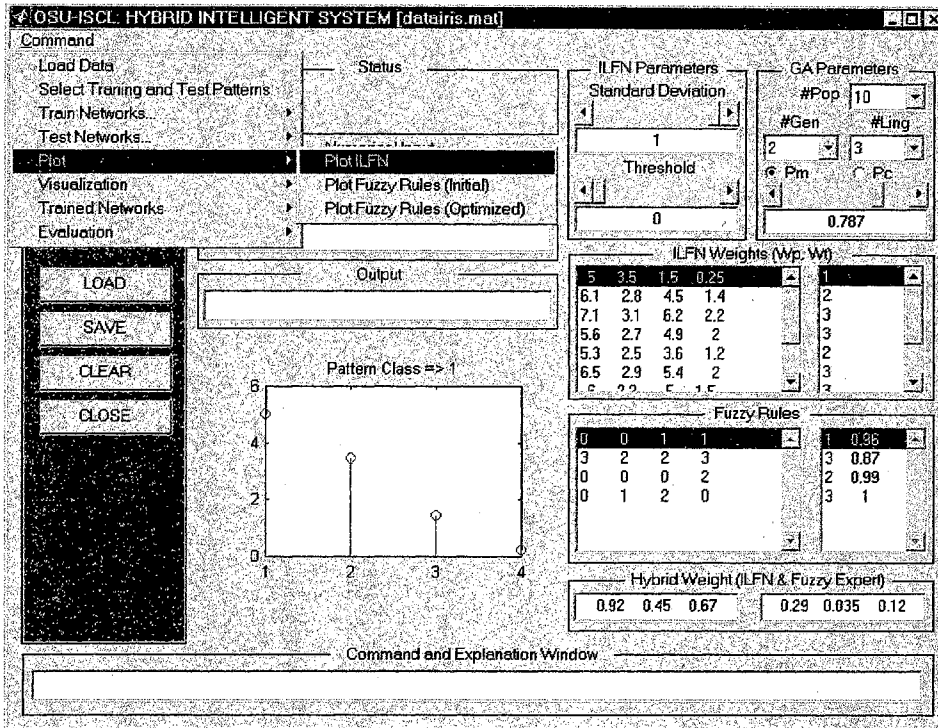


Figure 6.31: A Plot of ILFN Weight

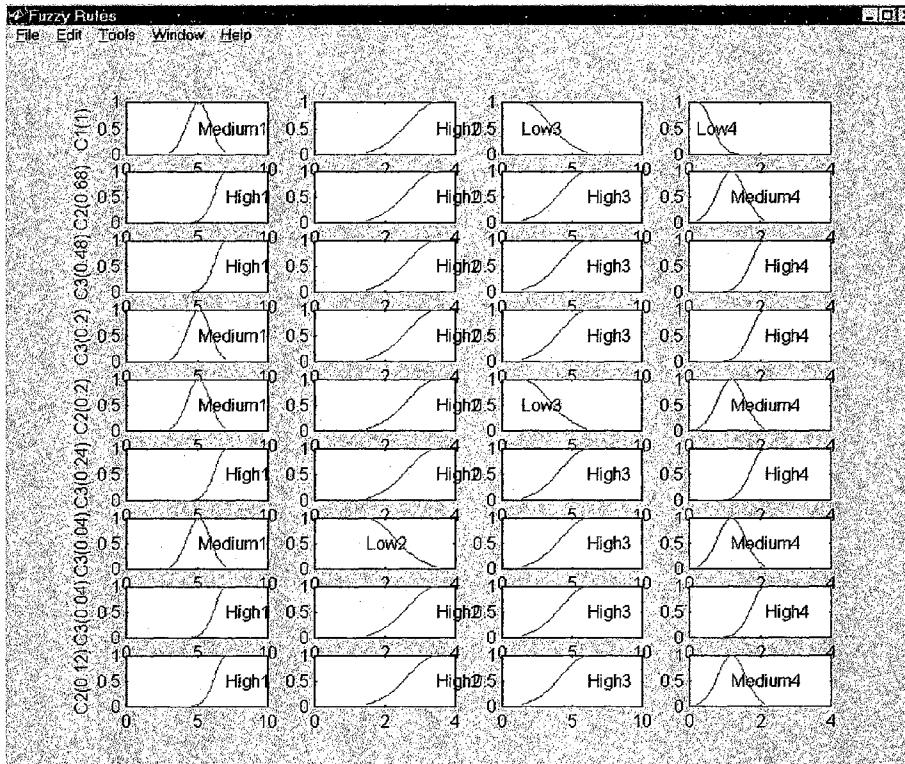


Figure 6.32: A Plot of Initial Fuzzy Rules

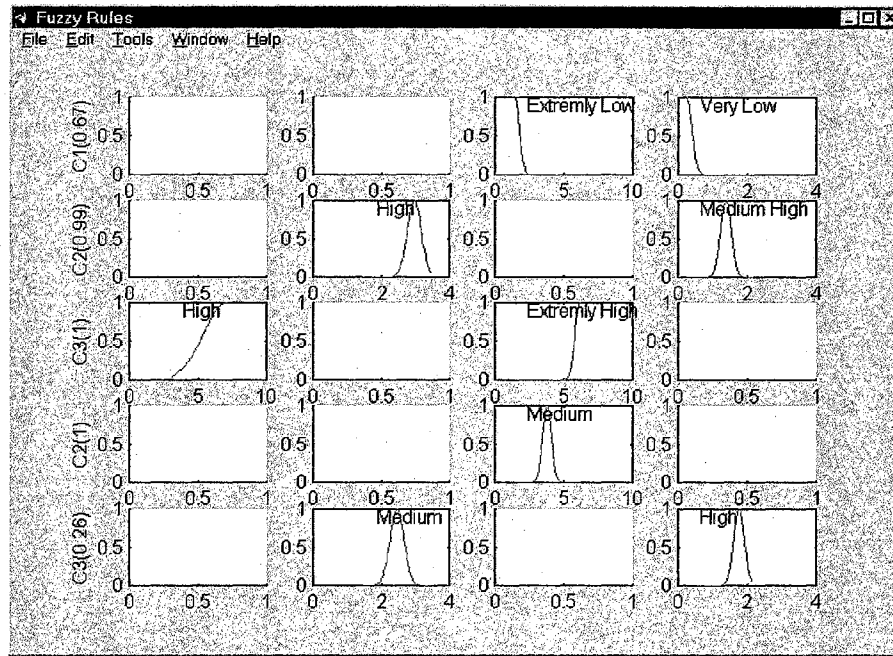


Figure 6.33: A Plot of Optimized Rules

#### 6.14 Visualization

The visualization command menu, as shown in Figure 6.34, provides a way to visualize the internal information of the data and the trained systems. There are four commands in the visualization menu: “Visualize DATA,” “Visualize Clusters of ILFN,” “Visualize Fuzzy Rule Surface,” and “Visualize Hybrid Surface.” A scatter plot of data window, as shown in Figure 6.35, appears when the “Visualize Data” command is selected. The user can plot two-dimensional plot or three-dimensional graph by selecting the X, Y, and Z dimensions. The pattern is plotted, one color for each class. The user can view the characteristic of the internal structure of the data. Figure 6.36 shows the clusters of the ILFN, when the “Visualize Clusters of ILFN” command is selected. This command allows user to see the locations of the ILFN clusters and their shapes. Figure 6.37 illustrates the “Rule Surface Viewer” window that is appeared when the “Visualize Fuzzy

Rule Surface” command is selected. This command allows the user to understand the relationships between two dimensions of fuzzy rules.

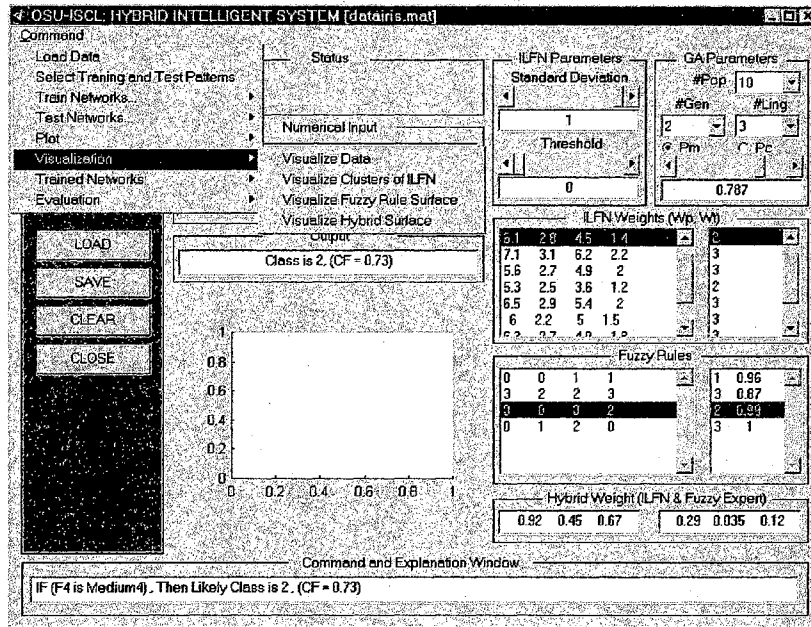


Figure 6.34: Visualization Command Menu

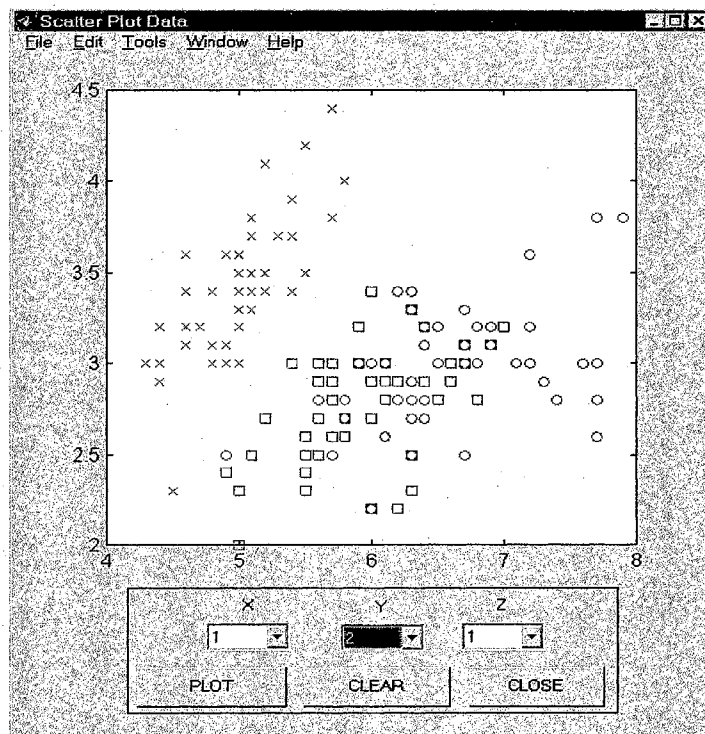


Figure 6.35: Visualize Data

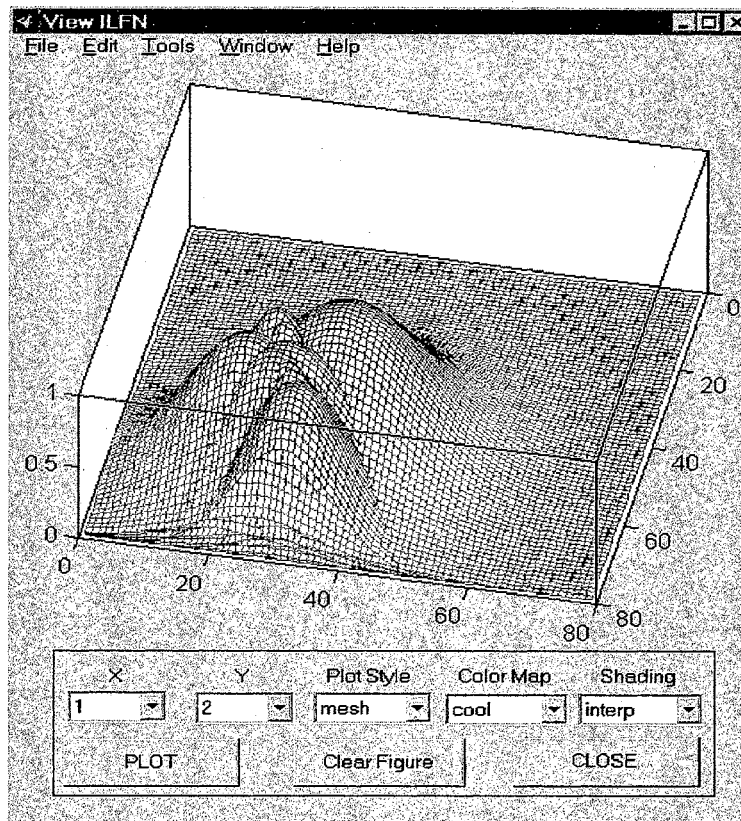


Figure 6.36: Visualize Clusters of ILFN

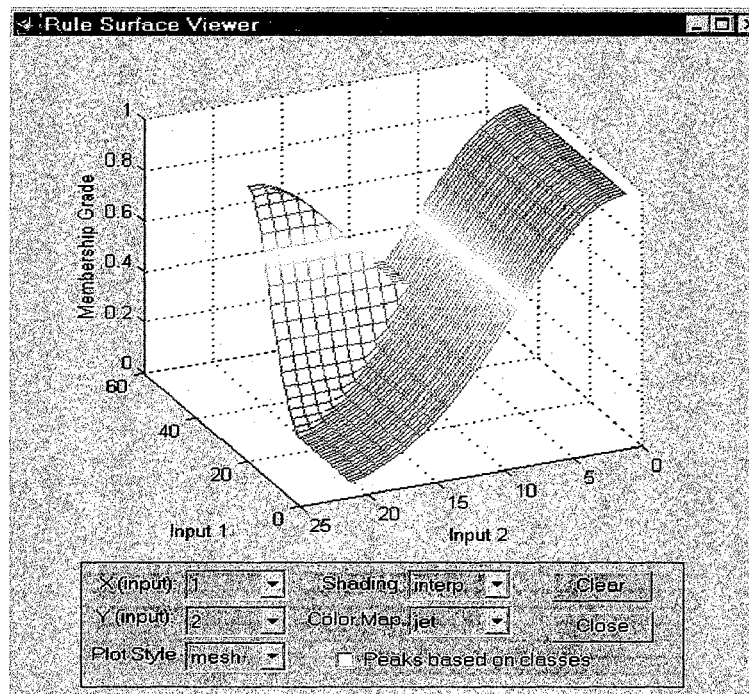


Figure 6.37: Visualize Fuzzy Rule Surface

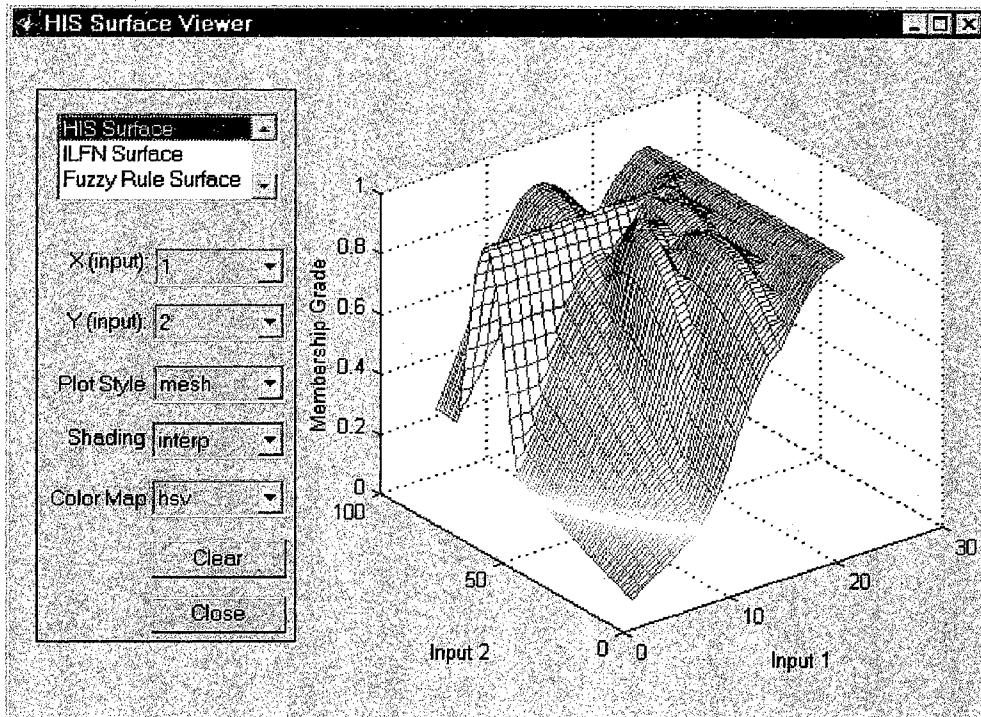


Figure 6.38: Visualize Hybrid Surface

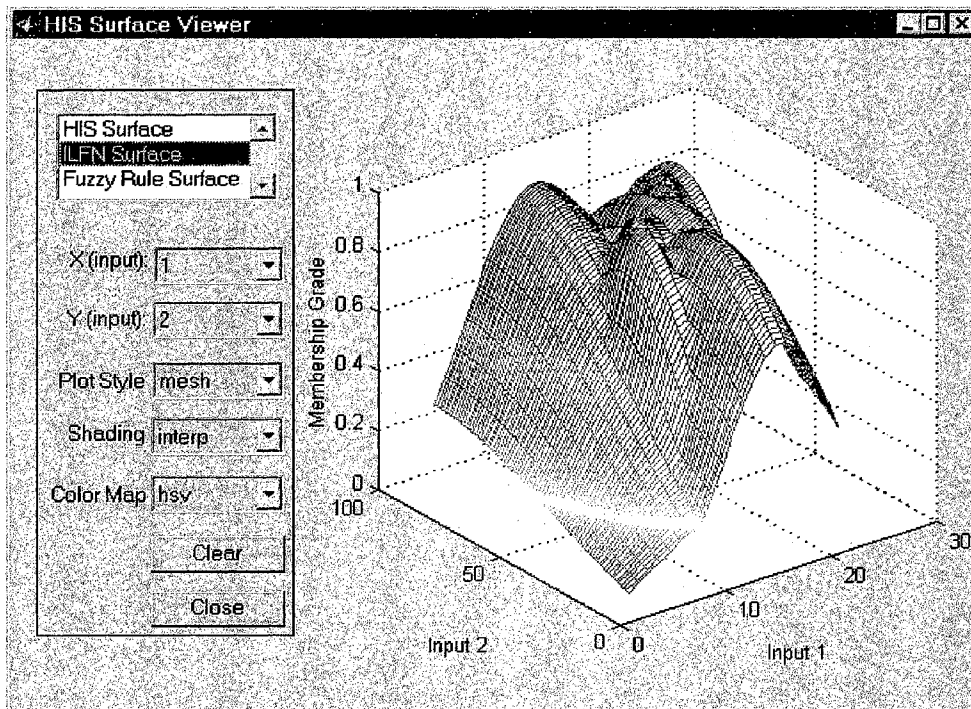


Figure 6.39: View Surface of ILFN

Figure 6.38 shows the “HIS Surface Viewer” window appeared when the “Visualize Hybrid Surface” command is selected. This command helps the user to visualize the structure of the HIS, which is the combination of ILFN clusters and fuzzy rules of the FES. The ILFN and the FES can be individually viewed in the “HIS Surface Viewer” window, as shown in Figures 6.39 and 6.40.

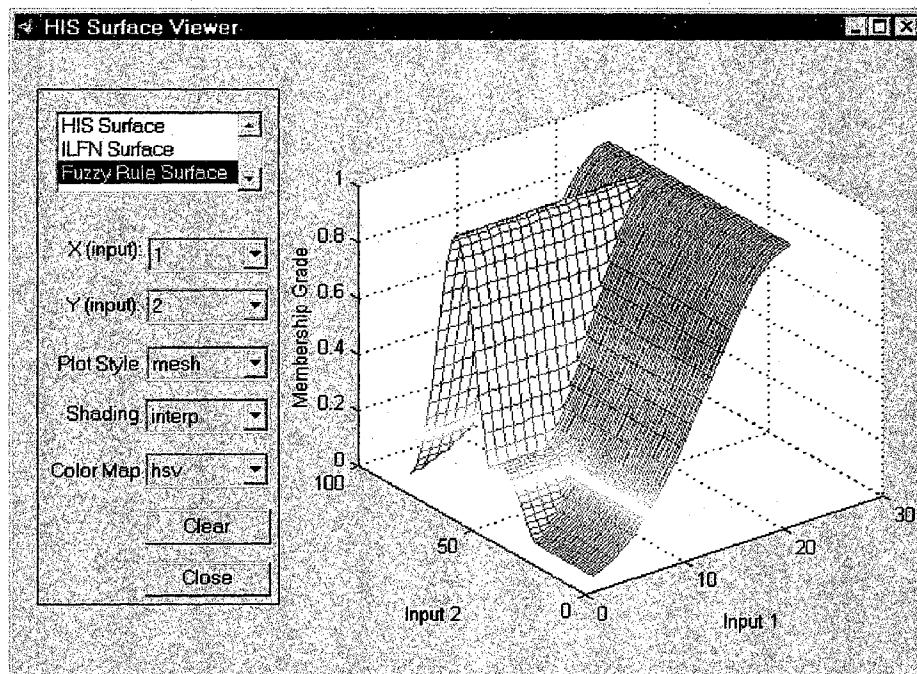


Figure 6.40: View Surface of FES

**CHAPTER VII**  
**ADDITIONAL STUDY: FUZZY TEMPORAL**  
**REPRESENTATION AND REASONING**

Temporal representation and reasoning has received growing attention in the design of computational intelligence systems. Knowledge representation and reasoning about time is a main subject to be considered in all those reasoning tasks which take account of a dynamic environment. Employing temporal reasoning in computational intelligence systems is practical in many application domains such as medical diagnosis [Dutta88], [Kiseliova01], industrial processes [Allouche97], management systems [Chinn00], [Aboelela99], and decision support systems [Chaal01].

Recently, fuzzy logic has been a considerable interest in applying to temporal representation and reasoning. A temporal language based on fuzzy temporal constraints turns out to be having an important effect for domains in which knowledge is imprecise or uncertain. Very beginning works for processing fuzzy temporal knowledge were proposed by Dutta [Dutta88] and Dubois and Prade [Dubois89]. Dutta modeled the lack of knowledge about events by means of fuzzy sets of time intervals. Dubois and Prade [Dubois89] proposed an approach based on possibility theory for the representation and management of imprecision and uncertainty in temporal knowledge.

Some applications of fuzzy temporal representation and reasoning include but do not limited to application in explanation, planning, industrial process supervision, and prediction. In explanation for a given problem, fuzzy temporal can be used to produce a description of the world at some past time which accounts for the world being the way it currently is. In developing a plan one has to consider the duration of the actions and tasks

that can be performed and find out the most appropriate temporal ordering taking into account their interaction over time. For industrial process supervision, to correctly control a process it is necessary to consider the different past states of the process, the historic evolution of the variables, which and when operations have been performed and how the actions that can be carried out would affect the evolution of the process. Fuzzy temporal also can be applied in prediction applications that determine the state of the world at a given future time or, more generally, the evolution of the world until a given future time.

### **7.1 Constructing Fuzzy Temporal Models from Data**

In the same fashion of normal fuzzy models, constructing fuzzy temporal models requires to define membership functions and their parameters in both antecedent parts and consequent parts. The membership functions define the fuzzy subspaces in both input spaces and output spaces. Traditionally, experts have to predefine the membership functions and their parameters. Input and output spaces are divided into fuzzy subspaces and determine the locations of the linguistic variables. The rule evaluation process is taken on to evaluate the goodness of the resulting linguistic model. However, due to the lack of experienced experts or the unavailability of experienced experts in a given problem, linguistic models are not always easy to construct.

Recently, there are many alternative methodologies that can be applied to construct fuzzy models based on the relationship between input and output data. Using data, it is much faster and easier to generate fuzzy if-then rules and to find the parameters for the membership functions. Fuzzy c-mean (FCM) clustering algorithm is a well-known approach to define fuzzy subspaces for fuzzy modeling. After FCM applied to the data, a



particular membership function is used for input variable to compose a rule condition. In this way fuzzy models generated utilize local fuzzy sets pertaining to individual rules rather than global fuzzy sets used by all rules. It is not reasonable to define different linguistic label to every different rule. This leads to resulting fuzzy systems that are not transparent or semantically meaningless. Besides the accuracy, many other researchers have paid more attention to the comprehensibility of the resulting linguistic models. Genetic algorithms (GAs) may be currently the most popular approach for applying to search for accurate and comprehensible systems. Though they are very effective in finding good systems, GAs have been suffered with too expensive for the cost of the learning time. GAs may be not applicable in such a system that requires a quick learning for tracking dynamic environments.

In this study, the ILFN is used to assist in constructing fuzzy temporal system. The ILFN network has been discussed in Section 3.1. The ILFN can be used as a clustering algorithm. Unlike FCM clustering, ILFN does not need to predefine the number of the clusters. ILFN finds the number of the clusters automatically. After the ILFN is applied to learn the data, the location of each cluster is mapped to fuzzy temporal system based on the fuzzy partition calculated from the numerical ranges of the data. A rule extraction algorithm is used to map from ILFN to fuzzy rules, which is detailed in Section 3.3. The details of the proposed fuzzy temporal system and the details of using ILFN and the rule extraction algorithm to construct fuzzy temporal system can be described as follows.

## 7.2 Temporal Representation and Reasoning by Fuzzy Temporal Systems

Usually, dealing with fuzzy temporal reasoning implies that the process is operating in dynamic environments. Fuzzy systems applied in dynamic environments may be called fuzzy temporal systems. The structure of a fuzzy temporal system proposed in this study is shown in Figure 7.1.

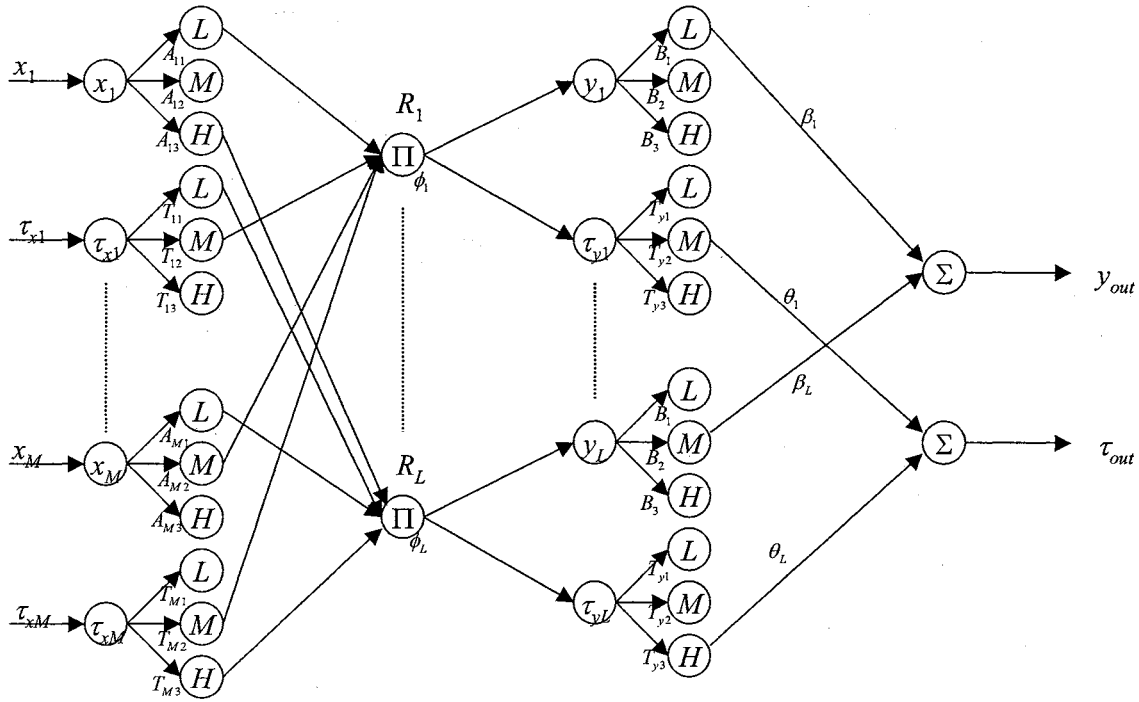


Figure 7.1: A Fuzzy Temporal Model

Like Mamdani fuzzy model, the proposed fuzzy temporal model is a linguistic model that has linguistic variables in both the antecedents and the consequents. The knowledge of a fuzzy temporal system can be represented by a finite set of rules that may contain fuzzy linguistic propositions and *time-dependent* information. The proposed fuzzy temporal model is a generalized form of Mamdani model. The knowledge of the proposed model is in the following form:

$$\begin{aligned}
R_i: \quad & \text{IF } (x_1(t) \text{ is } A_{i1}) \text{ [and } (\tau_{x1} \text{ is } T_{i1})] \text{ and } (x_2(t) \text{ is } A_{i2}) \text{ [and } (\tau_{x2} \text{ is } T_{i2})], \\
& \dots, \text{ and } (x_M(t) \text{ is } A_{iM}) \text{ [and } (\tau_{xM} \text{ is } T_{iM})], \text{ THEN } (y_i(t) \text{ is } B_i) \text{ with} \\
& \text{weight } \beta_i \text{ [and } (\tau_y \text{ is } T_{yi}) \text{ with weight } \theta_i].
\end{aligned} \tag{7.1}$$

$R_i$ ,  $i = 1 \dots, L$ , represents the  $i$ th rule. Each  $x_j(t)$ ,  $j = 1, \dots, M$ , represents an entity attribute that takes values in the input domain  $X_j$  at time  $t$ . Each  $A_{ij}$ ,  $i = 1 \dots, L$ ,  $j = 1, \dots, M$ , is a linguistic term representing a fuzzy subset of  $X_j$ . Each  $\tau_{xj}$  represents a time attribute associated with the input  $x_j$ .  $T_{ij}$ ,  $i = 1 \dots, L$ ,  $j = 1, \dots, M$ , represents linguistic variables of a time gap in the time domain. The squared brackets imply that temporal information may be omitted. (If all the temporal information attributes are omitted, the system becomes a Mamdani-like model.)  $y$  represents an entity attribute that takes values in the output domain  $Y$ .  $B$  is linguistic term representing a fuzzy subset of the output domain  $Y$ .

When the input  $\mathbf{p} = [x_1(t), \tau_{x1}, x_2(t), \tau_{x2}, \dots, x_M(t), \tau_{xM}]^T$  is given, the firing strength of the antecedents of the rule is calculated by

$$\phi_i = \prod_{j=1}^M A_{ij}(x_j(t)) \cdot T_{ij}(\tau_{xj}). \tag{7.2}$$

The outputs,  $y_{out}$  and  $\tau_{out}$ , of the fuzzy temporal model are the weighted sum of the matching degrees contributed from the outputs of every rule. *Center average defuzzifier* can be used to compute the total output of the model as

$$y_{out} = \sum_{i=1}^L v_i \beta_i \tag{7.3}$$

$$\tau_{out} = \sum_{i=1}^L w_i \theta_i \tag{7.4}$$

$$v_i = \frac{\phi_i}{\sum_{i=1}^L \phi_i} \cdot \text{defuzz}(B_i) \tag{7.5}$$

$$w_i = \frac{\phi_i}{\sum_{i=1}^L \phi_i} \cdot \text{defuzz}(T_{yi}), \quad (7.6)$$

where  $\text{defuzz}(B_i)$  and  $\text{defuzz}(T_{yi})$  are defined as the defuzzification of the consequent linguistic variables,  $B_i$  and  $T_{yi}$ , of the  $i$ th rule.

Equations (7.3) and (7.4) can be viewed as a special case of the linear regression model which can be rewritten as

$$\hat{y}_{out} = \sum_{i=1}^L v_i \beta_i + e_y \quad (7.7)$$

$$\hat{\tau}_{out} = \sum_{i=1}^L w_i \theta_i + e_\tau \quad (7.8)$$

where  $v_i$  and  $w_i$  are known as the regressors;  $\beta_i$  and  $\theta_i$  are the coefficient parameters of the regressors.  $e_y$  and  $e_\tau$  are the error signals which assumed to be uncorrelated with the regressors. Given  $N$  input-output pairs  $\{ \mathbf{p}(t), \mathbf{z}(t) \}$ ,  $t = 1, 2, \dots, N$ , where  $\mathbf{p}(t) = [x_1(t), \tau_{x1}, x_2(t), \tau_{x2}, \dots, x_M(t), \tau_{xM}]^T$  and  $\mathbf{z}(t) = [y(t), \tau_y(t)]^T$ , equations (7.7) and (7.8) can be expressed in the matrix forms as

$$\mathbf{y}_{out} = \mathbf{V}\boldsymbol{\beta} + \mathbf{e}_y \quad (7.9)$$

$$\boldsymbol{\tau}_{out} = \mathbf{W}\boldsymbol{\theta} + \mathbf{e}_\tau \quad (7.10)$$

where  $\mathbf{y}_{out} = [y(1), y(2), \dots, y(N)]^T \in \mathbb{R}^N$ ;  $\boldsymbol{\tau}_{out} = [\tau_y(1), \tau_y(2), \dots, \tau_y(N)]^T \in \mathbb{R}^N$ ;  $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_L] \in \mathbb{R}^{N \times L}$  and  $\mathbf{v}_i = [v_i(1), v_i(2), \dots, v_i(N)]^T \in \mathbb{R}^N$ ;  $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_L] \in \mathbb{R}^{N \times L}$  and  $\mathbf{w}_i = [w_i(1), w_i(2), \dots, w_i(N)]^T \in \mathbb{R}^N$ ;  $\boldsymbol{\beta} = [\beta_1, \beta_2, \dots, \beta_L]^T \in \mathbb{R}^L$ ;  $\boldsymbol{\theta} = [\theta_1, \theta_2, \dots, \theta_L]^T \in \mathbb{R}^L$ ;  $\mathbf{e}_y = [e_y(1), e_y(2), \dots, e_y(N)]^T \in \mathbb{R}^N$ ; and  $\mathbf{e}_\tau = [e_\tau(1), e_\tau(2), \dots, e_\tau(N)]^T \in \mathbb{R}^N$ . The matrix  $\mathbf{V}$  and  $\mathbf{W}$  are known in *a priori* from training data. The only unknown parameters

are  $\beta$  and  $\theta$  which can be determined by solving by the least squares estimations. The aim is to reduce the norm of the error vector  $\mathbf{e}_y$  and  $\mathbf{e}_\tau$  to close to zero. That is to minimize

$$\min_{\beta} \|\mathbf{e}_y\|_2 = \min_{\beta} \|\hat{\mathbf{y}}_{out} - \mathbf{V}\beta\|_2 \quad (7.11)$$

$$\min_{\theta} \|\mathbf{e}_\tau\|_2 = \min_{\theta} \|\hat{\boldsymbol{\tau}}_{out} - \mathbf{W}\theta\|_2. \quad (7.12)$$

$\hat{\mathbf{y}}_{out}$  and  $\hat{\boldsymbol{\tau}}_{out}$  are the estimated fact and time outputs. The solutions to equations (7.11) and (7.12) are

$$\beta = (\mathbf{V}^T \mathbf{V})^{-1} \mathbf{V}^T \mathbf{y}_{out} \quad (7.13)$$

and 
$$\theta = (\mathbf{W}^T \mathbf{W})^{-1} \mathbf{W}^T \boldsymbol{\tau}_{out}. \quad (7.14)$$

The solution for  $\beta$  will be valid if and only if  $\text{rank}(\mathbf{V}^T \mathbf{V}) = \text{dim}(\beta)$  and, similarly, the solution for  $\theta$  will be valid if and only if  $\text{rank}(\mathbf{W}^T \mathbf{W}) = \text{dim}(\theta)$ . This implies that all the rules have to receive enough excitation during training, which may be not true in every situation. In practice, the matrixes  $(\mathbf{V}^T \mathbf{V})$  and  $(\mathbf{W}^T \mathbf{W})$  may be singular when the rules have low excitation. The resulting rules fuzzy system will have significant errors.

Using an adaptive strategy, such as the recursive least squares (RLS) [Biermann77], to adapt only the consequence of those rules that has been excited can solve this problem. For the rule without excitation, an initial numerical value is assigned using *a priori* knowledge of experts, if available.

Alternatively, a solution to guarantee that the matrixes will not singular is to add small numbers called *excitation factor* to the matrixes  $(\mathbf{V}^T \mathbf{V})$  and  $(\mathbf{W}^T \mathbf{W})$ . So we have

$$\beta = (\mathbf{V}^T \mathbf{V} + \mathbf{U})^{-1} \mathbf{V}^T \mathbf{y}_{out} \quad (7.15)$$

and 
$$\theta = (\mathbf{W}^T \mathbf{W} + \mathbf{U})^{-1} \mathbf{W}^T \boldsymbol{\tau}_{out} \quad (7.16)$$

$$\mathbf{U} = \begin{bmatrix} \lambda & 0 & \dots & 0 \\ 0 & \lambda & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda \end{bmatrix} \quad (7.17)$$

where  $\mathbf{U}$  is a diagonal matrix composed of small numbers,  $\lambda$ , in the diagonal and zeros elsewhere and size of  $\mathbf{U}$  equal size of  $(\mathbf{V}^T\mathbf{V})$  and  $(\mathbf{W}^T\mathbf{W})$ .

### 7.3 An Algorithm for Generating Fuzzy Temporal Systems

It is aimed to design an algorithm to obtain a good compromise between numerical approximation accuracy, linguistic comprehensibility, and the completeness of the fuzzy system. This tradeoff has been of interest in developing fuzzy systems. The main step for generating fuzzy temporal systems are as follows:

**Step 1:** *Data preparing:* Collect  $N$  points of data from the input-output pairs

$\{\mathbf{p}(t), \mathbf{z}(t)\}$ ,  $t = 1, 2, \dots, N$ , where  $\mathbf{p}(t) = [x_1(t), \tau_{x1}, x_2(t), \tau_{x2}, \dots, x_M(t),$

$\tau_{xM}]^T \in \mathfrak{R}^{2M}$  and  $\mathbf{z}(t) = [y(t), \tau_y(t)]^T \in \mathfrak{R}^2$ . Use  $\mathbf{p}(t)$  and  $\mathbf{z}(t)$  to reformat

data  $\mathbf{x}(t) = [\mathbf{p}(t)^T, \mathbf{z}(t)^T]^T \in \mathfrak{R}^{2M+2}$ .

**Step 2:** *Data clustering:* Use the new formatted data  $\mathbf{x}(t)$  to train to ILFN in an

unsupervised mode, i.e., there are no target vectors for the ILFN. The

ILFN will learn the data and find the cluster of the formatted data. The

ILFN constructs a cluster matrix  $\mathbf{W}_P$  and a target matrix  $\mathbf{W}_T$ . (Please

note here that  $\mathbf{W}_T$  keeps only the cluster target generated by the ILFN. It

is not the real target or real output, of the input data. Since the real target

outputs have been incorporated in the reformatted data.)

**Step 3:** *Mapping ILFN to Rule:* The *ilfn2rule* algorithm is used to map ILFN to rule. The user needs to pre-specify the type of membership functions and the number of linguistic variable in each dimension of the formatted data. Using the *ilfn2rule* algorithm and a trained ILFN as its input argument, the *ilfn2rule* algorithm finds the variables ranges for each dimension of the formatted data based on the trained ILFN. Then the *ilfn2rule* algorithm finds parameters of the linguistic labels in all dimensions. Finally, the cluster matrix  $\mathbf{W}_P$  is mapped to linguistic rules.

**Step 4:** *Reorganizing antecedents and consequents of the rules for step 3:* Please note that the *ilfn2rule* algorithm is mapped  $\mathbf{W}_P$  to the antecedents of the rules and  $\mathbf{W}_T$  to the consequents of the rules. Nevertheless, the resulting fuzzy rules are not correct yet. Since we include both input and output into the same vector to reformat the data, the resulting rules will contain the antecedents that have both input and output features; while the consequents contain the target label generated by the ILFN. To obtain correct antecedents and consequents of the rules, we recompose the new antecedent by using the first  $2M$  columns of the original antecedents. The new consequents are obtained from the next 2 columns of the original antecedents.

**Step 5:** *(Optionally) Finding the consequent weights of the fuzzy temporal system:* After the antecedents and the consequents are obtained, it is read to determined the additional numerical weights  $\beta$  and  $\theta$ . Apply Equations (7.13) and (7.14) to find  $\beta$  and  $\theta$ .

**Step 6: Giving final rules:** The final rules are composed of antecedents from step 4, while the consequents are the consequents from step 4 concatenating with the additional numerical weights  $\beta$  and  $\theta$  obtained from step 5.

## 7.4 Simulation Study

Simulation study was performed to prove the concept of the proposed method. The simulation study was based on a process of hot-and-cold water mixing simulator. The aim was to use the proposed rule generation method to derive the cause-and-effect relationships induced by the process. Knowing the cause and effect relationships in the process, it is easy to manage the process based on the linguistic rule generated.

### 7.4.1 A Process of Hot-And-Cold Water Mixing Simulator

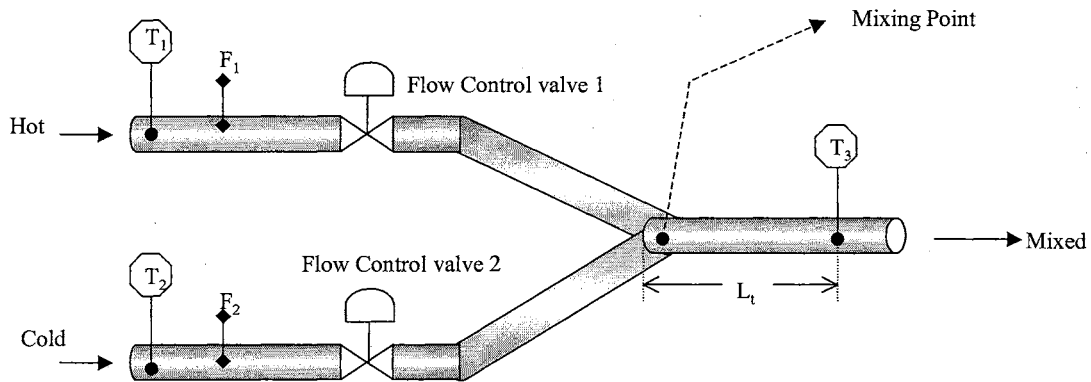


Figure 7.2: A Process of Hot-And-Cold Water Mixing Simulator

The process of hot-and-cold water mixing simulator was used as an example for reasoning the cause-and-effect relationships. The process comprises of two input water pipes and an output water pipe. The water output in pipe 3 is the mixing water between pipe 1 and pipe 2. The temperature and flow rate of the two water inputs can be adjusted.



Adjusting temperature and flow rate of pipes 1 and 2 will cause the water output at pipe 3 changes in temperature as well as flow rate. Figure 7.2 shows the diagram of the process of hot-and-cold water mixing simulator.

In this data simulation, the temperature of pipe 2 was kept constant. The temperature of pipe 1 ( $T_1$ ), flow rate 1 ( $F_1$ ), and flow rate 2 ( $F_2$ ) were changed. There are 1,079 data points with 7 dimensions. All the dimensions of the data are used in clustering. Dimension 1 to 3 (Inlet Temp1, Flow rate 1, and Flow rate 2) are used as clauses. Dimension 4 to 6 are the effects (Temp3 after small delay, Temp3 after medium delay, and Temp after large delay). Dimension 7 (Total Delay) is used as auxiliary information to select one out of the three outputs. A portion of the data is shown in Table 7.1 and the plot of the data is shown in Figure 7.3.

TABLE 7.1:  
Simulated Data from Hot-And-Cold Water Mixing Process

Inlet Temp 1	Flow rate 1	Flow rate 2	Output TempT3 after small delay	Output TempT3 after medium delay	Output TempT3 after large delay	Total Delay
4.758129	8.803611	4.251697	34.55271	49.13313	49.03167	9.820508
9.063462	16.93759	10.24756	34.48225	49.56707	48.77365	6.552082
12.95909	19.77439	12.32197	34.44295	49.65956	48.39556	6.094354
16.484	20.45669	12.94478	34.42908	49.67792	47.77696	5.995535
19.67347	20.56671	13.09719	34.42444	49.68147	46.89688	5.976493
22.55942	20.5805	13.12065	34.4226	49.68207	45.77946	5.973815
25.17074	20.58112	13.12515	34.42171	49.68207	44.37085	5.973446
27.53356	20.58025	13.12661	34.42116	49.68207	42.93735	5.973404
29.67152	20.57949	13.12737	34.42078	49.68207	41.78879	5.973404
31.60603	20.57893	13.12787	34.4204	49.68207	41.00483	5.973408
33.35644	20.57855	13.1282	34.42021	49.68207	40.51879	5.973411
34.94028	20.5783	13.12844	34.42014	49.68207	40.23433	5.973412
36.3734	20.57811	13.1286	35.35246	49.67899	40.07803	5.973415
37.67014	20.57793	13.12872	38.95541	49.63313	40.09012	5.973419
38.84346	20.57786	13.12881	43.49283	49.48849	40.57659	5.973418
40.00416	20.57783	13.12884	47.4348	49.26642	41.83931	5.973417
41.72237	20.57783	13.12884	49.13313	49.03167	43.98223	5.973417
44.13358	20.57783	13.12884	49.56707	48.77365	47.17145	5.973417
46.88627	20.57783	13.12884	49.65956	48.39556	50.68148	5.973417
49.56163	20.57783	13.12884	49.67792	47.77696	53.41776	5.973417
51.98241	20.57783	13.12884	49.68147	46.89688	55.13325	5.973417
54.17282	20.57783	13.12884	49.68207	45.77946	56.09249	5.973417
56.15478	20.57783	13.12884	49.68207	44.37085	56.60021	5.973417
57.94814	20.57783	13.12884	49.68207	42.93735	56.86259	5.973417
59.57083	20.57783	13.12884	49.68207	41.78879	56.99699	5.973417

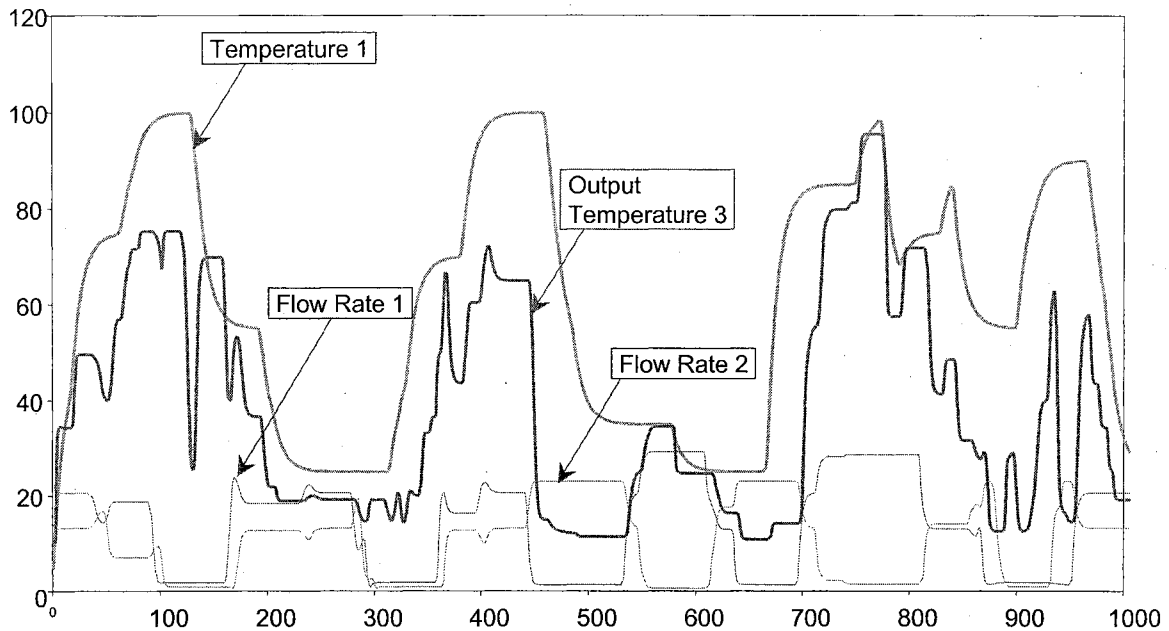


Figure 7.3: The Plot of Simulated Data from Hot-And-Cold Water Mixing Process

#### 7.4.2 Rule Generation

To generate linguistic rules from data, the algorithm discussed in Section 7.3 is applied. The data in this simulation does not have a temporal feature in the three-input clauses while the output effects have temporal feature namely “Time Delay.” The procedures of the rules generation are as follows.

- 1) **Data Preparing:** Reformatted input data by input-output data i.e., all 7 dimensions: Temp1, Flow Rate2, Flow Rate, Temp3 short time, Temp after medium delay, Temp3 after long delay, and Time Delay.
- 2) **Data Clustering:** Apply the data to ILFN algorithm. After training, ILFN gives  $W_P$ , which contains the clusters of the data, and  $W_T$ , which contains the regenerated target labels.

- 3) **Mapping ILFN to Rule:** Call the *ilfn2rule* algorithm to calculate the linguistic variables based on the ranges of the numerical values in each dimension of the data. The clusters of the data found in 2) are then mapped to linguistic variables calculated. In mapping, each center in each dimension of the clusters is mapped to the linguistic labels. Now each cluster corresponds to a rule that has the linguistic variables of *Temp1*, *Flow Rate1*, *Flow Rate2*, *Temp3 after short time*, *Temp3 after medium time*, *Temp3 after long delay*, and *Time Delay* for the antecedents. The consequents are the target values mapped from  $W_T$ .
- 4) **Reorganizing the rules:** Rearrange the rule such that each cluster corresponds to a rule that has the linguistic variables of *Temp1*, *Flow Rate1*, and *Flow Rate2* for the antecedents and linguistic variables of *Temp3 after short time*, *Temp3 after medium time*, and *Temp3 after long delay* for the consequents. The linguistic variables of the *Time Delay* attribute are kept separately to help in selecting the correct temperature output. In each rule, the correct output is associated with the time delay auxiliary linguistic variables. For example, if a rule has an auxiliary variable as *short*, then the correct output is *Temp3 after short delay*. So we keep only *Temp3 after short delay* for the consequent of the rule.
- 5) **Giving final rules:** The final rules are composed of antecedents and the consequents obtained from step 4.

### 7.4.3 The Resulting Linguistic Rules

The resulting number of rules seems to be small compared to the number of the available data. The number of the resulting rules is 17 rules generated out of 1,079 data points. It is found that the generated rules are easy to understand for domain expert. Among all 17 rules, four rules listed below, are identified to be the most straightforward knowledge for anyone with fluid dynamic training. The linguistic rules about the causes and effects of the process are expressed as follows.

1. IF input (Temp1 is *Medium*) and (Flow Rate1 is *Low*) and (Flow Rate2 is *Low*), Then output (*after long delay* Temp3 is *Medium*)
2. IF input (Temp1 is *High*) and (Flow Rate1 is *Low*) and (Flow Rate2 is *High*), Then output (*after short delay* Temp3 is *Low*)
3. IF input (Temp1 is *Medium*) and (Flow Rate1 is *Low*) and (Flow Rate2 is *High*), Then output (*after short delay* Temp3 is *Low*)
4. IF input (Temp1 is *Medium*) and (Flow Rate1 is *High*) and (Flow Rate2 is *Low*), Then output (*after short delay* Temp3 is *Medium*)
5. IF input (Temp1 is *Low*) and (Flow Rate1 is *High*) and (Flow Rate2 is *Low*), Then output (*after short delay* Temp3 is *Low*)

Figures 7.4, 7.5, and 7.6 show the linguistic variable of the input and output. From the figures, it is easily notice that the linguistic labels are very easy to comprehend, since the partitions are complete, each linguistic term is easily distinguishable, and the number of linguistic term is small. Only three linguistic terms are used in each dimension.

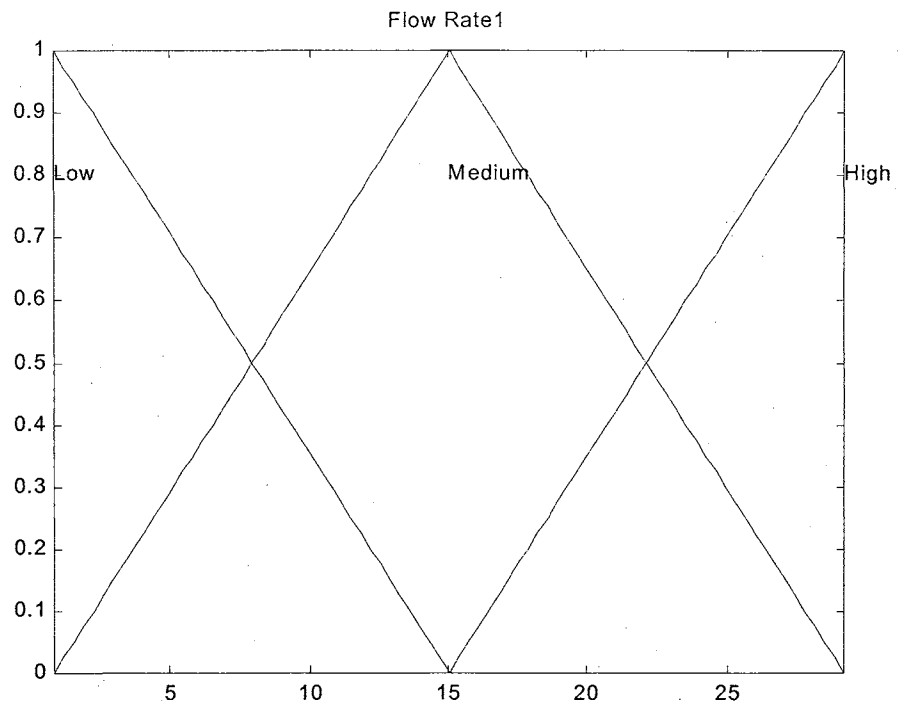


Figure 7.4: Linguistic Variables for Flow Rate 1 and Flow Rate 2

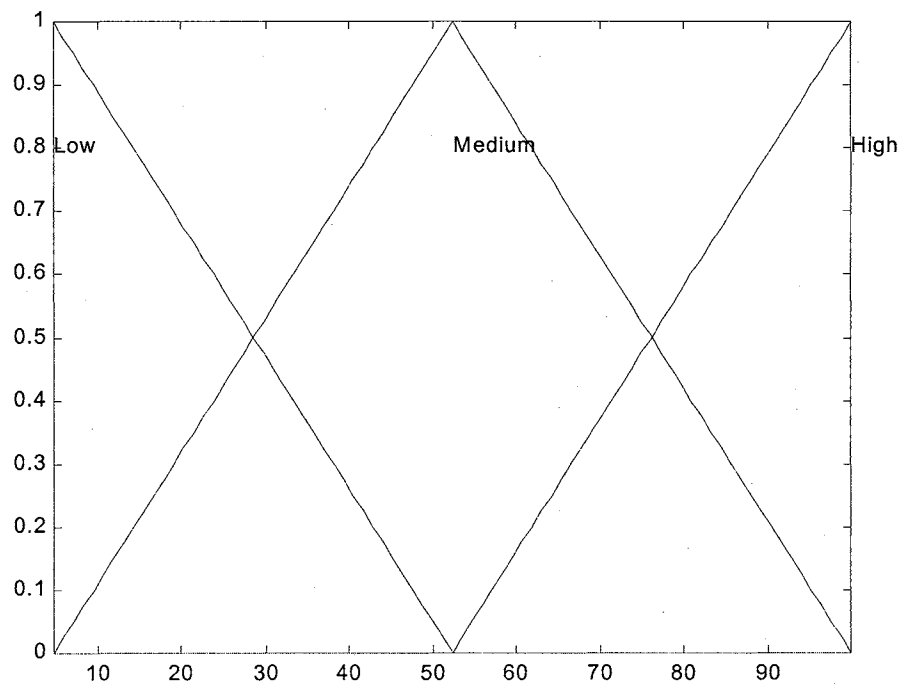


Figure 7.5: Linguistic Variables for Inlet Temp1 and Outlet Temp 3

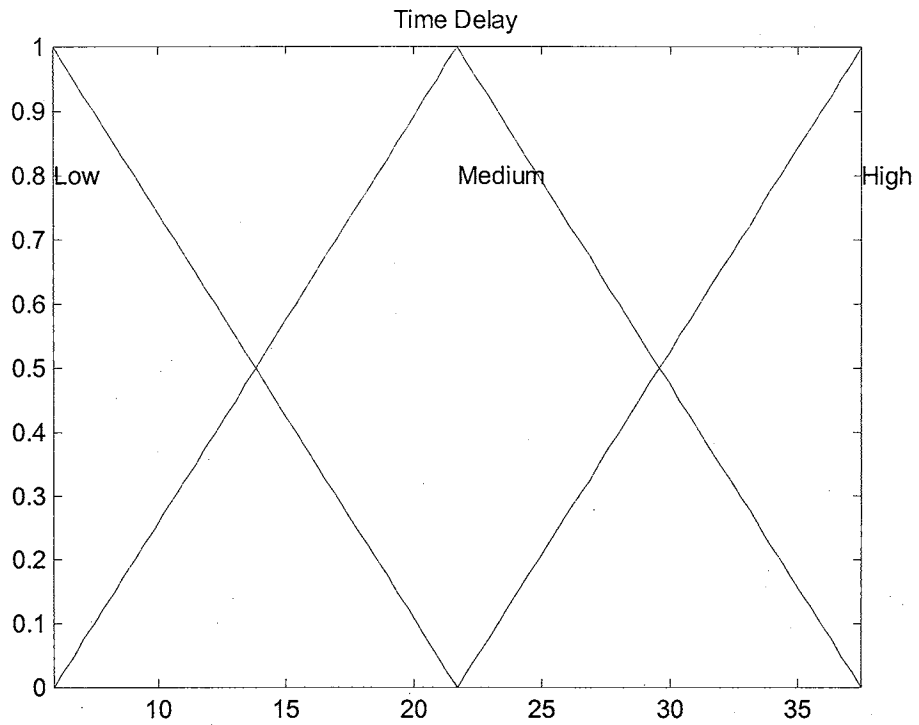


Figure 7.6: Linguistic Variables of Time Delay

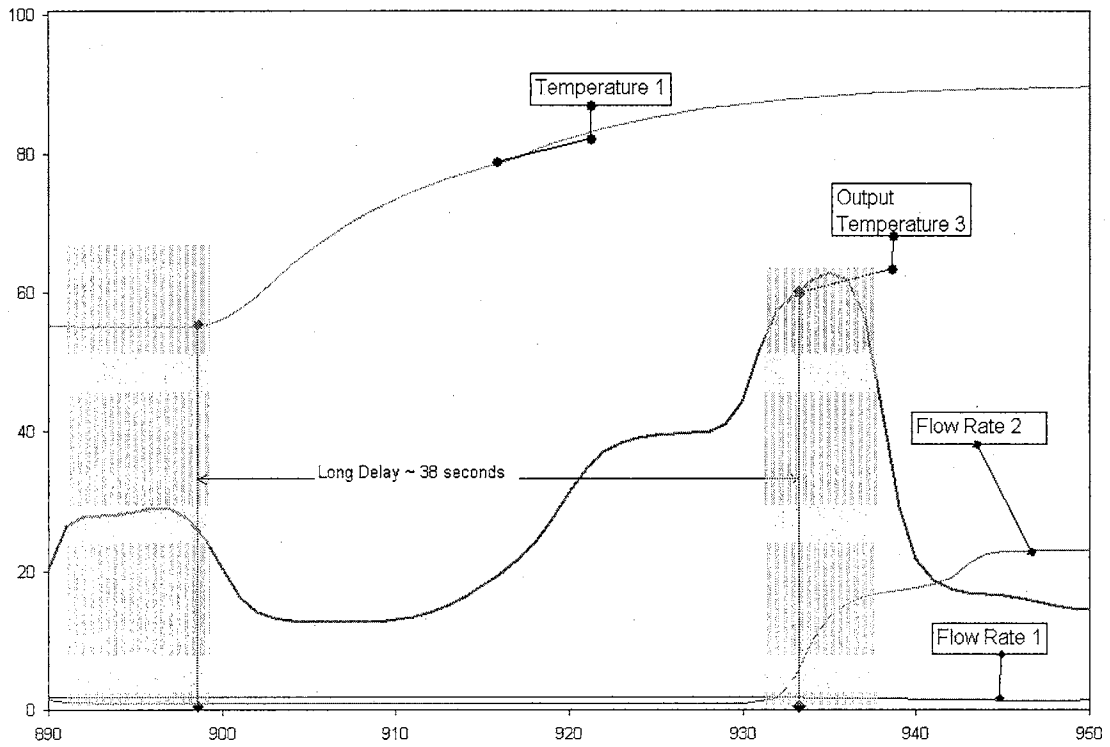


Figure 7.7: Sample Temporal Representation and Reasoning

The rule generation method proposed herein has been successfully applied to search for the cause-and-effect relationships for a hot-and-cold mixing water simulator. The number of linguistic rules generated by the proposed method was small number compared to the large number of the available training data points. The resulting linguistic rules were reasonable and acceptable for the expert who is knowledgeable about the process. For example, Rule 1, “IF input (Temp1 is *Medium*) and (Flow Rate1 is *Low*) and (Flow Rate2 is *Low*), Then output (*after long delay* Temp3 is *Medium*),” can be matched with the raw data as in Figure 7.7. In Figure 7.7, it is illustrated that Rule 1 agrees very well with the data that is when Temp1 is *medium* (about 40-60), Flow Rate1 and Flow Rate2 are *low* (about 0-5), the output Temp2 at *long delay* (about 38 seconds) will be *medium* (about 40-60).

This project was a first step of developing process cause-and-effect relationships for process management and automation. For future research, the algorithm should be applied to a more complex process. Temporal features of the process should be incorporated in the training data to take advantages of the rich of information in the dynamic environments. In addition, GA should be incorporated to minimize Type I error (the anticipation of an event when it does not happen), Type II error (no anticipation an event when it does happen), and rule complexity (the number of conjunctions included in rules). Moreover, rule evaluation method should be incorporated to check the goodness of the resulting fuzzy rules.

## CHAPTER VIII

### CONCLUSIONS AND FUTURE RESEARCH

Researchers have paid great attention to computational intelligence techniques such as artificial neural networks, fuzzy systems, genetic algorithms, and hybrid combinations of these methods i.e., *hybrid intelligent systems* (HIS) for years. The computational intelligence methods proposed by the researchers were derived from intelligent behaviors of human beings or other forms of intelligence in nature. According to previous studies, many of these methods were proved to be suitable for real world applications. Often, an intelligent system needs to possess a multitude of intelligent learning methodologies in order to achieve a better solution for a complex problem. For example, a problem might need a high degree of accuracy as well as its ability to be interpreted or reasoned by human. Therefore, the system may be required to incorporate two intelligence frameworks seamlessly together: an artificial neural network (for dealing with low level numerical data) and a fuzzy expert system with linguistic knowledge representation methods (for reasoning the decision). The need to develop combination of several intelligent approaches to form a unique intelligent system led to the studies documented in this dissertation.

#### 8.1 Concluding Remarks

The main objective of this dissertation was to propose an intelligent system that could mimic some of the intelligent behaviors of human beings -- the ability to learn and reason a problem. In this dissertation work, the focus was on the combination of two intelligent systems, namely “an Incremental Learning Fuzzy Neural Network (ILFN)”



and “Fuzzy Expert System (FES).” The combination of the trained ILFN network and the FES into a unified structure resulted in a “Hybrid Intelligent System” (HIS) useful for decision-making applications. This system can be useful for complex real-world applications, in particular, for medical diagnosis in which various processing strategies are required to support the decision-making.

The proposed HIS offers mutually complementary advantages inherent in an ILFN network (a low-level numerical representation) and a FES (a higher-level linguistic representation). In the proposed HIS, a mapping mechanism from high-level linguistic knowledge to a low level ILFN network and a fuzzy rules extraction from the ILFN network are incorporated. The higher-level FES allows domain experts to add or revise linguistic rules into the system. New knowledge incorporated from domain experts can be mapped back to the ILFN structure allowing the ILFN network to update its parameters.

In addition, the low-level ILFN can be trained from data in an incremental fashion. The linguistic knowledge of the FES can also be extracted from the trained ILFN. The mapping mechanisms of information from FES to ILFN, i.e., “rule2ilfn” algorithm, and ILFN to FES, i.e., “ilfn2rule” algorithm, are purposed to continuously maintain consistency between low-level and higher-level modules. The outputs of the ILFN and the FES are connected to the decision-explanation module which draws conclusions and provides explanations based on the information received from both the ILFN and the FES.

For validation proposes, computer simulations using the well-known Fisher iris data set and the Wisconsin breast cancer database as well as several real medical data sets were performed. In the simulation, first, we used an ILFN to learn the data. Next, the

linguistic rules in the FES were extracted directly from the trained ILFN network by using the `ilfn2rule` algorithm. After using the `ilfn2rule` algorithm to map the ILFN numerical parameters to linguistic labels, a genetic algorithm was then used to optimize the rule set. Based on the initial rules extracted, the number of rules and features of the FES were refined by using a genetic algorithm. A compact FES with only essential discriminatory features was obtained. Finally, the trained ILFN and the optimized FES were combined into a HIS.

The resulting knowledge from the proposed rule extraction procedure was represented in “if-then” linguistic form that was easily comprehensible. By integrating the ILFN and the FES, explanations and answers can be easily generated when needed while numerical accuracy is preserved. The results showed that the proposed HIS achieved acceptable classification results on both training and testing patterns. The low-level ILFN had a small number of hidden nodes while the higher-level linguistic model extracted had a small number of rules. The trained ILFN and the fuzzy linguistic rules were combined into a HIS, which yielded very good results based on the performance classification as compared to the original system as well as other rule-based methods.

Some quantitative measures pertaining to performance accuracy, comprehensibility, and completeness of fuzzy expert systems were also proposed herein. Quantitative measures were used as the fitness function to guide a genetic algorithm (GA) to search for an optimal fuzzy rule set. In the simulations on three medical domains: breast cancer data, lymphography data, and primary tumor data, the resulting fuzzy rule-based systems were able to competitively yield accurate performance to some state-of-the-art methods in literature. The resulting fuzzy knowledge bases showed a high degree

of comprehensibility. Quantitative measures on accuracy, comprehensibility, and completeness were developed to particularly evaluate the proposed FES, which has linguistic antecedents and constant numbers in the consequents. However, with minor modifications, the evaluation method could be adapted to other types of fuzzy systems as well.

## **8.2 Suggestions for Future Research**

There are many remaining possibilities to be pursued for future research such as 1) improving overall accuracy of the system, 2) incorporating a natural language processing, 3) studying the scalability issue, 4) providing theoretical proofs on the convergence and the robustness of the system, 5) incorporating descriptive and temporal features, and 6) applying the system in real world problems.

First of all, since accuracy is always a concern in every learning system, the overall accuracy of the HIS may be improved by using different methods to combine the decisions from the ILFN and the FES. The current implementation of HIS used a weighted average approach to combine the decisions. Other possible combination methods may include Bayesian framework, overall performance measures, and area of expertise [Taha97]. The accuracy of the HIS may also be improved by preprocessing data before entering into the system. The possible data preprocessing methods include reordering the sequence of the data, filling-in missing features, and applying data transformation and feature extraction.

In the current implementation of the proposed HIS, users have to enter the input data in the form of either numerical values or indexes of linguistic terms. It may be more

desirable if natural language processing (NLP) is incorporated into the HIS framework, so that users will be able to interact with the system using natural language sentences. A decision support system should be able to translate human language into a form that can communicate with the host computer.

Scalability is another important issue when developing a learning system. The scalability of a learning system is affected by the size of the dimension of data, the number of data points, and the complexity of time. A good learning system should be able to learn from data with any size of dimensions as well as with any number of the available data without difficulties or complications. The time complexity is also an important aspect in the scalability issue. When learning a large-scale data set, i.e., large dimensions and large number of patterns in training data set, a good learning system should be able to learn the data in a reasonable period of time. A learning system that cannot handle large-scale data is usually not acceptable to apply in real world situations. In order to apply the proposed method in real world application, the scalability issues need to be carefully evaluated.

It is also very important that a learning system can be justified by the rigor of mathematical analysis. Without mathematical proofs, a system may not be reliable to be applied in some real world applications. Since the learning system developed here is model free, mathematical proofs may be difficult, if not impossible. In the current study, the validation of the learning system was accomplished only by using test data, and mathematical proofs are left for possible future work. The main issue needed to be proved is the convergence of the learning process as well as the robustness of the system.

Another concern is about the temporal representation and reasoning. Temporal representation and reasoning have received increasing attention when designing a computational intelligence system. The temporal feature is a concept that is demanding in many problem domains. The knowledge representation and reasoning about time is one main subject to be considered in all the reasoning tasks that consider the dynamic environment; therefore, incorporating the temporal representation and reasoning into a HIS may be practical in many application domains.

Lastly, the proposed HIS needs to be further investigated under real world applications. The HIS was developed specifically for pattern classification domains and was tested to work well with real medical data sets. Nevertheless, the developed system, HIS, may also be useful in other application domains such as function approximation, control system, and signal processing and warrants further investigation in these areas.

## BIBLIOGRAPHY

- [Abe95] S. Abe and M. S. Lan, "A Method for Fuzzy Rules Extraction Directly from Numerical Data and Its Application to Pattern Classification," *IEEE Transactions on Fuzzy Systems*, Vol. 3, No. 1, pp. 18-28, 1995.
- [Aboelela99] E. Aboelela and C. Douligieris, "Fuzzy Temporal Reasoning Model for Event Correlation in Network Management," in *Proceedings of the Conference on Local Computer Networks (LCN '99)*, 1999, pp. 150-159.
- [Adlassnig01] K. P. Adlassnig, "The Section on Medical Expert and Knowledge-Based Systems at the Department of Medical Computer Sciences of the University of Vienna Medical School," *Artificial Intelligence in Medicine*, Vol. 21, No. 1-3, pp. 139-146, 2001.
- [Allouche97] M. K. Allouche, C. Sayettat, and O. Boissier, "Towards a Multi-agent System for the Supervision of Dynamic Systems," in *Proceedings of the Third International Symposium on Autonomous Decentralized Systems (ISADS 97)*, 1997, pp. 9-16.
- [Andrews72] H. C. Andrews, *Mathematical Techniques in Pattern Recognition*. New York, NY: John Wiley & Sons, 1972.
- [Andrews95] R. Andrews, J. Diederich, and A. B. Tickle, "A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks," Neurocomputing Research Centre, Queensland University of Technology, 1995.
- [Angeline94] P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An Evolutionary Algorithm That Constructs Recurrent Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 5, No. 1, pp. 54-65, 1994.
- [Battiti92] R. Battiti, "First- and Second-Order Methodes for Learning: Between Steepest Descent and Newton's Method," *Neural Computation*, Vol. 4, No. 2, pp. 141-166, 1992.
- [Belacel01] N. Belacel, Ph. Vincke, J. M. Scheiff, and M. R. Boulassel, "Acute Leukemia Diagnosis Aid Using Multicriteria Fuzzy Assignment Methodology," *Computer Methods and Programs in Biomedicine*, Vol. 64, No. 2, pp. 145-151, 2001.
- [Bensaoula98] A. Bensaoula, H. A. Malki, and A. M. Kwari, "The Use of Multilayer Neural Networks in Material Synthesis," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 11, No. 3, pp. 421-431, 1998.
- [Bezdek81] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York, NY: Plenum, 1981.
- [Bezdek86] J. C. Bezdek, S. K. Chuah, and D. Leep, "Generalized K-Nearest Neighbor Rules," *Fuzzy Sets and Systems*, Vol. 18, No. 3, pp. 237-256, 1986.
- [Biermann77] G. Biermann, *Factorization Methods for Discrete Sequential Estimation*. New York: Academic, 1977.
- [Blake98] C. L. Blake and C. J. Merz, *UCI Repository of Machine Learning Databases* [<http://www.ics.uci.edu/~mllearn/MLRepository.html>], Irvine, CA: University of California, 1998.
- [Bors99] A. G. Bors and I. Pitas, "Object Classification in 3-D Images Using Alpha-Trimmed Mean Radial Basis Function Network," *IEEE Transactions on Image Processing*, Vol. 8, No. 12, pp. 1744-1756, 1999.

- [Broomhead88] D. S. Broomhead and D. Lowe, "Multivariable Function Interpolation and Adaptive Networks," *Complex Systems*, Vol. 2, pp. 321-355, 1988.
- [Buchana84] B. G. Buchana and E. H. Shortcliffe, *Rule-Based Expert Systems: The MYCIN Experiment of the Stanford Heuristic Programming Project*. Reading, MA: Addison-Wesley, 1984.
- [Carpenter87] G. A. Carpenter and S. Grossberg, "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine", *Computer Vision, Graphics and Image Processing*, Vol. 37, pp.54-115, 1987.
- [Carpenter91] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System," *Neural Networks*, Vol. 4, pp. 759-771, 1991.
- [Carpenter92] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps," *IEEE Transactions on Neural Networks*, Vol. 3, No. 5, pp. 698-713, 1992.
- [Chaal01] M. Schaal and H. J. Lenz, "Best Time and Content for Delay Notification," in *Proceedings of the Eighth International Symposium on Temporal Representation and Reasoning (TIME 2001)*, 2001, pp. 75-80.
- [Chang77] R. L. Chang and T. Pavlidis, "Fuzzy Decision Tree Algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 7, No. 1, pp. 28-35, 1977.
- [Charalambous92] C. Charalambous, "Conjugate Gradient Algorithm for Efficient Training of Artificial Neural Networks," *IEE Proceedings G*, Vol. 139, No. 3, pp. 301-310, 1992.
- [Chaturvedi92] A. R. Chaturvedi and D. L. Nazareth, "Investigating the Effectiveness of Conditional Classification: An Application to Manufacturing Scheduling," *IEEE Transactions on Engineering Management*, Vol. 41, No. 2, pp. 183-193, 1992.
- [Chen98] E. L. Chen, P. C. Chung, C. L. Chen, H. M. Tsai, and C. I. Chang, "An Automatic Diagnostic System for CT Liver Image Classification," *IEEE Transactions on Biomedical Engineering*, Vol. 45, No. 6, pp. 783-794, 1998.
- [Chen99] S. Chen, Y. Wu, and B. L. Luk, "Combined Genetic Algorithm Optimization and Regularized Orthogonal Least Squares Learning for Radial Basis Function Networks," *IEEE Transactions on Neural Networks*, Vol. 10, No. 5, pp. 1239-1243, 1999.
- [Chen00] J. D. Z. Chen, Z. Lin, and R. W. McCallum, "Noninvasive Feature-Based Detection of Delayed Gastric Emptying in Humans Using Neural Networks," *IEEE Transactions on Biomedical Engineering*, Vol. 47, No. 3, pp. 409-412, 2000.
- [Chinn00] S. J. Chinn and G. R. Madey, "Temporal Representation and Reasoning for Workflow in Engineering Design Change Review," *IEEE Transactions on Engineering Management*, Vol. 47, No. 4, pp. 485-492, 2000.
- [Chow99] M. Y. Chow, S. Altug, and H. J. Trussell, "Heuristic Constraints Enforcement for Training of and Knowledge Extraction from a Fuzzy/Neural Architecture-Part I: Foundation," *IEEE Transactions on Fuzzy Systems*, Vol. 7, No. 2, pp. 143-150, 1999.
- [Cordon99] O. Cordon and F. Herrera, "A Two-Stage Evolutionary Process for Designing TSK Fuzzy Rule-Based Systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 29, No. 6, pp. 703-715, 1999.

- [Cordon00] \_\_\_\_\_, "A Proposal for Improving the Accuracy of Linguistic Modeling," *IEEE Transactions on Fuzzy Systems*, Vol. 8, No. 3, pp.335-344, 2000.
- [Conforti99] D. Conforti and L. D. Luca, "Computer Implementation of a Medical Diagnosis Problem by Patterns Classification," *Future Generation Computer Systems*, Vol. 15, No. 2, pp. 287-292, 1999.
- [Craven96] M. Craven, *Extracting Comprehensible Models from Trained Neural Networks*. Ph.D. Dissertation, Madison, WI: University Wisconsin, 1996.
- [Darwin58] C. Darwin, *Evolution by Natural Selection*. Cambridge, UK: Cambridge University Press, 1958.
- [Demiroz98] G. Demiroz, H. A. Govenir, and N. Ilter, "Learning Differential Diagnosis of Eryhemato-Squamous Diseases Using Voting Feature Intervals," *Artificial Intelligence in Medicine*, Vol. 13, No. 3, pp. 147-165, 1998.
- [Dennis96] S. Y. Dennis III, "A Bayesian Analysis of Tree-structured Statistical Decision Problems," *Journal of Statistical Planning and Inference*, Vol. 53, No. 3, pp. 323-344, 1996.
- [Devijver82] P. Devijver and J. Kittler, *Pattern Recognition: a Statistical Approach*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
- [Dimartino96] M. Dimartino, S. Fanelli, and M. Protasi, "Exploring and Comparing the Best Direct-Methods for the Efficient Training of MLP-Networks," *IEEE Transactions on Neural Networks*, Vol. 7, No. 6, pp. 1497-1502, 1996.
- [Dubois89] D. DuBois and H. Prade, "Processing fuzzy temporal knowledge," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 19, No. 4, pp. 729-744, 1989.
- [Dubois92] D. Dubois and H. Prade, "Possibility Theory as a Basis for Preference Propagation in Automated Reasoning," in *Proceedings of the IEEE International Conference on Fuzzy Systems*, 1992, pp. 821-832.
- [Duda73] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York, NY: John Wiley and Sons, 1973.
- [Dupuits98] F. M. H. M. Dupuits, A. Hasman, and P. Pop, "Computer-Based Assistance in Family Medicine," *Computer Methods and Programs in Biomedicine*, Vol. 55, No. 1, pp. 39-50, 1998.
- [Durg93] A. Durg, W. V. Stoecker, J. P. Cookson, S. E. Umbaugh, and R. H. Moss, "Identification of Variegated Coloring in Skin Tumors: Neural Network vs. Rule-Based Induction Methods," *IEEE Engineering in Medicine and Biology Magazine*, Vol. 12, No. 3, pp. 71-74, 98, 1993.
- [Dutta88] S. Dutta, "Temporal Reasoning in Medical Expert Systems," in *Proceedings of the Symposium on the Engineering of Computer-Based Medical Systems*, 1988, pp. 118-122.
- [Economou01] G. P. K. Economou, D. Lymberopoulos, E. Karvatselou, and C. Chassomeris, "A New Concept Toward Computer-Aided Medical Diagnosis - A Prototype Implementation Addressing Pulmonary Diseases," *IEEE Transactions on Information Technology in Biomedicine*, Vol. 5, No. 1, pp. 55-66, 2001.
- [Figueiredo99] M. Figueiredo and F. Gomide, "Design of Fuzzy Systems Using Neurofuzzy Networks," *IEEE Transactions on Neural Networks*, Vol. 10, No. 4, pp. 815-827, 1999.



- [Filippetti00] F. Filippetti, G. Franceschini, C. Tassoni, and P. Vas, "Recent Developments of Induction Motor Drives Fault Diagnosis Using AI Techniques," *IEEE Transactions on Industrial Electronics*, Vol. 47, No. 5, pp. 994-1004, 2000.
- [Foran00] D. J. Foran, D. Comaniciu, P. Meer, and L. A. Goodell, "Computer-Assisted Discrimination Among Malignant Lymphomas and Leukemia Using Immunophenotyping, Intelligent Image Repositories, and Telemicroscopy," *IEEE Transactions of Information Technology in Biomedicine*, Vol. 4, No. 4, pp. 265-273, 2000.
- [Fu94] L. M. Fu, "Rule Generation from Neural Networks," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 24, No. 8, pp. 1114-1124, 1994.
- [Fu96] L. M. Fu, H. H. Hsu, and J. C. Principe, "Incremental Backpropagation Learning Networks," *IEEE Transactions on Neural Networks*, Vol. 7, No. 3, pp. 757-761, 1996.
- [Gallant93] S. I. Gallant, *Neural Network Learning and Expert Systems*. Cambridge, MA: MIT Press, 1993.
- [Goldberg89] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
- [Gonzalez93] A. J. Gonzalez and D. D. Dankel, *The Engineering of Knowledge-Based Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
- [González99] A. González and R. Perez, "SLAVE: A Genetic Learning System Based on an Iterative Approach," *IEEE Transactions on Fuzzy Systems*, Vol. 7, No. 2, pp. 176-191, 1999.
- [Goonatilake95] S. Goonatilake and S. Khebbal, *Intelligent Hybrid Systems*. Chichester, UK: Wiley, 1995.
- [Hagan94] M. T. Hagan and M. Menhaj, "Training Feedforward Networks with the Marquardt Algorithm," *IEEE Transactions on Neural Networks*, Vol. 5, No. 6, pp. 989-993, 1994.
- [Hall92] L. Hall and S. Romaniuk, "Performance Issues of a Hybrid Symbolic, Connectionist Learning Algorithm," in A. Kandel, editor, *Hybrid Architectures for Intelligent Systems*, pp. 106-133, 1992.
- [Hayashi00] Y. Hayashi, "A Comparison Between Two Neural Network Rule Extraction Techniques for the Diagnosis of Hepatobiliary Disorders," *Artificial Intelligence in Medicine*, Vol. 20, No. 3, pp. 205-216, 2000.
- [Haykin94] S. Haykin, *Neural Networks: A Comprehensive Foundation*. New York, NY: MacMillan, 1994.
- [Heng98] R. B. W. Heng and M. J. M. Nor, "Statistical-Analysis of Sound and Vibration Signals for Monitoring Rolling Element Bearing Condition," *Applied Acoustics*, Vol. 53, No. 1-3, pp. 211-226, 1998.
- [Hilario94] M. Hilario, "An Overview of Strategies for Neurosymbolic Integration," in R. Sun and L. Bookman, editors, *Computational Architectures Integrating Symbolic and Neural Process*, New York, NY: Kluwer Academic Publishers, 1994.
- [Hinton96] G. E. Hinton, "Learning Distributed Representations of Concepts," in *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, Amherst, MA, pp. 1-12, 1996.

- [Holland75] J. H. Holland, *Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Ann Arbor, MI: University of Michigan Press, 1975.
- [Huang97a] S. H. Huang and M. R. Endsley, "Providing Understanding of the Behavior of Feedforward Neural Networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 27, No. 3, pp. 465-474, 1997.
- [Huang97b] S. J. Huang and C. L. Huang, "Application of Genetic-Based Neural Networks to Thermal Unit Commitment," *IEEE Transactions on Power Systems*, Vol. 12, No. 2, pp. 654-660, 1997.
- [Hwang94] Y. S. Hwang and S. Y. Bang, "A Neural Network Model APC-III and Its Application to Unconstrained Handwritten Digit Recognition," in *Proceedings of International Conference on Neural Information Processing*, Seoul: Korean Association for Intelligent Information Systems, pp. 1500-1505, 1994.
- [Hwang97] Y. S. Hwang and S. Y. Bang, "An Efficient Method to Construct a Radial Basis Function Neural Network Classifier," *Neural Networks*, Vol. 10, No. 8, pp. 1495-1503, 1997.
- [Ishibuchi99] H. Ishibuchi, T. Nakashima, and T. Murata, "Performance Evaluation of Fuzzy Classifier Systems for Multidimensional Pattern Classification Problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 29, No. 5, pp. 601-618, 1999.
- [Ishibuchi97] H. Ishibuchi, M. Nii, and T. Murata, "Linguistic Rule Extraction from Neural Networks and Genetic-Algorithm-Based Rule Selection," in *Proceedings of International Conference on Neural Networks (ICNN'97)*, Houston, TX, Vol. 4, pp. 2390-2395, 1997.
- [Ishibuchi92] H. Ishibuchi, K. Nozaki, and H. Tanaka, "Distributed Representation of Fuzzy Rules and Its Application to Pattern Classification," *Fuzzy Sets and Systems*, Vol. 52, pp. 21-32, 1992.
- [Ishibuchi95] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting Fuzzy If-Then Rules for Classification Problems Using Genetic Algorithm," *IEEE Transactions on Fuzzy Systems*, Vol. 3, No. 3, pp. 260-270, 1995.
- [Jang93a] J. S. R. Jang and C. T. Sun, "Functional Equivalence Between Radial Basis Function Networks and Fuzzy Inference Systems," *IEEE Transactions on Neural Networks*, Vol. 4, No. 1, pp. 156-158, 1993.
- [Jang93b] J. S. Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, No. 3, pp. 665-685, 1993.
- [Jin99] Y. Jin, W. V. Seelen, and B. Sendhoff, "On Generating FC<sup>3</sup> Fuzzy Rule Systems from Data Using Evolution Strategies," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 29, No. 6, pp. 829-845, 1999.
- [Jin00] Y. Jin, "Fuzzy Modeling of High-Dimensional Systems: Complexity Reduction and Interpretability Improvement," *IEEE Transactions on Fuzzy Systems*, Vol. 8, No. 2, pp. 212-221, 2000.
- [Juang98] C. F. Juang and C. T. Lin, "An On-Line Self-Constructing Neural Fuzzy Inference Network and Its Applications," *IEEE Transactions on Fuzzy Systems*, Vol. 6, No. 1, pp. 12-32, 1998.

- [Keller85] J. M. Keller, M. R. Gray, and J. A. Givens, "A Fuzzy K-Nearest Neighbor Algorithm," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 15, No. 4, pp. 580-585, 1985.
- [Kiseliouva01] T. Kiseliouva and C. Moraga, "Modelling Temporal Distribution of Symptoms and Diseases with Fuzzy Logic," in *Proceedings of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, 2001, pp. 1637-1641.
- [Kohonen97] T. Kohonen, *Self-Organizing Maps*. 2nd Ed., New York, NY: Springer, 1997.
- [Kovalerchuk00] B. Kovalerchuk, E. Vityaev, and J. F. Ruiz, "Consistent Knowledge Discovery in Medical Diagnosis," *IEEE Engineering in Medicine and Biology Magazine*, Vol. 19, No. 4, pp. 26-37, 2000.
- [Kumar97] S. A., Kumar, H. V. Ravindra, and Y. G. Srinvasa, "In-Process Tool Wear Monitoring Through Time-Series Modeling and Pattern-Recongition," *International Journal of Production Research*, Vol. 35, No. 3, pp. 739-751, 1997.
- [Kundu94] A. Kundu, G. C. Chen, and C. E. Persons, "Transient Sonar Signal Classification Using Hidden Markov Models and Neural Nets," *IEEE Journal of Oceanic Engineering*, Vol. 19, No. 1, pp. 87-99, 1994.
- [Le95] T. Van Le, "A Fuzzy Temporal Logic Scheme for Fuzzy Dynamic Systems," in *Proceedings of the Third Australian and New Zealand Conference on Intelligent Information Systems, (ANZIIS-95)*, 1995, pp. 152-157.
- [Li00] X. Li, S. K. Tso, and J. Wang, "Real-Time Tool Condition Monitoring Using Wavelet Transforms and Fuzzy Techniques," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 30, No. 3, pp. 352-357, 2000.
- [Linkens93] D. A. Linkens, and J. Nie, "Learning Control Using Fuzzified Self-Organizing Radial Basis Function Neural Network," *IEEE Transactions on Fuzzy Systems*, Vol. 1, No. 4, pp.280-287, 1993.
- [Lippmann89] R. P. Lippmann, "Pattern Classification Using Neural Networks," *IEEE Communications Magazine*, Vol. 27, No. 11, pp. 47-64, 1989.
- [Liu99] G. P. Liu and V. Kadirkamanathan, "Multiobjective Criteria for Neural Network Structure Selection and Identification of Nonlinear Systems Using Genetic Algorithms," *Proceedings of Control Theory and Applications*, Vol. 146, No. 5, pp. 373-382, 1999.
- [Lu97] J. Lu, P. Brinkley, and S. C. Fang, "A Fuzzy Expert System Model for RF Receiver Module Testing," *International Journal of Systems Science*, Vol. 28, No. 8, pp. 791-798, 1997.
- [Makris99] L. Makris, I. Kamilatos, E. V. Kopsacheilis, and M. G. Strintzis, "Teleworks: A CSCW Application for Remote Medical Diagnosis Support and Teleconsultation," *IEEE Transactions on Information Technology in Biomedicine*, Vol. 2, No. 2, pp. 62-73, 1998.
- [Mamdani74] E. H. Mamdani, "Application of Fuzzy Algorithms for Control of Simple Dynamic Plant," *IEEE Proceedings*, Vol. 121, No. 12, pp. 1585-1588, 1974.
- [Man00] L. W. Man, L. Wai, S. L. Kwong, S. N. Po, and J. C. Y. Cheng, "Discovering Knowledge From Medical Databases Using Evolutionary Algorithms," *IEEE Engineering in Medicine and Biology Magazine*, Vol. 19, No. 4, pp. 45-55, 2000.

- [Maniezzo94] V. Maniezzo, "Genetic Evolution of the Topology and Weight Distribution of Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 5, No. 1, pp. 39-53, 1994.
- [Marcu97] T. Marcu and L. Mirea, "Robust Detection and Isolation of Process Faults Using Neural Networks," *IEEE Control Systems Magazine*, Vol. 17, No. 5, pp. 72-79, 1997.
- [Matthews01] R. A. J. Matthews, "Why Should Clinicians Care About Bayesian Methods?," *Journal of Statistical Planning and Inference*, Vol. 94, No. 1, pp. 43-58, 2001.
- [McGarry99] K. McGarry, S. Wermter, and J. MacIntyre, "Hybrid Neural Systems: From Simple Coupling to Fully Integrated Neural Networks," *Neural Computing Surveys*, Vol. 2, pp. 62-93, 1993.
- [Medsker94] L. R. Medsker, *Hybrid Neural Network and Expert Systems*. Boston, MA: Kluwer Academic Publishers, 1994.
- [Meesad98] P. Meesad, *Pattern Classification by an Incremental Fuzzy Learning Neural Network*, MS Thesis, Stillwater, OK: Oklahoma State University, 1998.
- [Meesad00] P. Meesad and G. Yen, "Pattern Classification by a Neurofuzzy Network: Application to Vibration Monitoring," *ISA Transactions*, Vol. 39 No. 3, pp. 293-308, 2000.
- [Michalski86] R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac, "The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains," in *Proceedings of the Fifth National Conference on Artificial Intelligence*, Philadelphia, PA: Morgan Kaufmann, pp.1041-1045, 1986.
- [Mitra97] S. Mitra, R. K. De, and S. K., Pal, "Knowledge-Based Fuzzy MLP for Classification and Rule Generation," *IEEE Transactions on Neural Networks*, Vol. 8, No. 6, pp. 1338-1350, 1997.
- [Mitra00] S. Mitra and Y. Hayashi, "Neuro-Fuzzy Rule Generation: Survey in Soft Computing Framework," *IEEE Transactions on Neural Networks*, Vol. 11, No. 3, pp. 748-768, 2000.
- [Moody89] J. Moody and C. J. Darken, "Fast Learning in Networks of Locally-Tuned Processing Units," *Neural Computation*, Vol. 1, No. 2, pp. 281-294, 1989.
- [Mucientes01] M. Mucientes, R. Iglesias, C. V. Regueiro, A. Bugarin, P. Carinena, and S. Barro, "Fuzzy temporal rules for mobile robot guidance in dynamic environments," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 31, No. 3, pp. 391-398, 2001.
- [Musavi92] M. Musavi, W. Ahmed, K. Chan, K. Faris, and D. Hummels, "On the Training of Radial Basis Function Classifiers," *Neural Networks*, Vol. 5, pp. 595-603, 1992.
- [Narazaki96] H. Narazaki, T. Watanabe, and M. Yamamoto, "Reorganizing Knowledge in Neural Networks: An Explanatory Mechanism for Neural Networks in Data Classification Problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 26, No. 1, pp. 107-117, 1996.
- [Nawa99] N. E. Nawa and T. Furuhashi, "Fuzzy System Parameters Discovery by Bacterial Evolutionary Algorithm," *IEEE Transactions on Fuzzy Systems*, Vol. 7, No. 5, pp. 608-616, 1999.

- [Nikovski00] D. Nikovski, "Constructing Bayesian Networks for Medical Diagnosis from Incomplete and Partially Correct Statistics," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, No. 4, pp. 509-516, 2000.
- [Nozaki96] K. Nozaki, H. Ishibuchi, and H. Tanaka, "Adaptive Fuzzy Rule-Based Classification Systems," *IEEE Transactions on Fuzzy Systems*, Vol. 4, No. 3, pp. 138-250, 1996.
- [Omlin98] C. W. Omlin, K. K. Thornber, and C. L. Giles, "Fuzzy Finite-State Automata Can Be Deterministically Encoded into Recurrent Neural Networks," *IEEE Transactions on Fuzzy Systems*, Vol. 6, No. 1, pp. 76-89, 1998.
- [Pal92] S. K. Pal and Mandal, D. P. "Linguistic Recognition System Based on Approximate Reasoning," *Information Science*, Vol. 61, pp. 135-161, 1992.
- [Pan99] J. Pan, "Vector-Scalar Classification for Transform Image Coding," *IEEE Transactions on Image Processing*, Vol. 8, No. 9, pp. 1175-1182, 1999.
- [Pattichis95] C. S. Pattichis, C. N. Schizas, and L. T. Middleton, "Neural Network Models in EMG Diagnosis," *IEEE Transactions on Biomedical Engineering*, Vol. 42, No. 5, pp. 486-496, 1995.
- [Peña99] C. A. Peña-Reyes and M. Sipper, "Designing Breast Cancer Diagnostic via a Hybrid Fuzzy-Genetic Methodology," in *Proceedings of the 1999 IEEE International Fuzzy Systems Conference*, pp. 135-139, 1999.
- [Peña00] C. A. Peña-Reyes and M. Sipper, "Evolutionary Computation in Medicine: an Overview," *Artificial Intelligence in Medicine*, Vol. 19, No. 1, pp. 1-23, 2000.
- [Podgorelec99] V. Podgorelec, P. Kokol, and J. Završnik, "Medical Diagnosis Prediction Using Genetic Programming," in *Proceedings of the 12th IEEE Symposium on Computer-Based Medical Systems*, pp. 202-207, 1999.
- [Pradhan96] M. Pradhan, M. Henrion, G. Provan, B. Del Favero, and K. Huang, "The Sensitivity of Belief Networks to Imprecise Probabilities: an Experimental Investigation," *Artificial Intelligence*, Vol. 85, No. 1-2, pp. 363-397, 1996.
- [Qi95] Y. Qi and B. R. Hunt, "A Multiresolution Approach to Computer Verification of Handwritten Signatures," *IEEE Transactions on Image Processing*, Vol. 4, No. 6, pp. 870-874, 1995.
- [Rumelhart86a] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing*. Cambridge, MA: MIT Press, 1986.
- [Rumelhart86b] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-Propagating Errors," *Nature*, Vol. 323, pp. 533-536, 1986.
- [Rumelhart86c] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagating," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland (Eds.), Cambridge, MA: MIT Press, 1986.
- [Russo98] M. Russo, "FuGeNeSys: A Fuzzy Genetic Neural System for Fuzzy Modeling," *IEEE Transactions on Fuzzy Systems*, Vol. 6, No. 3, pp. 373-388, 1998.
- [Sabatini01] A. M. Sabatini, "A Digital-Signal-Processing Technique for Ultrasonic Signal Modeling and Classification," *IEEE Transactions on Instrumentation and Measurement*, Vol. 50, No. 1, pp. 15-21, 2001.
- [Sacha00] J. P. Sacha, K. J. Cios, and L. S. Goodenday, "Issues in Automating Cardiac SPECT Diagnosis," *IEEE Engineering in Medicine and Biology Magazine*, Vol. 19, No. 4, pp. 78-88, 2000.

- [Samad92] T. Samad, "Hybrid Distributed/Local Connectionist Architectures," in A. Kandel, editor, *Hybrid Architecture for Intelligent Systems*, pp. 200-218, 1992.
- [Seka97] L. P. Seka, A. Fresnel, D. Delamarre, C. Riou, A. Burgun, B. Pouliquen, R. Duvaufferrier, and P. Le Beux, "Computer Assisted Medical Diagnosis Using the Web," *International Journal of Medical Informatics*, Vol. 47, No. 1-2, pp. 51-56, 1997.
- [Setiono96] R. Setiono and H. Liu, "Symbolic Representation of Neural Networks," *IEEE Computer*, Vol. 29, No. 3, pp. 71-77, 1996.
- [Setiono97] R. Setiono, "A Penalty Function Approach for Pruning Feedforward Neural Networks," *Neural Computation*, Vol. 1, pp. 185-204, 1997.
- [Setiono00] \_\_\_\_\_, "Generating Concise and Accurate Classification Rules for Breast Cancer Diagnosis," *Artificial Intelligence in Medicine*, Vol. 18, No. 3, pp. 205-219, 2000.
- [Setnes98] M. Setnes, R. Babuska, and H. B. Verbruggen, "Rule-Based Modeling: Precision and Transparency," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 28, No. 1, pp. 165-169, 1998.
- [Setnes98] M. Setnes, R. Babuska, U. Kaymak, and H. R. van Nauta Lemke, "Similarity Measures in Fuzzy Rule Base Simplification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 28, No. 3, pp. 376-386, 1998.
- [Shi99] Y. Shi, R. Eberhart, and Y. Chen, "Implementation of Evolutionary Fuzzy Systems," *IEEE Transactions on Fuzzy Systems*, Vol. 7, No. 2, pp. 109-119, 1999.
- [Simpson92] P. K. Simpson, "Fuzzy Min-Max Neural Networks-Part 1: Classification," *IEEE Transactions on Neural Networks*, Vol. 3, No. 5, pp. 776-786, 1992.
- [Smith68] F. W. Smith, "Pattern Classifier Design by Linear Programming," *IEEE Transactions on Computers*, Vol. 17, pp. 67-372, 1968.
- [Specht88] D. F. Specht, "Probabilistic Neural Networks for Classification, Mapping, or Associative Memory," in *Proceeding of the IEEE International Conference on Neural Networks*, San Diego, CA, Vol. 1, 1988, pp. 525-532.
- [Specht90] D. F. Specht, "Probabilistic Neural Networks and the Polynomial Adaline as Complementary Techniques for Classification," *IEEE Transactions on Neural Networks*, Vol. 1, No. 1, pp. 111-121, 1990.
- [Specht92] D. F. Specht, "Enhancements to Probabilistic Neural Networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'92)*, Baltimore, MD, Vol. 1, pp. 761-768, 1992.
- [Specht94] D. F. Specht, and H. Romsdahl, "Experience with Adaptive Probabilistic Neural Networks and Adaptive General Regression Neural Networks," in *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, Orlando, FL, Vol. 2, pp. 1203-1208, 1994.
- [Sugeno88] M. Sugeno and G. T. Kang, "Structure Identification of Fuzzy Model," *Fuzzy Sets and Systems*, Vol. 28, No. 1, pp. 15-33, 1988.
- [Sun95] R. Sun, "Robust Reasoning: Integrating Rule-Based and Similarity-Based Reasoning," *Artificial Intelligence*, Vol. 75, No. 2, pp. 214-295, 1995.
- [Taha97] I. Taha, "A Hybrid Intelligent Architecture for Revising Domain Knowledge," *Ph.D. Thesis*, Austin, TX: The University of Texas at Austin, 1997.

- [Taha99] I. A. Taha and J. Ghosh, "Symbolic Interpretation of Artificial Neural Networks," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, No. 3, pp. 448-463, 1999.
- [Takagi85] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and Its Application to Modeling and Control," *IEEE Transactions on System, Man, Cybernetics*, Vol. 15, No. 1, pp. 116-132, 1985.
- [Tan97] A. H. Tan, "Cascade ARTMAP: Integrating Neural Computation and Symbolic Knowledge Processing," *IEEE Transactions on Neural Networks*, Vol. 8, No. 2, pp. 237-250, 1997.
- [Tang98] K. S. Tang, I. F. Man, Z. F. Liu, and S. Kwong, "Minimal Fuzzy Memberships and Rules Using Hierarchical Genetic Algorithms," *IEEE Transactions on Industrial Electronics*, Vol. 45, No. 1, pp. 162-169, 1998.
- [Tickle98] A. B. Tickle, R. Andrews, M. Golea, and J. Diederich, "The Truth Will Come to Light: Directions and Challenges in Extracting the Knowledge Embedded within Trained Artificial Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 9, pp. 1057-1068, 1998.
- [Tilbury00] J. B. Tilbury, W. J. V. Eetvelt, J. M. Garibaldi, J. S. H. Curnsw, and E. C. Ifeachor, "Receiver Operating Characteristic Analysis for Intelligent Medical Systems-A New Approach for Finding Confidence Intervals," *IEEE Transactions on Biomedical Engineering*, Vol. 47, No. 7, pp. 952-963, 2000.
- [Tino99] P. Tino and M. Koteles, "Extracting Finite-State Representations from Recurrent Neural Networks Trained on Chaotic Symbolic Sequences," *IEEE Transactions on Neural Networks*, Vol. 10, No. 2, pp. 284-302, 1999.
- [Tontini96] G. Tontini, and A. A. de Queiroz, "RBF Fuzzy-ARTMAP: A New Fuzzy Neural Network for Robust On-Line Learning and Identification Of Patterns," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 2, pp. 1364-1369, 1996.
- [Towell94] G. Towell and J. Shavlik, "Knowledge-Based Artificial Neural Networks," *Artificial Intelligence*, Vol. 70, No. 1-2, pp. 119-165, 1994.
- [Uebele96] V. Uebele, S. Abe, and M. S. Lan, "A Neural-Network-Based Fuzzy Classifier," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 25, No. 2, pp. 353-361, 1996.
- [Vuorimaa95] P. Vuorimaa, T. Jukarainen, and E. Karpanoja, "A Neuro-Fuzzy System for Chemical Agent Detection," *IEEE Transactions on Fuzzy Systems*, Vol. 3, No. 4, pp. 415-424, 1995.
- [Walter00] D. Walter and C. K. Mohan, "ClaDia: a Fuzzy Classifier System for Disease Diagnosis," in *Proceedings of the 2000 Congress on Evolutionary Computation*, pp. 1429-1435, 2000.
- [WangHong98] C. H. Wang, T. P. Hong, and S. S. Tseng, "Integrating Fuzzy Knowledge by Genetic Algorithms," *IEEE Transactions on Evolutionary Computation*, Vol. 2, No. 4, pp.138-149, 1998.
- [Wang92] L. X. Wang and J. M. Mendel, "Generating Fuzzy Rules by Learning from Examples," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 22, No. 6, pp. 1414-1427, 1992.

- [Wermter00] S. Wermter and R. Sun, "An Overview of Hybrid Neural System," in *Hybrid Neural Systems*, S. Wermter and R. Sun, Eds., Berlin, Germany: Springer-Verlag, 2000, pp.1-13.
- [West00] D. West and V. West, "Model Selection for a Medical Diagnostic Decision Support System: A Breast Cancer Detection Case," *Artificial Intelligence in Medicine*, Vol. 20, No. 3, pp. 183-204, 2000.
- [Wiegerinck99] W. A. J. J. Wiegerinck, H. J. Kappen, E. W. M. T. ter Braak, W. J. P. P. ter Burg, M. J. Nijman, Y. L. O, and J. P. Neijt, "Approximate Inference for Medical Diagnosis," *Pattern Recognition Letters*, Vol. 20, No. 11-13, pp. 1231-1239, 1999.
- [Williamson96] J. R. Williamson, "Gaussian ARTMAP: A Neural Network for Fast Incremental Learning of Noisy Multidimensional Maps," *Neural Networks*, Vol. 9, No. 5, pp. 881-897, 1996.
- [Wolberg90] W. H. Wolberg and O. L. Mangasarian, "Multi-surface Method of Pattern Separation for Medical Diagnosis Applied to Breast Cytology," *Proceedings of the National Academy of Sciences, U.S.A.*, Vol. 87, pp. 9193-9196, 1990.
- [Yan00] L. Yan and D. J. Miller, "General Statistical Inference for Discrete and Mixed Spaces by an Approximate Application of the Maximum Entropy Principle," *IEEE Transactions on Neural Networks*, Vol. 11, No. 3, pp. 558-573, 2000.
- [Yao97] X. Yao and Y. Liu, "A New Evolutionary System for Evolving Artificial Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 8, No. 3, pp. 694-713, 1997.
- [Yen99] G. Yen and P. Meesad, "Pattern Classification by an Incremental Learning Fuzzy Neural Network," in *Proceedings of the International Joint Conference on Neural Networks*, pp. 3230-3235, 1999.
- [Yen01] \_\_\_\_\_, "An Effective Neuro-Fuzzy Paradigm for Machinery Condition Health Monitoring," *IEEE Transactions on System, Man, and Cybernetics, Part B: Cybernetics*, Vol. 31, No. 4, pp. 523-536, 2001.
- [Yen91] J. Yen, R. Neches, and R. MacGregor, "CLASP: Integrating Term Subsumption Systems and Production Systems," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 3, No. 1, pp. 25-32, 1991.
- [Zadeh65] L. A. Zadeh, "Fuzzy Sets," *Information and Control*, Vol. 8, pp. 338-353, 1965.
- [Zahan01] S. Zahan, "A Fuzzy Approach to Computer-Assisted Myocardial Ischemia Diagnosis," *Artificial Intelligence in Medicine*, Vol. 21, No. 1-3, pp. 271-275, 2001.



2  
VITA

Phayung Meesad

Candidate for the Degree of

Doctor of Philosophy

Thesis: A HYBRID INTELLIGENT SYSTEM AND ITS APPLICATION  
TO MEDICAL DIAGNOSIS

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Nakornratchasima, Thailand, On October 19, 1968, the son of Phayom and Hem Meesad.

Education: Received a Diploma degree in Electrical Power Technology from the Institute of Technology and Vocational Education Nakornratchasima, Thailand in March 1989; Received a Bachelor of Science in Technical Education degree (Electrical Engineering) from King Mongkut's Institute of Technology North Bangkok, Bangkok, Thailand in May 1994. Received a Master of Science degree in Electrical Engineering at Oklahoma State University in December 1998. Completed the requirements for the Doctor of Philosophy with a major in Electrical Engineering at Oklahoma State University in May 2002.

Experience: Employed as a technician, 1990 to 1994, and a lecturer, 1994 to present by King Mongkut's Institute of Technology North Bangkok, Thailand; employed by Oklahoma State University, School of Electrical and Computer Engineering as a teaching assistant, Spring 1998; employed by Oklahoma State University, Computer and Information Services as a Computer Assistant, Fall 1999 to Fall 2001.

Professional Memberships: The Institute of Electrical and Electronics Engineers.