

CAIRO UNIVERSITY
INSTITUTE OF STATISTICAL STUDIES AND RESEARCH
DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE

A NEW APPROACH FOR EXTRACTING FUZZY RULES USING ARTIFICIAL NEURAL NETWORKS

Submitted By

Mohamed Farouk Abdel Hady Mohamed

Teaching Assistant at Institute of Statistical Studies and Research

Supervised By

Prof. Adel S. Elmaghraby

Institute of Statistical Studies and Research
Cairo University

Dr. Mervat H. Gheith

Institute of Statistical Studies and Research
Cairo University

Dr. Mahmoud A. Wahdan

Ministry of Telecommunications and
Information Technology

A thesis submitted to the institute of Statistical Studies and Research, Cairo University, in partial fulfillment of the requirements for the degree of master in Computer Science in the Department of Computer and Information Science.

2005

I certify that this work has not been accepted in substance for any academic degree and is not being concurrently submitted in candidature for any other degree.

Any portions of this thesis for which I am indebted to other sources are mentioned and explicit reference are given.

Student: Mohamed Farouk Abdel Hady

ACKNOWLEDGMENTS

I would like to thank everyone who has given his assistance and support during the completion of this thesis. Special thanks must go to my supervisors: Prof. Adel Elmaghraby, Dr. Mervat Geith, and Dr. Mahmoud Wahdan. They gave me freedom to do my research more independently. Their valuable comments on my work helped me to have a successful defense. Second, I would like to thank my colleagues at Institute of Statistical Studies and Research (ISSR) especially Dr. Hesham Hefny. Whenever I had a problem, he was always a friend. He always understood and supported me. Finally, I would like to thank my committee members.

There is a person without whom I would not be able to finish my M.Sc.: My mother. She knew that M.Sc. was my dream and she always supported me. She sacrificed a lot for me to reach my dream.

The UCI Repository of Machine Learning Databases and Domain theories (ml-repository@ics.uci.edu) kindly supplied the benchmark data used in this thesis.

ABSTRACT

Knowledge discovery and data mining have been become very important in our society where the amount of data double almost every year. In these complex databases, much information is often hidden as trends, dependencies and relationships. Data mining is the process of acquiring knowledge, such as behavioral patterns, associations, and significant structures from data, and transforming this information into a compact and interpretable decision system. For complex and high-dimensional classification tasks, data-driven identification of classifiers has to deal with structural problems such as the *effective initial partitioning of the input domain* and the *selection of the relevant features*. This thesis focuses on these problems by presenting a new neuro-fuzzy approach for building interpretable fuzzy rules, used for pattern classification and medical diagnosis. The proposed approach combines the merits of the fuzzy logic theory, and neural networks. Fuzzy rules are extracted in three phases: initialization, optimization, and simplification of the fuzzy model. In the first phase, the data set is partitioned automatically into a set of clusters based on input-similarity and output-similarity tests. Membership functions associated with each cluster are defined according to statistical means and variances of the data points. Then, a fuzzy if-then rule is extracted from each cluster to form a fuzzy model. In the second phase, the extracted fuzzy model is used as starting point to construct a network then the fuzzy model parameters are refined, by analyzing the nodes of the network that was trained via the backpropagation gradient descent method. Real-world classification applications usually have many features. This increases the complexity of the classification task. Choosing a subset of the features may increase accuracy and reduce complexity of the knowledge acquisition. In the third phase, feature subset selection by relevance simplification method is used to reduce the extracted fuzzy rules. Finally, A number of case studies is applied to evaluate the effectiveness of the proposed approach according to the defined evaluation criteria.

TABLE OF CONTENTS

ACKNOWLEDGMENTS.....	II
ABSTRACT	III
TABLE OF CONTENTS	IV
LIST OF FIGURES.....	VIII
LIST OF TABLES.....	X
CHAPTER 1	1
INTRODUCTION	1
1.1 Background	1
1.2 Problem Statement	2
1.3 Previous Work	4
1.4 Organization of Thesis	6
CHAPTER 2	8
RULE EXTRACTION BACKGROUND.....	8
2.1 Overview of Artificial Neural Networks.....	8
2.1.1 Introduction to Artificial Neural Network	8
2.1.1.1 Processing Units.....	9
2.1.1.2 Activation and Output Functions.....	9
2.1.1.2.1 Non-local Transfer Functions	10
2.1.1.2.2 Local Transfer Functions	11
2.1.1.3 Network Topologies	12
2.1.1.4 Training of Artificial Neural Networks	12
2.1.1.5 Learning Algorithms.....	13
2.1.2 Local Function Neural Networks.....	13
2.1.2.1 Advantages of Local Function Networks	14

2.1.2.2	<i>Disadvantages of Local Function Networks</i>	14
2.1.3	<i>Architecture of Rapid Back Propagation Networks</i>	15
2.2	Overview of Fuzzy Set Theory	18
2.2.1	<i>Fuzzy Sets</i>	18
2.2.2	<i>Membership Functions</i>	19
2.2.3	<i>Fuzzy Rules and Fuzzy Reasoning</i>	20
2.2.3.1	<i>Fuzzy If-Then Rules</i>	21
2.2.3.2	<i>Fuzzy Reasoning</i>	21
2.2.4	<i>Fuzzy Inference Systems</i>	21
2.2.4.1	<i>Mamdani Fuzzy Model</i>	22
2.2.4.2	<i>Tsukamoto Fuzzy Model</i>	23
2.2.4.3	<i>Sugeno Fuzzy Model</i>	23
2.2.4.4	<i>Overview of Input Space Partitioning</i>	24
2.3	Overview of Neuro-Fuzzy and Soft Computing	25
2.3.1	<i>Soft Computing</i>	26
2.3.2	<i>General Comparisons of Fuzzy Systems and Neural Networks</i>	26
2.3.3	<i>Different Neuro-Fuzzy Hybridizations</i>	27
2.3.4	<i>Techniques of Integrating Neuro-Fuzzy Models</i>	27
2.3.5	<i>Neural Fuzzy Systems</i>	28
2.4	Evaluation Criteria for neuro-fuzzy Approaches	28
2.4.1	<i>Computational Complexity</i>	28
2.4.2	<i>Quality of the Extracted Rules</i>	29
2.4.3	<i>Translucency</i>	29
2.4.4	<i>Consistency</i>	30
2.4.5	<i>Portability</i>	30
2.4.6	<i>Space Exploration Methodology</i>	30
2.5	Some Rule Extraction Algorithms	30
2.5.1	<i>RULEX Technique</i>	30
2.5.1.1	<i>Description</i>	30
2.5.1.2	<i>Algorithm Evaluation</i>	31
2.5.2	<i>M-of-N Technique</i>	34
2.5.2.1	<i>Description</i>	34
2.5.2.2	<i>Algorithm Evaluation</i>	34
2.5.3	<i>BIO-RE Technique</i>	36
2.5.3.1	<i>Description</i>	36
2.5.3.2	<i>Algorithm Evaluation</i>	36
2.5.4	<i>Partial-RE Technique</i>	37
2.5.4.1	<i>Description</i>	37
2.5.4.2	<i>Algorithm Evaluation</i>	38
2.5.5	<i>Full-RE Technique</i>	39
2.5.5.1	<i>Description</i>	39
2.5.5.2	<i>Algorithm Evaluation</i>	39
CHAPTER 3		41
FRULEX – FUZZY RULES EXTRACTOR		41
3.1	Overview of FRULEX Approach	41

3.2	Self-Constructing Rule Generator	44
3.3	Backpropagation Training for RBP Neural Network.....	47
3.3.1	Introduction	47
3.3.2	Backpropagation Learning Algorithm	47
3.4	Feature Subset Selection by Relevance.....	52
3.4.1	Overview of Feature Subset Selection	53
3.4.1.1	Search Algorithms	54
3.4.1.1.1	Exponential Search Algorithms	54
3.4.1.1.2	Sequential Search Algorithms.....	54
3.4.1.1.3	Randomized Search Algorithms	55
3.4.1.2	Filter Approach	56
3.4.1.3	Wrapper Approach	56
3.4.2	Feature Subset Selection By Feature Relevance	56
3.4.2.1	Phase 1: Sorted Search Phase.....	57
3.4.2.2	Phase 2: Neighbor Search Phase	57
3.4.2.3	Phase 3: Finding Final Subset Phase.....	58
CHAPTER 4		62
EVALUATION OF FRULEX APPROACH	62	
4.1	Description of Case Studies	62
4.2	Case Study 1: Iris Flower Classification Dataset.....	63
4.2.1	Description of Case Study	63
4.2.2	Initialization Phase.....	64
4.2.3	Optimization Phase.....	64
4.2.4	Simplification Phase.....	65
4.2.5	Analysis of Results	68
4.3	Case Study 2: Wisconsin Breast Cancer Dataset.....	71
4.3.1	Description of Case Study	71
4.3.2	Initialization Phase.....	72
4.3.3	Optimization Phase.....	72
4.3.4	Simplification Phase.....	73
4.3.5	Analysis of Results	76
4.4	Case Study 3: Cleveland Heart Disease Dataset.....	79
4.4.1	Description of Case Study	79
4.4.2	Initialization Phase.....	80
4.4.3	Optimization Phase.....	80
4.4.4	Simplification Phase.....	81
4.4.5	Analysis of Results	84
4.5	Case Study 4: Pima Indians Diabetes Dataset	87
4.5.1	Description of Case Study	87
4.5.2	Initialization Phase.....	87
4.5.3	Optimization Phase.....	88
4.5.4	Simplification Phase.....	89
4.5.5	Analysis of Results	92

4.6	Evaluation	94
4.6.1	<i>Rule Format.....</i>	94
4.6.2	<i>Complexity of the Approach</i>	94
4.6.3	<i>Quality of the Extracted Rules</i>	94
4.6.3.1	<i>Comprehensibility.....</i>	95
4.6.3.2	<i>Accuracy.....</i>	95
4.6.3.3	<i>Fidelity.....</i>	95
4.6.4	<i>Portability of the Approach</i>	95
4.6.5	<i>Translucency of the Approach</i>	96
4.6.6	<i>Consistency of the Approach</i>	96
CHAPTER 5		97
CONCLUSIONS AND FUTURE WORK.....		97
5.1	Conclusions	97
5.2	Future Work	99
BIBLIOGRAPHY.....		100
APPENDIX A		106
LIST OF ABBREVIATIONS.....		106
APPENDIX B		107
FRULEX FLOWCHART		107
APPENDIX C		108
FRULEX CLASS DIAGRAM.....		108

LIST OF FIGURES

Figure 2.1. <i>Artificial Neural Network</i>	9
Figure 2.2. <i>Decision regions formed using sigmoid processing functions</i>	11
Figure 2.3. <i>Construction of a ridge [Andrews and Geva, 1995]</i>	15
Figure 2.4. <i>Cylindrical Extension of a ridge [Andrews and Geva, 1995]</i>	16
Figure 2.5. <i>Intersection of two Ridges [Andrews and Geva, 1995]</i>	16
Figure 2.6. <i>Production of an LRU [Andrews and Geva, 1995]</i>	17
Figure 2.7. <i>Membership Functions: (a) Triangle (b) Trapezoid [Jang et al., 1998]</i>	19
Figure 2.8. <i>Bell Membership Function [Jang et al., 1998]</i>	20
Figure 2.9. <i>Fuzzy Inference System [Jang et al., 1998]</i>	22
Figure 2.10. <i>Partitioning Methods (a) grid partition; (b) tree partition; (c) scatter partition [Jang et al., 1998]</i>	25
Figure 3.1. <i>Outline of FRULEX Approach</i>	42
Figure 3.2. <i>Architecture of the Proposed Backpropagation Neural Network</i>	43
Figure 3.3. <i>Feature Subset Selection Search Space</i>	54
Figure 3.4. <i>Feature Subset Selection by Relevance Algorithm</i>	61
Figure 4.1. <i>Case Study 1: Graphical representation of FRB obtained after optimization</i> ..	65
Figure 4.2. <i>Case Study 1: Performance of RBPN during removal of input features</i>	67
Figure 4.3. <i>Case Study 1: Performance of the RBPN with different features</i>	67
Figure 4.4. <i>Case Study 1: Graphical representation of the FRB obtained after simplification</i>	67
Figure 4.5. <i>Case Study 1: Textual representation of the FRB obtained after simplification</i>	68
Figure 4.6. <i>Case Study 1: Summary of Classification results of FRULEX</i>	69
Figure 4.7. <i>Case Study 2: Graphical representation of the FRB obtained after optimization</i>	73
Figure 4.8. <i>Case Study 2: Performance of RBPN during removal of input features</i>	75
Figure 4.9. <i>Case Study 2: Performance of the RBPN with different features</i>	75
Figure 4.11. <i>Case Study 2: Textual Representation of the FRB obtained after simplification</i>	76
Figure 4.10. <i>Case Study 2: Graphical representation of the FRB obtained after simplification</i>	76
Figure 4.12. <i>Case Study 2: Summary of Classification results of FRULEX</i>	77
Figure 4.13. <i>Case Study 3: Graphical representation of the FRB obtained after optimization</i>	81
Figure 4.14. <i>Case Study 3: Performance of RBPN during removal of input features</i>	83
Figure 4.15. <i>Case Study 3: Performance of the RBPN with different features</i>	83
Figure 4.16. <i>Case Study 3: Graphical Representation of the FRB obtained after simplification</i>	83

Figure 4.17. <i>Case Study 3: Textual representation of the FRB obtained after simplification</i>	84
Figure 4.18. <i>Case Study 3: Summary of Classification results of FRULEX</i>	85
Figure 4.19. <i>Case Study 4: Graphical representation of the FRB obtained after optimization</i>	89
Figure 4.20. <i>Case Study 4: Performance of RBPN during removal of input features</i>	90
Figure 4.21. <i>Case Study 4: Performance of the RBPN with different features</i>	91
Figure 4.22. <i>Case Study 4: Textual representation of the FRB obtained after simplification</i>	91
Figure 4.23. <i>Case Study 4: Graphical representation of the FRB obtained after simplification</i>	92
Figure 4.24. <i>Case Study 4: Summary of Classification results of FRULEX</i>	92

LIST OF TABLES

Table 2.1. <i>Rule Quality Assessment [Andrews and Geva, 1995]</i>	33
Table 2.2. <i>Complexity of the M-of-N algorithm [Towell and Shavlik, 1993]</i>	35
Table 4.1. <i>Description of Case Studies</i>	62
Table 4.2. <i>Case Study 1: Classes</i>	63
Table 4.3. <i>Case Study 1: Features and Feature values</i>	63
Table 4.4. <i>Case Study 1: Results of the 10-fold cross validation after initialization</i>	64
Table 4.5. <i>Case Study 1: Results of the 10-fold cross validation after optimization</i>	65
Table 4.6. <i>Case Study 1: Results of 10-fold cross validation after sorted and neighbor search</i>	66
Table 4.7. <i>Case Study 1: Results of the 10-fold cross validation after simplification</i>	66
Table 4.8. <i>Case Study 1: Summary of Classification results of FRULEX</i>	68
Table 4.9. <i>Case Study 1: Statistical and Neural Classifiers</i>	69
Table 4.10. <i>Case Study 1: Crisp Rule-Based Classifiers</i>	69
Table 4.11. <i>Case Study 1: Fuzzy Rule-Based Classifiers</i>	70
Table 4.12. <i>Case Study 2: Classes</i>	71
Table 4.13. <i>Case Study 2: Features and Feature values</i>	71
Table 4.14. <i>Case Study 2: Results of the 10-fold cross validation after initialization</i>	72
Table 4.15. <i>Case Study 2: Results of the 10-fold cross validation after optimization</i>	73
Table 4.16. <i>Case Study 2: Results of 10-fold cross validation after sorted and neighbor search</i>	74
Table 4.17. <i>Case Study 2: Results of the 10-fold cross validation after simplification</i>	74
Table 4.18. <i>Case Study 2: Summary of Classification results of FRULEX</i>	76
Table 4.19. <i>Case Study 2: Statistical and Neural Classifiers</i>	77
Table 4.20. <i>Case Study 2: Crisp Rule-Based Classifiers</i>	77
Table 4.21. <i>Case Study 2: Fuzzy Rule-Based Classifiers</i>	78
Table 4.22. <i>Case Study 3: Classes</i>	79
Table 4.23. <i>Case Study 3: Features and Feature values</i>	79
Table 4.24. <i>Case Study 3: Results of 10-fold cross validation after initialization</i>	80
Table 4.25. <i>Case Study 3: Results of 10-fold cross validation after optimization</i>	81
Table 4.26. <i>Case Study 3: Results of 10-fold cross validation after sorted and Neighbor Search</i>	82
Table 4.27. <i>Case Study 3: Results of 10-fold cross validation after simplification</i>	82
Table 4.28. <i>Case Study 3: Summary of Classification results of FRULEX</i>	85
Table 4.29. <i>Case Study 3: Statistical and Neural Classifiers</i>	85
Table 4.30. <i>Case Study 3: Crisp Rule-Based Classifiers</i>	86
Table 4.30. <i>Case Study 3: Fuzzy Rule-Based Classifiers</i>	86
Table 4.32. <i>Case Study 4: Classes</i>	87
Table 4.33. <i>Case Study 4: Features and Feature values</i>	87
Table 4.34. <i>Case Study 4: Results of the 10-fold cross validation after initialization</i>	88

Table 4.35. <i>Case Study 4: Results of the 10-fold cross validation after optimization.....</i>	88
Table 4.36. <i>Case Study 4: Results of 10-fold cross validation after sorted and neighbor search</i>	89
Table 4.31. <i>Case Study 4: Results of the 10-fold cross validation after simplification.....</i>	90
Table 4.38. <i>Case Study 4: Summary of Classification results of FRULEX.....</i>	92
Table 4.39. <i>Case Study 4: Statistical and Neural Classifiers.....</i>	93
Table 4.40. <i>Case Study 4: Crisp Rule-Based Classifiers.....</i>	93
Table 4.41. <i>Case Study 4: Fuzzy Rule-Based Classifiers</i>	93

INTRODUCTION

1.1 Background

System modeling is the task of modeling the operation of an unknown system from a combination of prior knowledge and measured input-output data. It plays a very important role in many areas such as pattern classification, control, medical diagnosis, etc. Through the simulated system model, one can easily understand the underlying properties of the unknown system and handle it properly. To model a complex system, usually the only available information is a collection of imprecise data; it is called *fuzzy modeling*, whose objective is to extract a model in the form of fuzzy inference rules. Zadeh proposed the fuzzy set theory to deal with such kind of uncertain information and many researchers have pursued research on fuzzy modeling, however, this approach lacks a definite method to determine the number of fuzzy rules required and the membership functions associated with each rule. Also, it lacks an effective learning ability to refine these functions to minimize output errors. Another approach using neural networks was proposed, which like fuzzy modeling, is considered to be a universal approximator. This approach has advantages of excellent learning capability and high precision. However, the most important weakness of neural networks is that they are like black boxes. Knowledge acquired by a neural network is encoded in its topology, in the weights on the connections and in the activation functions of the hidden and output nodes. Also, it usually suffers from slow convergence, local minima, and low understandability. Considerable work has been done to integrate neural networks with fuzzy modeling, resulting in *neuro-fuzzy modeling* approach.

Knowledge discovery and data mining have been become very important in our society where the amount of data double almost every year. In these complex databases, much information is often hidden as trends, dependencies and relationships. Data mining is the process of acquiring knowledge, such as patterns, associations, and significant structures from data, and transforming this information into a compact and interpretable decision system. It provides the users of neural networks with an explanation capability, which makes it possible for the user to validate the internal logic of the system decision, especially in medical diagnosis. Acquiring knowledge from human experts, by knowledge engineers, while designing the knowledge base of traditional expert systems, may be difficult and time consuming. Extracting knowledge in the form of If-Then rules from numerical input–output data makes knowledge acquisition much easier. This will be helpful especially in domains where there is large data but not many experts.

Here are some reasons of extracting fuzzy rules instead of crisp rules:

- Using crisp rules, ONLY one class label is identified as the correct one, thus providing a black-and-white picture where the user needs additional information. (For medical diagnosis, we may wish to quantify “how severe the disease is” with numbers in $[0, 1]$. For pattern classification, we need to know “how typical this pattern is”.)
- The interest in using fuzzy rule-based systems arises from the fact that they provide a good platform to deal with uncertain, noisy, imprecise or incomplete information which is often handled in any human-cognition system.
- Reliable crisp rules may reject some cases as unclassified.
- Using the number of errors given by the crisp rules for the cost function, makes optimization difficult since ONLY non-gradient optimization methods may be used.

1.2 Problem Statement

For complex and high-dimensional classification tasks, data-driven identification of such classifiers has to deal with two structural problems, which are the *effective initial partitioning of the input domain* and the *selection of the relevant features*. Therefore, the

identification of fuzzy classifiers is a challenging topic. Also, linguistic *interpretability* is an important aspect of these classifiers. Fuzzy logic helps improving the interpretability of knowledge-based classifiers through its semantics that provide insight in the classifier internal structure. However, Fuzzy logic is not a guarantee for interpretability. That is, real effort must be made to keep the resulting classifier interpretable. Two main approaches are followed in the literature. First, select a low number of input variables in order to make a compact classifier. Second, make a large set of possible rules, by using all inputs, then make a useful selection out of these rules. Often genetic algorithm is applied for this rule-selection process.

Most neuro-fuzzy approaches for rule extraction are usually limited to the description of new algorithms, presenting only a partial solution to the problem of knowledge extraction from data. That is, Most of these approaches pursue accuracy as ultimate goal and take no care about the interpretability of the extracted knowledge. Control of the tradeoff between interpretability and accuracy, optimization of the linguistic variables and final rules, and estimation of the reliability of rules are most never discussed.

Common initializations methods such as grid-type partitioning [Castellano *et al.*, 2002], tree-type partitioning [Kubat, 1998] and *rule generation on extrema initialization*, result in complex and non-interpretable initial models. As a result, the rule base simplification and reduction step become computationally demanding. Thus for high-dimensional systems, the initialization step of the fuzzy model becomes very significant. For this purpose fuzzy clustering or similar covariance based initialization techniques were put forward. Therefore, gaining interpretability is the main advantage derived from the initialization step.

This thesis focuses on these problems by presenting a new neuro-fuzzy approach for extracting fuzzy classifiers from labeled data, where each instance given to the classifier is associated with one out of a limited number of predefined classes. The proposed approach used a specified type of neural networks, which is known as Rapid Back Propagation Neural Networks, solving both the interpretability and simplicity problems. These classifiers can be used for medical diagnosis and pattern classification. The new approach is called **FRulex** (**F**uzzy **R**ules extractor).

1.3 Previous Work

In recent years, a large number of different methods for extracting rules have been proposed in the literature ([Andrews *et al.*, 1995] and [Mitra and Hayashi, 2000] provide rich sources of references). Mitra, [Mitra and Hayashi, 2000], classified the different methods into fuzzy, neural, and neuro-fuzzy approaches. Let us touch upon some of the fuzzy and neural approaches before start focusing on neuro-fuzzy approaches.

- Taha and Ghosh [Taha and Ghosh, 1996a,b] have extracted rules along with certainty factors from trained feedforward networks. Input features are discretized and a linear programming problem is formulated and solved. A greedy rule evaluation mechanism is used to order the extracted rules on the basis of three performance measures that are soundness, completeness, and false-alarm. A method of integrating the output decisions of both the extracted rule base and the corresponding trained network is described, with a goal of improving the overall performance of the system.
- Kantardzic and Elmaghraby [Kantardzic and Elmaghraby, 1997] have developed an experimental algorithm for logical interpretation of a neuron's computational model without heuristic approximations. The algorithm is based on the general logical function *NofM* which covers all possible logical combinations of node inputs on the level of one class of input's weight factors. The algorithm gives us additional possibilities to analyze *don't-care* states in a logical model which correspond to loose input-output training sets in neural networks.
- Castro, Mantas, and Benitez [Castro *et al.*, 2002] have presented a procedure to represent the action of an ANN in terms of fuzzy rules. This method extends another one, [Benitez *et al.*, 1997], which was proposed previously. The main achievement of the new method is that the fuzzy rules obtained are in agreement with the domain of the input variables. In order to keep the equality relationship between the ANN and a corresponding fuzzy rule-based system, a new operator has been presented.
- Tresp, Hollatz and Ahmed [Tresp *et al.*, 1993] describe a method for extracting rules from Gaussian Radial Basis Function (RBF) network.

- Berthold and Huber [Berthold and Huber, 1995] describe a method for extracting rules from a specialized local function network, the Rectangular Basis Function (RecBF) network.
- Abe and Lan [Abe and Lan, 1995] describe a recursive method for constructing hyper-boxes and extracting fuzzy rules from them and apply it to pattern classification.
- Duch *et al* [Duch *et al.*, 1999, 2001] describe a method for extraction, optimization and application of sets of fuzzy rules from ‘soft trapezoidal’ membership functions.
- Lapedes and Faber [Lapedes and Faber, 1987] give a method for constructing locally responsive units using pairs of axis-parallel logistic sigmoid functions. Subtracting the value of one sigmoid from the other one will construct such local response region. They did not however offer a training scheme for networks constructed of such units. Geva and Sitte [Geva and Sitte, 1994] describe a parameterization and training scheme for networks composed of such sigmoid based hidden units. Andrews and Geva [Andrews and Geva, 1995, 1999] propose a method to extract and refine crisp rules from these networks.

Recently, neuro-fuzzy approaches for rule extraction have attracted a lot of attention [Lin *et al.*, 1997], [Farag *et al.*, 1998], [Rojas *et al.*, 2000], [Wu *et al.*, 2000], [Wu *et al.*, 2001] and [Castellano *et al.*, 2000a, 2002]. In general, this approach involves two major phases, *structure identification* and *parameter identification*. Fuzzy modeling and neural network techniques are usually used in the two phases. As a result, neuro-fuzzy modeling gains the benefits of fuzzy modeling and neural networks, which are adaptability, quick convergence and high accuracy. Fuzzy rules are discovered from the set of given input-output data in the phase of *structure identification*. For the purpose of higher precision, the fuzzy rules are then optimized by a learning algorithm of neural networks in the second phase of *parameter identification*. Neural network can be used for numeric inference, or refined fuzzy rules can be extracted from the networks for symbolic reasoning.

For structure identification, Lin *et al.*, [Lin *et al.*, 1997], proposed a method of fuzzy partitioning to extract initial fuzzy rules, but it is hard to decide the locations of cuts and too much time is needed to select best cuts. Castellano *et al.*, [Castellano *et al.*, 2002], used grid partitioning to generate human-understandable knowledge from data, but it encounters the

problem of an exponential increase in the number of rules when the number of inputs is large. For instance, a fuzzy model with 10 inputs and 2 MFs would result in $2^{10} = 1024$ fuzzy if-then rules, which is very large. Kubat [Kubat, 1998] used tree partitioning to initialize Radial-Basis Function Networks. The tree partition relieves the problem of the exponential increase in the number of rules. However, more MFs for each input are needed to define these fuzzy regions, and these MFs do not usually bear clear linguistic meanings such as “small”, “big”, and so on. Farag, [Farag *et al.*, 1998], presents a neuro-fuzzy approach capable of handling both quantitative and qualitative knowledge. This approach used Kohonen’s self-organizing feature map algorithm.

For parameter identification, most approaches, including [Lin *et al.*, 1997] and [Castellano *et al.*, 2002] used gradient descent back propagation to refine parameters of the system. Farag, [Farag *et al.*, 1998], used a multiresolutional dynamic genetic algorithm (GA) for tuning of membership functions of the extracted linguistic fuzzy rules.

Some approaches have been proposed to obtain interpretable knowledge by neuro-fuzzy learning ([Nauck *et al.*, 1996, 1999], [Castellano *et al.*, 2000b] and [Lozowski and Zurada, 2000]). In ([Nauck *et al.*, 1996], the authors propose NEFCLASS, an approach that creates fuzzy systems from data by applying a heuristic data-driven learning algorithm that constraints the modifications of the fuzzy set parameters to take the semantical properties of the underlying fuzzy system into account. However, a good interpretation cannot always be guaranteed, especially for high-dimensional problems. Hence, in [Nauck *et al.*, 1999] the NEFCLASS algorithm is added with interactive strategies for pruning rules and variables so as to improve readability. This approach shows good results, but it results in a long interactive process that cannot extract rules automatically but requires the user to supervise and interpret the learning process in all its stages.

1.4 Organization of Thesis

The thesis is organized as follows.

- **Chapter 2** gives an overview about artificial neural networks, especially rapid back propagation neural networks, fuzzy logic, and neuro-fuzzy hybridization, which is the

most well known methodology in soft computing. In addition, it gives an exhaustive survey on rule extraction methods and an evaluation of them.

- **Chapter 3** introduces *FRULEX* fuzzy rules extraction approach. First it reviews the general algorithm. Then it discusses Self-Constructing Rule Generator (SCRG) method. Next it discusses the back propagation gradient-descent learning algorithm. Finally, it presents the method used to simplify the fuzzy rules extracted.
- **Chapter 4** gives an evaluation of the *FRULEX* approach, and the experimental results performed to evaluate the effective of the different parts of the new approach. It provides graphical and textual representations of the fuzzy rule bases extracted for each dataset using *MATLAB™ Fuzzy Toolbox*.
- **Chapter 5** summaries the major features of this thesis and proposes some research points that can be investigated for future work.
- **Appendix A** lists the set of abbreviations used the thesis.
- **Appendix B** illustrates the flow chart of the *FRULEX* approach using *Rational™ Rose*.
- **Appendix C** shows the class diagram for the implementation of the *FRULEX* approach using *Rational™ Rose*.

RULE EXTRACTION BACKGROUND

2.1 Overview of Artificial Neural Networks

Neural networks are of particular interest because they offer a means of efficiently modeling large and complex problems. Neural networks may be used in classification problems (where the output is a categorical variable) or for regressions (where the output variable is continuous). A detailed discussion about neural networks is provided in [Jang *et al.*, 1998].

2.1.1 Introduction to Artificial Neural Network

An artificial neural network can be defined as an information processing system consisting of many processing elements joined together in a structure inspired by the cerebral cortex of the brain. The processing elements considered in the definition of ANN are usually organized in a sequence of layers, with full connections between layers. Typically, there are three (or more) layers: an **input layer** where data are presented to the network through an input buffer, an **output layer** with a buffer that holds the output response to a given input, and one or more intermediate or **hidden layers**. (See Figure 2.1)

The operation of an artificial neural network involves two processes: **learning** and **recall**. Learning is the process of updating the connection weights in response to external stimuli presented at the input buffer. The network “learns” in accordance with a learning rule governing the adjustment of connection weights in response to learning examples

applied at the input and output buffers. Recall is the process of accepting an input and producing a response determined by the geometry and synaptic weights of the network.

2.1.1.1 Processing Units

Each processing element (or neuron) receives input (signal) from neighbors or external sources and use this to compute an output signal which is propagated to other units. Apart of this processing, a second task is the adjustment of the weights. The system is inherently parallel in the sense that many units can carry out their computations at the same time.

During operation, units can be updated either **synchronously** or **asynchronously**. With synchronous updating, all units update their activation simultaneously; with asynchronous updating, each unit has a (usually fixed) probability of updating its activation at a time t , and usually only one unit will be able to do this at a time.

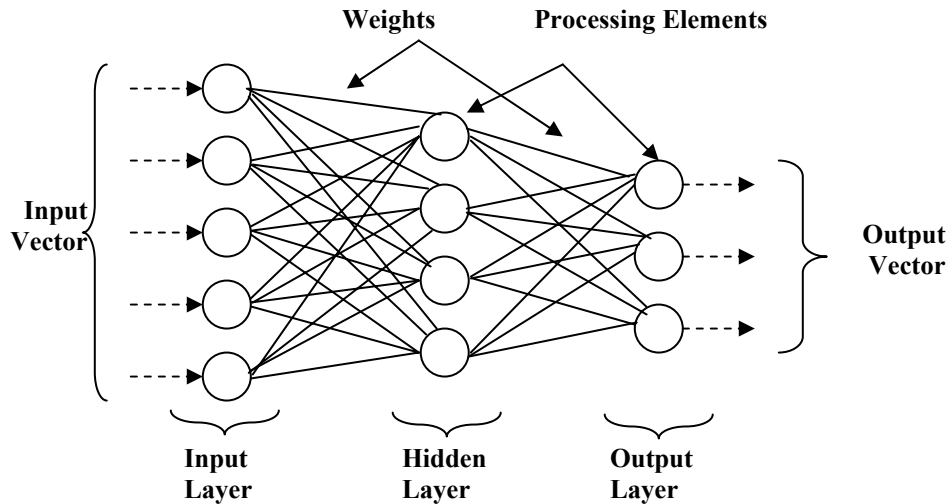


Figure 2.1. Artificial Neural Network

2.1.1.2 Activation and Output Functions

Two functions determine the way signals are processed by neurons. *The activation function* determines the total signal neuron receives. In most cases, a linear combination of the incoming signals is used. For neuron i connected to neurons j (for $j = 1, \dots, N$) sending signals x_j with the strength of the connections w_{ij} the total activation signal I_i is

$$I_i(x) = \sum_{j=1}^N w_{ij}(t)x_j \quad (2.1)$$

The second function determining neuron's signal processing is *the output function* $o(I)$. These two functions together determine the values of the neuron outgoing signals. The total function acts in the N-dimensional *input space*, called also *the parameter space*. The composition of these two functions is called the *transfer function* $o(I(x))$. The activation and the output functions of the input and the output layers may be of different type than those of the hidden layer, in particular frequently linear functions are used for inputs and outputs and non-linear output functions for hidden layers.

2.1.1.2.1 *Non-local Transfer Functions*

The first neural network models proposed in the 40-ties by McCulloch and Pitts [McCulloch and Pitts, 1943] were based on the logical processing elements of the threshold type. The output function of the logical elements is of *the step function* type, and is known also as the Heaviside $\theta(x)$ function: it is 0 below the threshold value and 1 above it. The greatest advantage of the logical elements is the speed of computations.

Classification regions of the logical networks are of the hyper-plane type rotated by the w_{ij} coefficients. An intermediate multi-step type of functions between continuous sigmoidal functions and step functions are sometimes used, with a number of thresholds. Instead of the step function, semi-linear functions were used and later generalized to *the sigmoidal functions*, leading to the *graded response neurons*:

$$\sigma(x; s) = \frac{1}{1 + e^{-sx}} \quad (2.2)$$

The constant s determines the slope of the sigmoid function around the linear part. The arcos tangent or the *hyperbolic tangent function* may also replace this function:

$$\tanh(x; s) = \frac{e^{sx} - e^{-sx}}{e^{sx} + e^{-sx}} \quad (2.3)$$

Other sigmoid functions may be useful to speed up computations:

$$s_1(x; s) = \theta(x) \frac{x}{x + s} - \theta(-x) \frac{x}{x - s} \quad (2.4)$$

$$s_2(x; s) = \frac{\sqrt{1 + s^2 x^2} - 1}{sx} \quad (2.5)$$

where $\theta(x)$ is a step function Sigmoid functions have non-local behavior, i.e. they are non-zero in infinite domain. Sigmoid output functions smooth out many shallow local minima in the total output functions of the network.

For classification problems this is very desirable, but for general mappings it limits the precision of the adaptive system. For sigmoid functions, powerful mathematical results exist showing that a universal approximator may be built from only single layer of processing elements. Figure 2.2 illustrates how the decision regions for classification are formed.

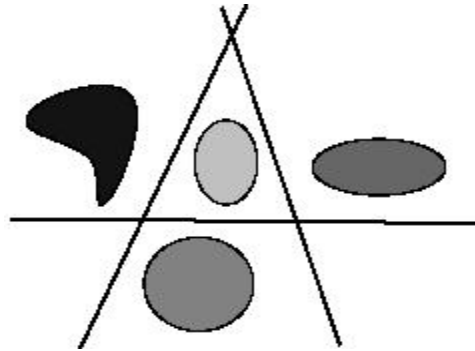


Figure 2.2. *Decision regions formed using sigmoid processing functions*

2.1.1.2.2 Local Transfer Functions

From the point of view of a neural network used as a classification device one can either divide the total parameter space into regions of classification using non-local functions or set up local regions around the data points. A few attempts were made to use localized functions in the neural network; Moody and Darken [Moody and Darken, 1989] used locally tuned processing units to learn real-valued mappings and classifications in a learning method combining self-organization and supervised learning. They have selected

locally tuned units to speed up the learning process of back propagation networks. Bottou and Vapnik [Bottou and Vapnik, 1992] showed the power of local training algorithms in a more general way. Although the processing power of neural networks based on non-local processing, units does not depend strongly on the type of neuron processing functions such is not the case for localized units. Gaussian functions are perhaps the simplest but not the least expensive to compute.

2.1.1.3 Network Topologies

Network topologies are divided, as provided in [Jang *et al.*, 1998], into two categories:

- **Feed forward network** where the data flow from input to output units is strictly feed forward. The data processing can extend over multiple (layers of) units, but no feedback connection are present, that is, connections extending from outputs of units to inputs of units in the same layer or previous layers.
- **Recurrent network** that can contain feedback connections. On contrary to feed forward networks, the dynamical properties of the network are important. In some case, the activation values of the units undergo a relaxation process such that the network will evolve to a stable state in which these activations do not change anymore. In other applications, the change of the activation values of the output neurons are significant, with the dynamical behavior constitutes the output of the network.

2.1.1.4 Training of Artificial Neural Networks

The learning techniques can be classified, as provided in [Jang *et al.*, 1998], into:

- **Supervised learning** or **Associative learning** in which the network is trained by providing it with input and matching output pattern. These input-output pairs can be provided by an external teacher, or by the system, which contains the network.
- **Unsupervised learning** or **Self-organization** in which an (output) unit is trained to respond to clusters of pattern within the input. In this paradigm the system is supposed to discover statistically salient features of the input population. Unlike the supervised

learning paradigm, there is no a priori set of categories into which the patterns are to be classified; rather the system must develop its own representation of the input stimuli.

2.1.1.5 Learning Algorithms

There are different learning algorithms. The most common learning algorithms, as discussed in [Jang *et al.*, 1998], are:

- **Hebbian unsupervised learning** where a connection weight on an input path to a processing element is incremented if the input is high and the desired output is high. This is analogous to the biological process, in which a neural pathway is strengthened each time it is used. A detailed discussion is provided in [Parker, 1987].
- **Delta-rule supervised learning** (sometimes called mean square error learning) where the error (difference between the desired output and the actual output) is minimized using a least-squares process. Back propagation is the most common implementation of Delta-rule learning and probably is used in at least 75% of ANN applications, such as pattern recognition, signal processing, data compression, and automatic control.
- **Competitive unsupervised learning** where the processing elements compete; only the processing element yielding the strongest response to a given input can modify itself, becoming more like the input. In all cases, the final values of the weighting functions constitute the “memory” of the ANN.

2.1.2 Local Function Neural Networks

Locally tuned and overlapping receptive fields are well-known structures that have been studied in regions of the cerebral cortex, the visual cortex, and others. In the field of Artificial Neural Networks, (ANN's), there are several types of networks that utilize units with local response characteristics (LRUs) to solve real-world problems in the field of pattern classification, function approximation, and medical diagnosis. We will discuss the advantages and disadvantages of the utilization of such type of neural networks in the following two subsections.

2.1.2.1 Advantages of Local Function Networks

For the point of view of an adaptive system used as a classifier, one can either divide the total input space into regions of classification using non-local transfer functions or setup local regions around the data points. Experiments have emphasized that the use of locally-tuned processing units speed up the learning process of back propagation networks.

Andrews and Geva, [Andrews and Geva, 1995], have stated that local function networks are attractive for rule extraction for two reasons.

- **First**, it is conceptually easy to imagine how the weights of a local response unit can be converted into a symbolic rule. This obviates the necessity for exhaustive search and test strategies used by other non-LRU based rule extraction methods. Hence, the computational effort required to extract rules from LRUs is significantly less than that required using other methods.
- **Second**, because each LRU can be described by the conjunction of some range of values in each input dimension, it makes it easy to add units to the network during training such that the added unit has a meaning that is directly related to the problem domain.

2.1.2.2 Disadvantages of Local Function Networks

Andrews and Geva, [Andrews and Geva, 1995], have proved that there are also disadvantages associated with local function networks.

- **Local Nature**: By definition, the rules extracted from such networks are themselves local in nature which makes the explanation of non-local problems difficult.
- **Overlap Problem**: It is that caused by overlapped LRUs. One of the main advances of rule extraction from non-overlapped local response units is the ease with which a unit can be directly decompiled into a rule. But if the LRUs are allowed to overlap, more than one unit will show significant activation when presented with an input pattern that fell in the region of overlap. The pattern will be classified by the network, but when the individual units are decompiled into rules, these rules may not classify these patterns.

2.1.3 Architecture of Rapid Back Propagation Networks

The rapid back propagation networks are similar to radial basis function networks (RBFN) in that the hidden layer consists of a set of locally responsive units. The hidden units of the RBF network are sigmoid-based locally responsive units (LRU's) that have the effect of partitioning the training data into a set of regions, each region being represented by a single hidden layer unit. Each LRU is composed of a set of ridges, one ridge for each dimension of the input. The LRU output is the threshold sum of the activations of the ridges.

The sigmoid-based local response unit of the hidden layer of the RBP network is constructed as follows:

- In each input dimension, form a region of local response according to the equation

$$\begin{aligned}
 r(x_i; c_i, b_i, k_i) &= \sigma^+(x_i; c_i, b_i, k_i) - \sigma^-(x_i; c_i, b_i, k_i) \\
 &= \sigma(k_i, (x_i - c_i + b_i)) - \sigma(k_i, (x_i - c_i - b_i)) \\
 &= \frac{1}{1 + e^{-(x_i - c_i + b_i)k_i}} - \frac{1}{1 + e^{-(x_i - c_i - b_i)k_i}}
 \end{aligned} \tag{2.6}$$

- This construction forms an axis parallel ridge function in the i^{th} dimension of the input space, $r(x_i; c_i, b_i, k_i)$, that is almost zero everywhere except in the region between the steepest part of the two logistic sigmoid functions. (See Figure 2.3 and Figure 2.4)

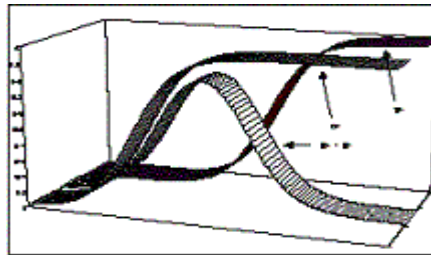


Figure 2.3. Construction of a ridge [Andrews and Geva, 1995]

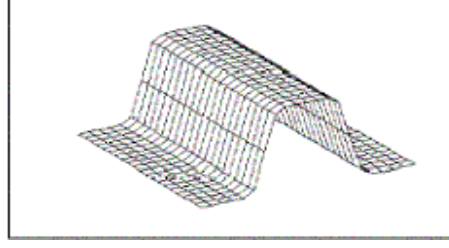


Figure 2.4. *Cylindrical Extension of a ridge* [Andrews and Geva, 1995]

- The parameters c_i , b_i , and k_i of the sigmoid functions $\sigma^+(x_i; c_i, b_i, k_i)$ and $\sigma^-(x_i; c_i, b_i, k_i)$ represent the center, breadth, and edge steepness respectively of the ridge, and x_i is the input value.
- The intersection of such N ridges, with a common center, produces a function f that represents a local peak at the point of intersection with secondary ridges extending to infinity, on either sides of the peak, in each dimension (See Figure 2.5). The function f is the sum of the N ridge functions

$$f(\underline{x}; \underline{c}, \underline{b}, \underline{k}) = \sum_{i=1}^N r(x_i; c_i, b_i, k_i) \quad (2.7)$$

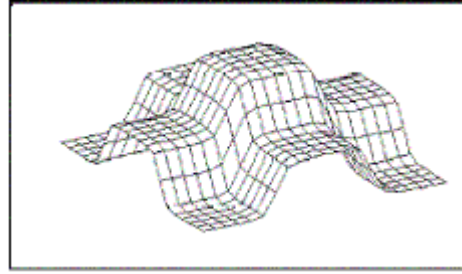


Figure 2.5. *Intersection of two Ridges* [Andrews and Geva, 1995]

- To make the function local, these component ridges must be cut off by the application of a suitable sigmoid to leave a local response region in the input space (see Figure 2.6). The function $\ell(\underline{x}; \underline{c}, \underline{b}, \underline{k})$ eliminates the unwanted regions of the radiated ridge functions.

$$\ell(\underline{x}; \underline{c}, \underline{b}, \underline{k}) = \sigma(K, f(\underline{x}; \underline{c}, \underline{b}, \underline{k}) - B) \quad (2.8)$$

Where B is selected to ensure that the maximum value of the function f , located at $\underline{x} = \underline{c}$, coincides with the center of the linear region of the output sigmoid. The parameter K determines the steepness of the output sigmoid function $\ell(\underline{x}; \underline{c}, \underline{b}, \underline{k})$.

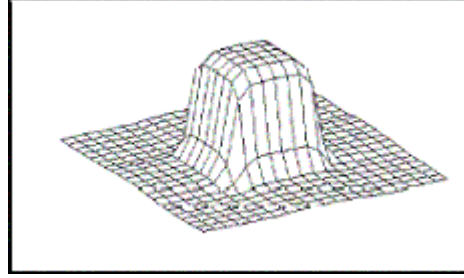


Figure 2.6. *Production of an LRU* [Andrews and Geva, 1995]

- The parameter B is set to produce appreciable activation only when each of the x_i input values lie in the ridge defined in the i^{th} dimension. The parameter K is chosen such that output sigmoid $\ell(\underline{x}; \underline{c}, \underline{b}, \underline{k})$ cuts off the secondary ridges outside the boundary of the local function. Experiment has shown that good network performance can be obtained if B is set equal to the input dimensionality, $B = N$ and K is set in the range 2-4.
- A network that is suitable for function approximation and binary classification tasks can be created with an input layer, a hidden layer of ridge functions, a hidden layer of local functions, and an output unit.
- The activation for the output unit is given as:

$$y(\underline{x}) = \sum_{j=1}^J w_j \ell(\underline{x}; \underline{c}_j, \underline{b}_j, \underline{k}_j) \quad (2.9)$$

which is a linear combination of J local response functions with centers $\underline{c}_j$, widths $\underline{b}_j$, and steepness $\underline{k}_j$. Where w_j is the output weight associated with each of the individual local response functions ℓ . (Network output is simply the weighted sum of the outputs of the local response functions.)

- For multi-class classification problems, several such networks can be combined together; one network per class, with the output class being the maximum of the activations of the individual networks, that combination is called *MCRBP Network*.

- The RBP network is trained using gradient descent on an error surface to adjust the parameters (output weights, and the individual ridge center, breadth, and edge steepness).

2.2 Overview of Fuzzy Set Theory

2.2.1 Fuzzy Sets

A classical crisp set is a collection of distinct object. The concept of a set has become one of the most fundamental notions of mathematics. Crisp set theory was founded by the German mathematician George Cantor (1845-1918). It is defined in such a way as to divide the elements of a given universe of discourse into two group members and nonmembers. Finally, a crisp set can be defined by the so-called *characteristic function*. Let U be a universe of discourse. The characteristic function $\mu_A(x)$ of a crisp set A in U is defined as:

$$\mu_A(x) = \begin{cases} 1 & \text{iff } x \in A \\ 0 & \text{iff } x \notin A \end{cases} \quad (2.10)$$

Zadeh introduced fuzzy sets [Zadeh, 1965], where a more flexible sense of membership is possible. In fuzzy sets, many degrees of membership are allowed. The degree of membership to a set is indicated by a number between 0 and 1. Hence, fuzzy sets may be viewed as an extension and generalization of the basic concepts of crisp sets.

A fuzzy set A in the universe of discourse U can be defined as a set of ordered pairs,

$$A = \{(x, \mu_A(x)) \mid x \in U\} \quad (2.11)$$

where μ_A is called the *membership function* of A and $\mu_A(x)$ is the degree of membership of x in A , which indicates the degree that x belongs to A . The membership function μ_A maps U to the membership space M , that is $\mu_A: U \rightarrow M$. When $M = \{0, 1\}$, set A is non-fuzzy and μ_A is the characteristic function of the crisp set A . For fuzzy set, the range of the membership function is a subset of the nonnegative real numbers. In most general cases, M is set to the unit interval $[0, 1]$.

2.2.2 Membership Functions

For representation of the membership functions, we can use the following functions:

- *Triangular Membership Functions*

A triangular MF, as shown in Figure 2.7 (a), is a function with 3 parameters defined by

$$triangle(x; a, b, c) = \max(\min(\frac{x-a}{b-a}, \frac{c-x}{c-b}), 0) \quad (2.12)$$

- *Trapezoidal Membership Functions*

A Trapezoidal MF, as shown in Figure 2.7 (b), is a function with 4 parameters defined by

$$trapezoid(x; a, b, c, d) = \max(\min(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}), 0) \quad (2.13)$$

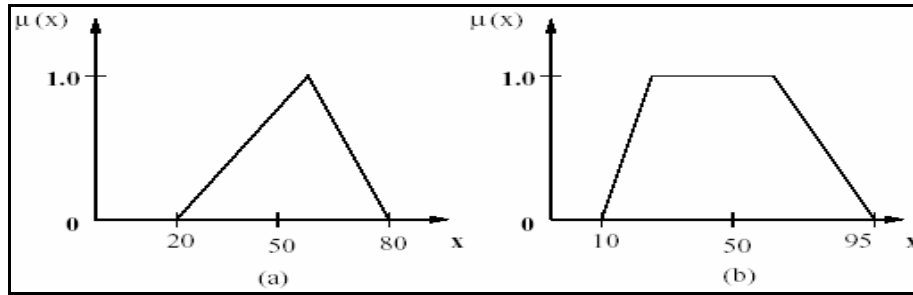


Figure 2.7. Membership Functions: (a) Triangle (b) Trapezoid [Jang *et al.*, 1998]

- *Gaussian Membership Functions*

A Gaussian MF is a function with two parameters defined by

$$gaussian(x; \sigma, c) = e^{-\frac{(x-c)^2}{\sigma}} \quad (2.14)$$

where c is the center and σ is the width of membership function.

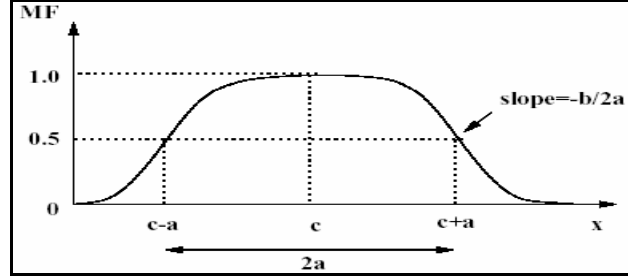


Figure 2.8. Bell Membership Function [Jang *et al.*, 1998]

- *Bell* Membership Functions

A bell MF, as shown in Figure 2.8, is a function with two parameters defined by

$$bell(x; a, b, c) = \frac{1}{1 + \left| \frac{x - c}{a} \right|^{2b}} \quad (2.15)$$

- *Sigmoidal* Membership Function

A Sigmoid MF is a function with two parameters defined by

$$sigmoid(x; k, c) = \frac{1}{1 + e^{-k(x-c)}} \quad (2.16)$$

where parameter k influences sharpness of function in the point where $a = c$. If $k > 0$ the function is open on right site, on the other hand, if $k < 0$ the function is open on left site and therefore this function can be use for describing conceptions like “very big” or “very small”. Sigmoid function is very often used in Neural Networks like activation function.

2.2.3 Fuzzy Rules and Fuzzy Reasoning

Fuzzy rules and fuzzy reasoning are the backbone of fuzzy inference systems, which are the most important modeling tool based on fuzzy set theory. They have been applied to a wide range of real-world problems, such as expert systems, pattern recognition, and data classification. A detailed discussion about fuzzy inference systems is provided in [Jang *et al.*, 1998].

2.2.3.1 Fuzzy If-Then Rules

Fuzzy if-then rules (also known as fuzzy conditional statements) are expressions of the form

$$\text{if } x \text{ is } A, \text{ then } y \text{ is } B \quad (2.17)$$

where A and B are linguistic labels defined by fuzzy sets on universe of discourse X and Y , respectively. Often “ x is A ” is called the antecedent or premise, while “ y is B ” is called the consequence or conclusion. Due to their concise form, fuzzy if-then rules are often used to capture the imprecise modes of reasoning and play an essential role in the human ability to make decisions in an environment of uncertainty and imprecision. Fuzzy if-then rules have been used extensively in both modeling and control. From another angle, due to the qualifiers on the premise parts, each fuzzy if-then rule can be viewed as a local description of the system under consideration.

2.2.3.2 Fuzzy Reasoning

Fuzzy reasoning, also known as approximate reasoning, is an inference procedure that derives conclusions from a set of fuzzy if-then rules and known facts.

2.2.4 Fuzzy Inference Systems

The fuzzy inference system [Takagi and Sugeno, 1985] is a popular computing framework based on the concepts of fuzzy set theory, fuzzy If-Then rules, and fuzzy reasoning. It has found successful applications in a wide variety of fields, such as automatic control, data classification, decision analysis, expert systems, robotics, and pattern recognition. The fuzzy inference system is also known by numerous other names, such as **fuzzy expert system**, **fuzzy model**, **fuzzy-rule-based system**, **fuzzy logic controller**, and simply **fuzzy system**. The basic structure of a fuzzy inference system, shown in Figure 2.9, consists of five functional components:

1. **Rule base**, which contains a selection of fuzzy rules.
2. **Database**, which defines the membership functions used in the fuzzy rules.

3. **Reasoning mechanism**, which performs the inference procedure upon the rules and given facts to derive a reasonable conclusion.
4. **Fuzzification interface**, which transform the crisp inputs into degrees of match with linguistic values.
5. **Defuzzification interface**, which transform the fuzzy results of the inference into a crisp output.

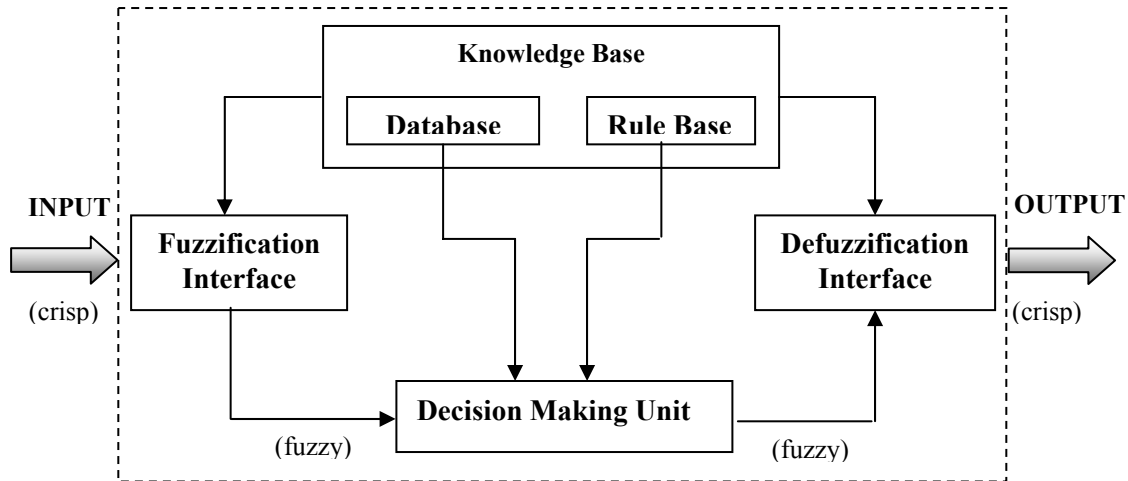


Figure 2.9. *Fuzzy Inference System* [Jang et al., 1998]

The following are the steps of fuzzy reasoning (inference operations upon fuzzy if-then rules), performed by fuzzy inference systems are:

1. Compare the input variables with the membership functions on the antecedent part to obtain the membership values of each linguistic label. (Fuzzification Step)
2. Combine (through a specific T-norm operator, usually multiplication or min) the membership values on the premise part to get firing strength (weight) of each rule.
3. Generate the qualified consequents (either fuzzy or crisp) of each rule depending on the firing strength.
4. Aggregate the qualified consequents to produce a crisp output. (Defuzzification Step)

2.2.4.1 Mamdani Fuzzy Model

The **Mamdani fuzzy inference system** was proposed as the first attempt to control a steam engine and boiler combination by a set of linguistic control rules obtained from experienced

human operators. An example of if-then rules that is daily used in our linguistic expression is

$$\text{if } \textit{pressure is high, then volume is small} \quad (2.18)$$

where *pressure* and *volume* are *linguistic variables*, *high* and *small* are *linguistic values* or *label* that are characterized by membership functions.

2.2.4.2 Tsukamoto Fuzzy Model

In the **Tsukamoto fuzzy models**, the consequent part of each fuzzy if-then rule is specified by a membership function of a step function centered at the constant. As a result, the inferred output of each rule is defined as a crisp value induced by the rule's firing strength. The overall output is taken as the weighted average of each rule's output. This fuzzy model avoids the time consumed by the defuzzification process since it aggregates each rule's output by the method of weighted average. However, this fuzzy model is not used since it is not as transparent as either Mamdani or Sugeno fuzzy models.

2.2.4.3 Sugeno Fuzzy Model

The **Sugeno fuzzy model** (also known as the **TSK fuzzy model**) was proposed by Takagi, Sugeno, and Kang in an effort to develop a systematic approach to generating fuzzy rules from an input-output data set, [Takagi and Sugeno, 1983]. Sugeno fuzzy model was implemented into the neural fuzzy system *ANFIS* [Jang, 1993].

A typical fuzzy rule in Sugeno fuzzy model has the format

$$\text{if } x \text{ is } A \text{ and } y \text{ is } B, \text{ then } z = f(x,y) \quad (2.19)$$

where *A* and *B* are fuzzy sets in the antecedent; $z = f(x, y)$ is a crisp function in the consequent part. Usually, $f(x, y)$ is a polynomial in the input variables *x* and *y*, but it can be any other functions that can appropriately describe the output of the system within the fuzzy region specified by the antecedent part of the rule.

When $f(x, y)$ is a first-order polynomial, we have the **first-order** Sugeno fuzzy model. When f is a constant, we then have the **zero-order** Sugeno fuzzy model, which can be

viewed either as a special case of the Mamdani fuzzy inference system where each rule's consequent part is specified by a fuzzy set, or a special case of Tsukamoto's fuzzy model. Moreover, a zero-order Sugeno fuzzy model is functionally equivalent to a radial basis function network under certain minor constraints. By using Takagi and Sugeno's fuzzy if-then rule, we can describe the resistant force on a moving object as follow:

$$\text{if } velocity \text{ is } high \text{ then } force = k * (velocity)^2 \quad (2.20)$$

where *high* in the premise part is a linguistic label characterized by an appropriate membership function. However, the consequent part is described by a non-fuzzy equation of the input variable, *velocity*.

2.2.4.4 Overview of Input Space Partitioning

It should be clear that the antecedent of a fuzzy rule defines a local fuzzy region, while the consequent describes the behavior within that region via various constituents. The consequent constituent can be a consequent MF (Mamdani and Tsukamoto fuzzy models), a constant value (zero-order Sugeno fuzzy model), or a linear equation (**first-order** Sugeno fuzzy model). Different consequent constituents result in different fuzzy inference systems, but their antecedents are always the same. Therefore, the following discussion of methods of partitioning input spaces to form the antecedents of fuzzy rules is applicable to all three types of fuzzy inference systems.

- **Grid Partition:** Figure 2.10 (a) illustrates a typical grid partition in a two-dimensional input space. This partition method is often chosen in designing a fuzzy controller, which usually involves only several state variables as the inputs to the controller. This partition strategy needs only a small number of MFs for each input. However, it encounters problems when we have a moderately large number of inputs. For instance, a fuzzy model with 10 inputs and 2 MFs would result in $2^{10} = 1024$ fuzzy if-then rules, which is very large. Grid partition is used by Castellano *et al.* [Castellano *et al.*, 2002] to generate human-understandable knowledge from data.

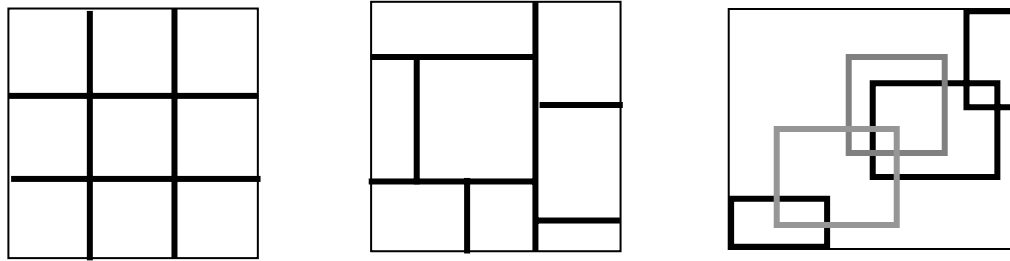


Figure 2.10. *Partitioning Methods (a) grid partition; (b) tree partition; (c) scatter partition [Jang et al., 1998]*

- **Tree Partition:** Figure 2.10 (b) shows a typical tree partition, in which each region can be uniquely specified along a corresponding decision tree. The tree partition relieves the problem of an exponential increase in the number of rules. However, more MFs for each input are needed to define these fuzzy regions, and these MFs do not usually bear clear linguistic meanings such as “small”, “big”, and so on. Tree partition is used by Kubat [Kubat, 1998] to initialize Radial-Basis Function Networks.
- **Scatter Partition:** As shown in Figure 2.10 (c), by covering a subset of the whole input space that characterizes a region of possible occurrence of the input vectors, the scatter partition can also limit the number of rules to a reasonable amount. However, the scatter partition is usually dictated by desired input-output data pairs. This makes it hard to estimate the overall mapping directly from the consequent of each rule’s output. Scatter partition is used by Abe and Lan [Abe and Lan, 1995] to extract fuzzy rules directly from numerical data and apply them to pattern classification.

2.3 Overview of Neuro-Fuzzy and Soft Computing

The following sections focus on the basic concepts and rationale of integrating fuzzy logic and neural networks into a working functional system. This happy marriage of the techniques of fuzzy logic system and neural networks suggest the novel idea of transforming the burden of designing fuzzy logic control and decision systems to the training and learning of connectionist neural networks.

2.3.1 Soft Computing

Zadeh [Zadeh, 1994] defines soft computing as a collection of methodologies that works synergistically and provides, in one form or another, flexible information processing systems for handling real life ambiguous situations. Its aim is to exploit the tolerance for partial truth, uncertainty, approximate reasoning, and imprecision in order to achieve robustness, and low-cost solutions. The guiding principle is to design methods of computation that lead to an acceptable solution at low cost by seeking for an approximate solution to an imprecisely/precisely formulated problem.

Soft computing consists of several computing paradigms, including fuzzy logic (FL), artificial neural networks (ANN's), genetic algorithms (GA's), and rough sets. Each of these constituents has its own strength. The integration of these constituents forms the core of soft computing; this integration allows soft computing to incorporate human knowledge effectively, to deal with imprecision, partial truth, and uncertainty, and to adapt to changes in environment for better performance.

2.3.2 General Comparisons of Fuzzy Systems and Neural Networks

Both Fuzzy systems and neural networks are dynamic, parallel processing systems. They are both able to improve the intelligence of systems, working in uncertain, imprecise, and noisy environments. Although fuzzy system and neural networks are formally similar, there are also significant differences between them.

Neural networks have a large number of highly interconnected processing elements, which demonstrate the ability to learn and generalize from training pattern or data. Fuzzy system, on the other hand, base their decisions on inputs in the form of linguistic variable derived from membership functions which are formulas used to determine the fuzzy set to which a value belongs and the degree of membership in that set. Fuzzy systems deal with imprecision, approximate reasoning, and computing with words. Jang and Sun [Jang and Sun, 1993] have shown that fuzzy systems are functionally equivalent to a class of radial basis function (RBF) networks, based on the similarity between the local receptive fields of the network and the membership functions of the fuzzy system.

2.3.3 *Different Neuro-Fuzzy Hybridizations*

Fuzzy logic and neural networks are complementary technologies. A promising approach to obtain the benefits of both fuzzy system and neural networks and to solve their respective problems is to combine them into an integrated system. Integrated system can learn and adapt. They learn new associations, new patterns, and new functional dependencies. Mitra and Hayashi [Mitra and Hayashi, 2000] have characterized the efforts at merging these two technologies into three categories:

- Neural Fuzzy System (NFS): the use of neural networks as tools in fuzzy model, as applied in [Nauck *et al.*, 1996].
- Fuzzy Neural Network (FNN): fuzzification of conventional neural network model.
- Fuzzy-neural hybrid system: incorporating fuzzy technologies and neural networks into hybrid systems. Both fuzzy techniques and neural networks play a key role in hybrid system. They do their own job in serving different functions in the system.

2.3.4 *Techniques of Integrating Neuro-Fuzzy Models*

Pal *et al* [Pal *et al.*, 1996] have classified the neuro-fuzzy integration methodologies as follows, Note that classes 1-3 related to FNN, while class 4 refers to NFS.

- **Incorporating fuzziness into the neural network framework:** fuzzifying the input data, assigning fuzzy labels to the training samples, possibly fuzzifying the learning procedure, and obtaining neural network outputs in terms of fuzzy sets.
- **Changing the basic characteristics of the neurons:** neurons are designed to perform various operations used in fuzzy set theory (like fuzzy union, intersection, aggregation) instead of the standard multiplication and addition operations.
- **Using measures of fuzziness as the error or instability of a network:** the fuzziness or uncertainty measures of a fuzzy set are used to model the error or instability or energy function of the neural network-based system.
- **Making the individual neurons fuzzy:** the input and output of the neurons are fuzzy sets and the activity of the networks involving the fuzzy neurons is also a fuzzy process.

2.3.5 Neural Fuzzy Systems

Neural fuzzy systems aim at providing fuzzy systems with the kind of automatic tuning methods typical of neural networks but without altering their functionality (e.g., fuzzification, defuzzification, inference engine, and fuzzy logic base).

Neural networks are used in augmenting numerical processing of fuzzy sets, such as membership function elicitation and realization of mapping between fuzzy set that is utilized as fuzzy rules. Since neural fuzzy systems are inherently fuzzy logic systems, they are mostly used in control application and classification.

Usually for an NFS, it is easy to establish a one-to-one correspondence between the network and the fuzzy system. In other words, the NFS architecture has distinct nodes for antecedent clauses, conjunction operators, and consequent clauses. An NFS should be able to learn linguistic rules and/or membership functions, or optimize existing ones. There are two possibilities: The system starts without rules, and creates new rules until the learning problem is solved. Creation of a new rule is triggered by a training pattern, which is not sufficiently covered by the current rule base. The other possibility is that, the system starts with all rules that can be created due to the partitioning of the input space and deletes insufficient rules from the rule base based on an evaluation of their performance.

2.4 Evaluation Criteria for neuro-fuzzy Approaches

Andrews *et al.* [Andrews *et al.*, 1995] have provided six different evaluation criteria for rule extraction algorithms. A brief discussion of each is shown below:

2.4.1 Computational Complexity

A universal requirement of any algorithm is its efficiency. The efficiency of an algorithm is usually measured by the number of simple calculations required for performing the given task (time complexity) and the amount of storage space used (space complexity). The time complexity of a rule-extraction algorithm, depending on the method used for rule extraction, correlate to the size of the underlying ANN, i.e. the number of layers, neurons per layer, and connections, as well as to the number of training examples, input attributes and values per input attribute. Time complexity is the important factor when estimating the

efficiency of a method, whereas space complexity plays only a secondary role. In any case an algorithm with a low computational complexity is desirable.

2.4.2 *Quality of the Extracted Rules*

The rule quality is one of the most important evaluation criteria for rule extraction algorithms.

- The *accuracy* of extracted rules describes their ability to correctly classify examples of a domain not used for the training of the network (test set). Thus, the accuracy of a rule system is a measure of the generalization performance of the extracted rules.
- The *fidelity* of a rule system describes its ability to mimic the behavior of the ANN when applied to training and testing examples. A rule system with high fidelity captures all information embodied in the ANN; it correctly classifies all training examples and classifies unseen examples in the same way as the ANN.
- The number of extracted rules and the number of antecedents per rule often indicate the *comprehensibility* of a rule system.

2.4.3 *Translucency*

Rule extraction algorithms can be divided into 3 categories according to the degree to which the underlying ANN is used:

- **Decompositional Approach** This approach considers only the internal structure of the networks, i.e., rules are extracted by directly analyzing numerical values of the network such as activation values of hidden, and output neurons, and weights of connections between them. Often rules are extracted for each hidden and output neuron separately and the rule system for the whole network is derived from these rules in a separate rule rewriting process.
- **Black-Box Approach** This approach does not take the internal structure of the network into account. Rather, these algorithms directly extract rules, which reflect the correlation between the inputs and the outputs of a network.
- **Eclectic Approach** This approach incorporates principals of both decompositional and black-box approaches. In order to find a relation between the input and the output values of a network, they at least partly analyses the internal structure of the network.

2.4.4 Consistency

The consistency of a rule extraction algorithm describes how reliably, under differing training sessions, the algorithm is able to extract sets of rules with the same degree of accuracy.

2.4.5 Portability

This means the applicability of the rule extraction algorithm to different domains, different network's topologies and different learning techniques.

2.4.6 Space Exploration Methodology

Rule extraction algorithms can be classified according to the methodology used for exploring the space of possible rules. The main approaches are to use some kind of systematic search or to view the process of exploring the rule space as a learning task.

2.5 Some Rule Extraction Algorithms

2.5.1 RULEX Technique

2.5.1.1 Description

The technique is designed by Andrews and Geva, [Andrews and Geva, 1995], to exploit the manner of construction of a particular type of multi-layer perceptron (MLP). This is a representative of a class of local response ANN that performs function approximation and classification in a manner similar to Radial Basis Function (RBF), networks.

The hidden units of the CEBP network are sigmoid-based locally responsive units (LRUs) that have the effect of partitioning the training data into a set of disjoint regions, each region being represented by a single hidden layer unit. Each LRU is composed of a set of ridges, one ridge for each dimension of the input. A ridge will produce appreciable output only if the value presented as input lies within the active range of the ridge.

The LRUs are based on the fact that for the sigmoidal function $f(u) = 1/(1 + e^{-u})$, the expression $f(ax-c-b/2) - f(ax-c+b/2)$, with appropriate values for the parameters, defines a bump in one dimension with centre c and width b (See Figure 2.3). The LRU output is the threshold sum of the activations of the ridges. In order for a vector to be classified by an

LRU, each component of the input vector must lie within the active range of its corresponding ridge.

2.5.1.2 Algorithm Evaluation

a) Rule Format: In the directly extracted rule set each rule contains an antecedent condition for each input dimension as well as a rule consequent, which describes the output class covered by the rule. RULEX provides a rule simplification process, which removes redundant rules and antecedent conditions from the directly extracted rules. The reduced rule set contains rules that consist of only those antecedents that are actually used by the trained network in discriminating between input patterns.

*IF Ridge₁ is Active and ... and Ridge_N is Active
THEN the pattern belongs to the 'Target Class'*

The active range for each ridge can be calculated from its center, breadth, and steepness (c_i , b_i , k_i), weights in each dimension. This means that it is possible to directly decompile the LRU parameters into a conjunctive propositional rule of the form.

*IF $c_1 - b_1 + 2k_1^{-1} \leq x_1 \leq c_1 + b_1 - 2k_1^{-1}$ AND ...
AND $c_N - b_N + 2k_N^{-1} \leq x_N \leq c_N + b_N - 2k_N^{-1}$
THEN the pattern belongs to the 'Target Class'*

For discrete valued input, it is possible to enumerate the active range of each ridge as an OR'ed list of values that will activate the ridge. In this case it is possible to state the rule associated with the LRU in the form.

*IF v_{1a} OR v_{1b} ... OR v_{1n} AND AND v_{Na} OR v_{Nb} ... OR v_{Nn}
THEN the pattern belongs to the 'Target Class'*
*(where v_{ia} , v_{ib} , ..., v_{in} are contiguous values in the i^{th} input
dimension and $v_{ia} \geq c_i - b_i + 2k_i^{-1}$ and $v_{in} \leq c_i + b_i - 2k_i^{-1}$)*

b) Rule Quality: The rule quality criteria provide insight into the degree of trust that can be placed in the explanation.

(i) Accuracy: Despite the mechanism employed to avoid LRUs ‘overlapping’ during network training, it is clear that there is some degree of interaction between LRUs. (The larger the values of the parameters $k1$ and $k2$ the less the interaction between units but the slower the network training.) This effect becomes more apparent in problem domains with high dimension input space and in network solutions involving large numbers of LRUs. Further, RULEX approximates the hyper-ellipsoidal local cluster functions of the network with hyper-rectangles. It should be noted that while the accuracy for RULEX are worse than the underlying network they are comparable to those obtained from C4.5.

(ii) Comprehensibility: Comprehensibility is inversely related to the number of rules and to the number of antecedents per rule. The used network is based on a greedy, covering algorithm. Given that RULEX converts each LRU into a single rule, the extracted rule set contains, at most, the same number of rules as there are LRU’s in the trained network. The rule simplification procedures built into RULEX potentially reduces the size of the rule set and ensures that only significant antecedent conditions are included in the final rule set. This leads to extracted rules with as high comprehensibility as is possible.

(iii) Consistency: Rule extraction algorithms that generate rules by querying the trained network with patterns drawn randomly from the problem domain have the potential to generate different rule sets from any given training run of the neural network. Such algorithms have the potential for low consistency. RULEX on the other hand is a consistent algorithm that always generates the same rule set from any given training run of the network.

(iv) Fidelity: Fidelity is closely related to accuracy. In general, the rule sets extracted by RULEX display an extremely high degree of fidelity with the network from which they were drawn.

c) Translucency: RULEX is decompositional in that rules are extracted at the level of the hidden layer units. Each LRU is treated in isolation with the local cluster weights being converted directly into a rule.

Table 2.1. *Rule Quality Assessment* [Andrews and Geva, 1995]

Domain	CEBP network Accuracy	RULEX Accuracy	LRUs	Rules	Antecedents per Rule	RULEX Fidelity
Wisconsin Breast Cancer	96.8%	94.4%	5	5	24	97.5%
Horse Colic	86.5%	85.9%	5	2.5	8	99.3%
Glass Identification	60.9%	57.5%	22	19	6	94.3%
Cleveland Heart Disease	84.2%	80.2%	4	3	5	95.3%
Hungarian Heart Disease	85.4%	81.3%	3	2	5	95.2%
Hepatitis Prognosis	83.8%	78.7%	6	4	8	93.9%
Iris Plant Classification	95.3%	94.0%	3	3	3	98.6%

d) Algorithmic Complexity: The combination of ANN learning and ANN rule-extraction involves additional computational cost over direct rule-learning techniques. The majority of the modules are linear in the number of LRU's (or rules) and the number of input dimensions, $O(lc.n)$. The modules associated with rule simplification are, at worst, polynomial in the number of rules, $O(lc^2)$. RULEX is therefore computationally efficient and has some significant advantages over rule extraction algorithms that rely on a (potentially exponential) 'search and test' strategy.

e) Portability: RULEX is non-portable having been specifically designed to work with a specified type of neural networks. This means that it cannot be used as a general-purpose device for providing an explanation component for existing neural networks. However, the underlying network is applicable to a broad range of problem domains (including continuous valued, discrete valued domains and domains which include missing values). Hence RULEX is also potentially applicable to a broad variety of problem domains.

2.5.2 *M-of-N Technique*

To overcome the high complexity of SUBSET and to increase the comprehensibility of a rule system, Towell and Shavlik [Towell and Shavlik, 1993] developed a second rule extraction method known as M-of-N algorithm, which is one component of the Knowledge based Neural Network (KBNN) system.

2.5.2.1 *Description*

The phases of the M-of-N algorithm are shown below:

- **Clustering Step:** *Generate an Artificial Neural Network using the KBANN system and train using back-propagation. With each hidden and output unit, form groups of similarly-weighted links;*
- **Averaging Step:** *Set link weights of all group members to the average of the group;*
- **Eliminating Step:** *Eliminate any groups which do not significantly affect whether the unit will be active or inactive;*
- **Optimizing Step:** *Holding all link weights constant, optimize biases of all hidden and output units using the back-propagation algorithm;*
- **Rule Extracting Step:** *Form a single rule for each hidden an output unit; the rule consists of a threshold given by the bias and weighted antecedents specified by the remaining links;*
- **Simplifying Step:** *where possible, simplify rules to eliminate superfluous weights and thresholds.*

2.5.2.2 *Algorithm Evaluation*

a) Rule Format: If (M of the following N antecedents are true) then....

b) Rule Quality: There are two dimensions: (a) the rules must accurately categorize examples that were not seen during training, and (b) the extracted rules must capture the information contained in the KBNN, for assessing the quality of rules extracted both from

their own algorithm and from the set of algorithms they use for the purposes of comparison. The M-of-N idea yields a more compact rule representation than conventional conjunctive rules produced by algorithms such as Subset. In addition the M-of-N algorithm outperformed a subset of published symbolic learning algorithms in terms of the accuracy and fidelity of the rule set extracted from a cross-section of problem domains.

c) **Translucency:** Decompositional

d) **Algorithmic Complexity:** The algorithm addresses the question of reducing the complexity of rules searched by clustering the ANN weights into equivalence classes (and hence extracting *M-of-N* type rules). Using three indicative parameters: (1) the number of units in the ANN (u), (2) the average number of links received by a unit (l), and (3) the number of training examples (n). The complexity shown in Table 2.2.

Table 2.2. *Complexity of the M-of-N algorithm* [Towell and Shavlik, 1993].

Step No.	Name	Estimated Complexity
1	Clustering	$O(u.l^2)$
2	Averaging	$O(u.l)$
3	Eliminating	$O(n.u.l)$
4	Optimizing	precise analysis is inhibited by the use of back-propagation in this optimisation phase
5	Extracting	$O(u..l)$
6	Simplifying	$O(u.l)$

e) **Portability:** The *M-of-N* algorithm is applicable to feedforward networks with non-negative and approximately binary outputs of neurons. It also requires weighted connections which can easily be clustered into relatively few groups of similar weighted links.

There are a number of experiments used to illustrate the efficiency of the *M-of-N* technique including two from the field of molecular biology: (a) prokaryotic promoter recognition, and (b) primate splice-junction determination as well as the perennial '*Three Monks*' problem(s). In some experiments, M-of-N rules had a higher accuracy than the underlying network. This can be explained by a further generalization carried out when clustering and pruning connections in the network.

2.5.3 *BIO-RE Technique*

2.5.3.1 *Description*

Taha and Ghosh [Taha and Ghosh, 1996a] have developed a new technique known as Binarised Input-Output Rule Extraction (BIO-RE). It is a black-box algorithm that extracts binary rules from any ANN; BIO-RE consists of the following steps:

1. *Obtain the output of the network for each possible pattern of input attributes.*
2. *Generate a truth table by concatenating each input pattern with its corresponding network output.*
3. *Generate boolean functions from the truth table.*

It should be noted that for generating the truth table all possible input patterns, not only the training examples, are used. For generating rules the algorithm can make use of any available boolean simplification method.

2.5.3.2 *Algorithm Evaluation*

- a) **Rule Format:** propositional if-then rules
- b) **Translucency:** Black-Box
- c) **Algorithmic Complexity:** Taha and Ghosh report the complexity of BIO-RE as very low. Since logical minimization results in an optimal set of rules directly relating the inputs of the networks to its outputs, no further simplification and rule-rewriting is required after generating rules from the truth table. It should be noted, however, that the complexity of logical minimization grows exponentially with the number of attributes in the truth table. Therefore, the extraction of an optimal set of rules is only possible for domains with small number of attributes.
- d) **Portability:** BIO-RE is an algorithm without any requirements for certain network architectures and training regimes. However, it is only suitable for domains with binary attributes or attributes which can be binarised without degrading the performance of a network.

2.5.4 Partial-RE Technique

2.5.4.1 Description

Taha and Ghosh [Taha and Ghosh, 1996a] have developed a second technique known as Partial-RE. It extracts rules representing the most important knowledge embedded in a backpropagation network. The phases of the Partial-RE algorithm are shown below:

1. For each hidden or output node, j , the positive and negative incoming links are sorted in descending order of weight values into two sets.
2. Starting from the highest positive weight (say, i), the algorithm searches for individual incoming links that can cause the node j to be active regardless of other input links to this node.
3. If such links exist,
For each link, generate a rule: $\text{Node}_i \xrightarrow{cf} \text{Node}_j$, where cf represents the measure of belief in the extracted rule and is equal to the activation value of node j with this current combination of inputs. Mark this link as being used in a rule so that it cannot be used in any further combinations when inspecting node j .
4. Partial-RE continues checking subsequent weights in the positive set until it finds one that cannot activate the current node j by itself.
5. If more detailed rules are required (i.e., comprehensibility measure $p > 1$), then Partial-RE starts looking for combinations of two unmarked links starting from the first (maximum) element of the positive set. This process continues until Partial-RE reaches its terminating criteria. (That is, maximum number of antecedents = p)
6. Also, it looks for negative weights such their not being active allows a node in the next layer to be active, and extracts rule in the format:

$$\text{Not } \text{Node}_g \xrightarrow{cf} \text{Node}_j$$

7. Moreover, it looks for small combinations of positive and negative links that can cause any hidden/output node to be activate, to extract rules such as:

$$\text{Node}_i \text{ And } \text{Not } \text{Node}_g \xrightarrow{cf} \text{Node}_j$$

where the link between Node_i and Node_j is positive and between Node_g and Node_j is negative.

8. *After extracting all rules, a rewriting procedure takes place. Within this rewriting procedure any antecedent that represents an intermediate concept (i.e., a hidden node) is replaced by the corresponding set of conjuncted input features that causes it to be active. Final rules are written in the format:*

$$X_i \geq \mu_i \quad \text{And} \quad X_g \leq \mu_g \xrightarrow{cf} \text{Consequent}_j$$

2.5.4.2 Algorithm Evaluation

- a) **Rule Format:** propositional if-then rules
- b) **Translucency:** Decompositional
- c) **Algorithmic Complexity:** The complexity of the algorithm grows polynomial with number of incoming connections for hidden and output neurons.
- d) **Portability:** The algorithm is applicable to multi-layer feed-forward networks learning tasks in discrete domains. Partial-RE is, in contrast to BIO-RE, suitable for large size problems.

The advantages of Partial-RE technique:

1. It is easily parallelizable, as nodes can be inspected concurrently.
2. It avoids the rewriting procedure involved in SUBSET algorithms and is able to produce soft rules with associated measures of belief or certainty factors.
3. Partial-RE algorithm is suitable for large size problems, since extracting all possible rules is NP-hard and extracting only the most effective rules is a practical alternative.
4. The level of fidelity of the extracted rules is adjustable according to the needs of the application.

The disadvantage of Partial-RE technique:

The comprehensibility of the rules is similar to the comprehensibility of those extracted with BIO-RE, which was judged to be comparatively low.

2.5.5 Full-RE Technique

2.5.5.1 Description

Taha and Ghosh [Taha and Ghosh, 1996a] have developed a third technique known as Full Rule Extraction (Full RE). It extracts all possible rules and corresponding certainty factors for each neuron with monotonically increasing activation function in a feed-forward ANN. The phases of the Full-RE algorithm are shown below:

1. Initially, for each hidden neuron j , a rule

$$IF(w_{1j}X_1 + w_{2j}X_2 + \dots + w_{nj}X_n) > \alpha_j \xrightarrow{cf} Consquent_j$$

is formed where w_{ij} is the weight of the connection between the neuron i and j and α_j is a constant determined by the activation value of j .

2. Discretize each input value $X_i \in (a_i, b_i)$ into k intervals such that

$$X_i \in \{a_i, d_{i,1}, \dots, d_{i,k-1}, b_i\}$$

3. The following linear programming (LP) problem is then solved to find the minimal combination of input values required for the neuron to fire:

For each neuron minimize

$$w_{1j}X_1 + w_{2j}X_2 + \dots + w_{nj}X_n$$

Such that:

$$w_{1j}X_1 + w_{2j}X_2 + \dots + w_{nj}X_n > \alpha_j \quad \text{and} \quad X_i \in \{a_i, d_{i,1}, \dots, d_{i,k-1}, b_i\} \quad \forall i = 1, \dots, n.$$

Any LP tool can be used to solve this LP problem. Certainty factors are assigned to a rule depending on the neuron activation function.

4. For output neurons, Rules are extracted with a simplified version of the procedure described above.
5. Finally, rules containing references to hidden neurons in their antecedents are rewritten in terms of the attributes of the domain.

2.5.5.2 Algorithm Evaluation

a) **Rule Format:** propositional if-then rules

b) **Translucency:** Decompositional

- c) Algorithmic Complexity:** The complexity depends on the tool used for the LP problem. The SIMPLEX algorithm, for instance, takes worst-case exponential time in the number of neurons in a network layer. Other tools solve the LP problem in worst-case polynomial time.
- d) Portability:** Full-RE is applicable to feed-forward networks containing neurons with monotonically increasing activation function. It can extract rules from networks trained with continuous, discrete, and binary input attributes. This capability makes Full-RE a universal extractor.

FRULEX – FUZZY RULES EXTRACTOR

3.1 Overview of FRULEX Approach

FRULEX is a neuro-fuzzy approach for fuzzy rules extraction. It can also be said to be a fuzzy inference system creation algorithm. Classical fuzzy inference system creation algorithms use only the dataset to create the fuzzy system. FRULEX has both the dataset and the model of the dataset in the form of Neural Network. Experimental results of FRULEX have been shown in literature, [Abdel Hady *et al.*, 2003, and 2004]. Figure 3.1 shows the outline of the FRULEX approach. In the initialization phase, a set of initial fuzzy rules is extracted from the given data set with an adaptive self-constructing rule generator. The j^{th} fuzzy rule is defined as follow, [Jang *et al.*, 1998]:

$$\begin{aligned} R_j : IF (x_1 \text{ IS } \mu_j(x_1)) \text{ AND... AND } (x_i \text{ IS } \mu_j(x_i)) \text{ AND... AND } (x_N \text{ IS } \mu_j(x_N)) \\ THEN (y_1 \text{ IS } w_{j1}) \text{ AND...AND } (y_k \text{ IS } w_{jk}) \text{ AND...AND } (y_M \text{ IS } w_{jM}) \end{aligned} \quad (3.1)$$

where $\mu_{ij}(x_i)$ are membership functions, each of which is a normalized ridge function that is constructed from the difference of two sigmoidal functions, as shown below.

$$\mu_{ij}(x_i) = r(x_i; c_{ij}, b_{ij}, k_{ij}) = \frac{\sigma(k_{ij}, (x_i - c_{ij} + b_{ij})) - \sigma(k_{ij}, (x_i - c_{ij} - b_{ij}))}{\sigma(k_{ij}, b_{ij}) - \sigma(k_{ij}, -b_{ij})} \quad (3.2)$$

$$\sigma(k_{ij}, (x_i - c_{ij} + b_{ij})) = \frac{1}{1 + \exp(-(x_i - c_{ij} + b_{ij})k_{ij})} \quad (3.3)$$

with center c_{ij} , width b_{ij} and steepness k_{ij} , and w_{jk} is a constant represents the k^{th} consequent part. The firing strength of the rule j , [Jang *et al.*, 1998], has the form:

$$\alpha_j = \prod_{i=1}^N r(x_i; c_{ij}, b_{ij}, k_{ij}) \quad (3.4)$$

Also, we use the centroid defuzzification method to calculate the output of this fuzzy system as follow:

$$y_k^{(4)} = \frac{\sum_{j=1}^J \alpha_j \cdot w_{jk}}{\sum_{j=1}^J \alpha_j} \quad (3.5)$$

In the parameter optimization phase, we improve the accuracy of the initial fuzzy rule set with neural network techniques. In the rule base simplification phase, *FRULEX* implements facilities for simplifying the optimized rule set in order to improve the interpretability of the rule set. Figure 3.2 shows the four-layer *MCRBP* neural network constructed based on the fuzzy rules obtained in the first phase.

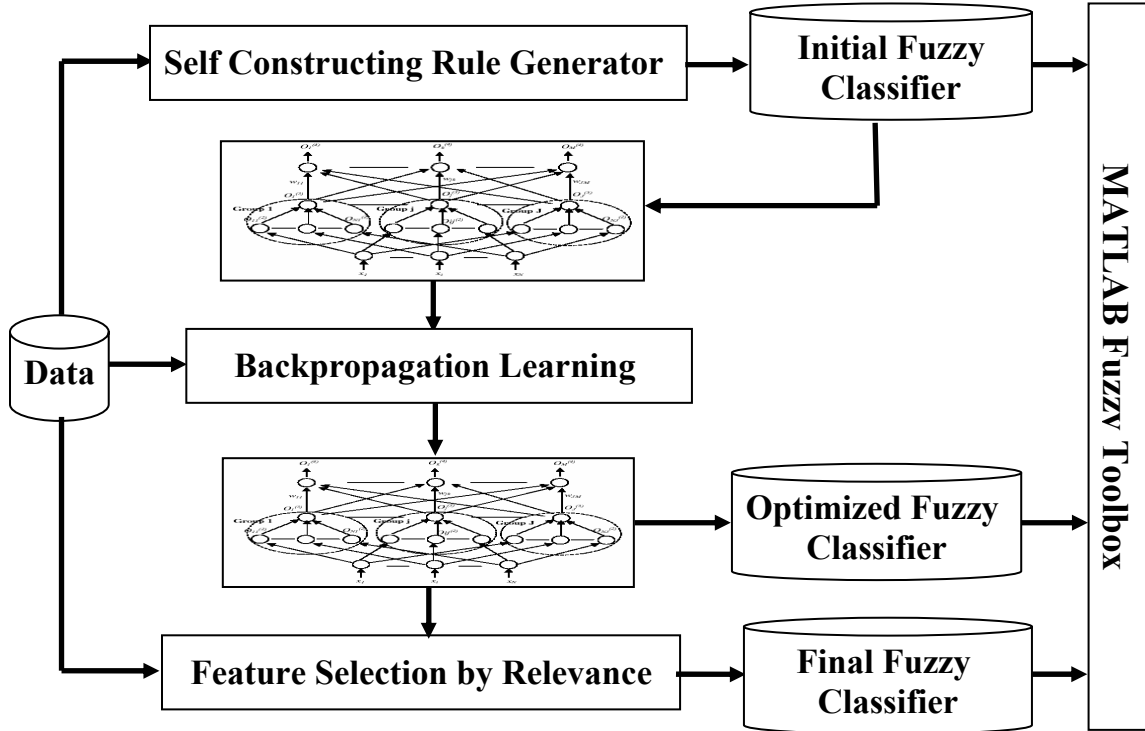


Figure 3.1. Outline of *FRULEX* Approach

The layers of the *MCRBP* neural network are described as follows:

- **Layer 1** contains N nodes. *Node i* of this layer produces output by transmitting its input signal directly to layer 2, i.e., for $1 \leq i \leq N$

$$O_i^{(1)} = x_i \quad (3.6)$$

- **Layer 2** contains J groups and each group contains N nodes. Each group representing the IF-part of a fuzzy rule. *Node (i, j)* of this layer produces its output by computing the value of the corresponding normalized ridge function, for $1 \leq i \leq N$ and $1 \leq j \leq J$

$$O_{ij}^{(2)} = r_{ij} = r(x_i; c_{ij}, b_{ij}, k_{ij}) = \frac{\sigma(k_{ij}, (x_i - c_{ij} + b_{ij})) - \sigma(k_{ij}, (x_i - c_{ij} - b_{ij}))}{\sigma(k_{ij}, b_{ij}) - \sigma(k_{ij}, -b_{ij})} \quad (3.7)$$

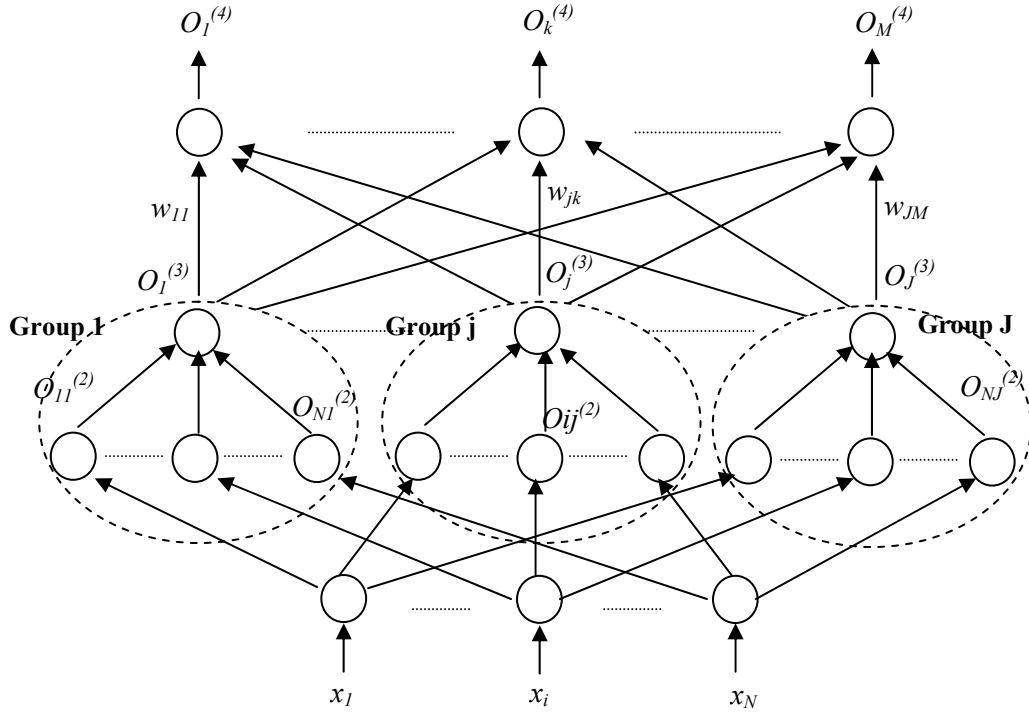


Figure 3.2. Architecture of the Proposed Backpropagation Neural Network

- **Layer 3** contains J nodes. *Node j* of this layer produces its output by computing the value of the logistic function, i.e., for $1 \leq j \leq J$

$$O_j^{(3)} = \ell(\underline{x}; \underline{c}_j, \underline{b}_j) = \sigma(K, \sum_{i=1}^N O_{ij}^{(2)} - B) \quad (3.8)$$

- **Layer 4** contains M nodes. *Node k* of this layer produces its output by the centroid defuzzification, i.e.,

$$O_k^{(4)} = \frac{\sum_{j=1}^J O_j^{(3)} \cdot w_{jk}}{\sum_{j=1}^J O_j^{(3)}} \quad (3.9)$$

Clearly, c_{ij} , b_{ij} , and w_{jk} are the parameters that can be tuned to improve the performance of the fuzzy system. We use the backpropagation gradient descent method to refine these parameters. Trained RBP networks can be used for numeric inference, or final fuzzy rules can be extracted from networks for symbolic reasoning.

3.2 Self-Constructing Rule Generator

First, the given input-output data set is partitioned into fuzzy (overlapped) clusters. The degree of association is strong for data points within the same fuzzy cluster and weak for data points in different fuzzy clusters. Then, a fuzzy if-then rule describing the distribution of the data in each fuzzy cluster is obtained. These fuzzy rules form a rough model of the unknown system and the precision of description can be improved in the phase of parameter identification.

Lee *et al.* [Lee *et al.*, 2003] have proposed an approach for neuro-fuzzy system modeling using this method. Unlike common clustering-based methods (e.g. c-means, fuzzy c-means) which require the number of clusters, and hence the number of rules, to be appropriately pre-selected, SCRG performs clustering with the ability to adapt the number of clusters as it proceeds.

- For a system with N inputs and M outputs, we define a fuzzy cluster j as a pair $(\ell_j(\underline{x}), \underline{w}_j)$ where $\ell_j(\underline{x})$ is defined as:

$$\ell_j(\underline{x}) = \ell(\underline{x}; \underline{c}_j, \underline{b}_j, \underline{k}_j) = \sigma(K, \sum_{i=1}^N r(x_i; c_{ij}, b_{ij}, k_{ij}) - B) \quad (3.10)$$

where $\underline{x} = [x_1, \dots, x_N]$, $\underline{c}_j = [c_1, \dots, c_N]$, $\underline{b}_j = [b_1, \dots, b_N]$, $\underline{k}_j = [k_1, \dots, k_N]$, K , and $\underline{w}_j$ denote the input vector, center vector, width vector, steepness and height vector respectively, of the cluster j .

- Let J be the number of existing fuzzy clusters and S_j be the size of cluster j . Clearly, J initially equals zero.
- For an input-output instance $v, (\underline{p}_v, \underline{q}_v)$ where $\underline{p}_v = [p_{v1}, \dots, p_{vN}]$, and $\underline{q}_v = [q_{v1}, \dots, q_{vM}]$. We calculate $\ell_j(\underline{p}_v)$ for each existing cluster $j, 1 \leq j \leq J$. We say that instance v passes input-similarity test on cluster j if

$$\ell_j(\underline{p}_v) \geq \rho \quad (3.11)$$

where $\rho, 0 \leq \rho \leq 1$, is a predefined threshold. Then, we calculate

$$e_{vjk} = |q_{vk} - w_{jk}| \quad (3.12)$$

for each cluster j on which instance v has passed the input-similarity test. Let $d_k = q_{kmax} - q_{kmin}$ where q_{kmax} and q_{kmin} are the maximum and minimum value of the k^{th} output, respectively, of the given data set.

- We say that instance v passed the output-similarity test on cluster j if

$$e_{vjk} \leq \tau d_k \quad (3.13)$$

where $\tau, 0 \leq \tau \leq 1$, is another predefined threshold.

- We have two cases. First, there is no existing fuzzy clusters on which instance v has passed both input-similarity and output-similarity tests. For this case, we assume that instance v is not close enough to any existing cluster and a new fuzzy cluster $k = J+1$ is created with

$$\underline{c}_k = \underline{p}_v, \underline{b}_k = \underline{b}_o, \text{ and } \underline{w}_k = \underline{q}_v \quad (3.14)$$

where $\underline{b}_o = [b_o, \dots, b_o]$ is a user-defined constant vector. Note that the new cluster k contains only one member, instance v , at this time. Of course, the number of existing clusters is increased by 1 and the size of cluster k should be initialed to 1,

$$J = J+1 \text{ and } S_k=1 \quad (3.15)$$

- Second, if there exist a number of fuzzy clusters on which instance v has passed both input-similarity and output-similarity tests, let these clusters are $j_1, j_2 \dots$ and j_f and let the cluster t be the cluster with the largest membership degree.

$$\ell_t(\underline{p}_v) = \max(\ell_{j_1}(\underline{p}_v), \ell_{j_2}(\underline{p}_v), \dots, \ell_{j_f}(\underline{p}_v)) \quad (3.16)$$

- In this case, we assume that instance v is closest to cluster t and cluster t should be modified to include instance v as its member. The modification to cluster t is shown below, [Lee *et al.*, 2003], for $1 \leq i \leq N$

$$b_{it} = \sqrt{\frac{(S_t - 1)(b_{it} - b_o)^2 + S_t c_{it}^2 + p_{vi}^2}{S_t} - \frac{S_t + 1}{S_t} \left(\frac{S_t c_{it} + p_{vi}}{S_t + 1} \right)^2} + b_0 \quad (3.17)$$

$$c_{it} = \frac{S_t c_{it} + p_{vi}}{S_t + 1} \quad (3.18)$$

$$w_{tk} = \frac{S_t w_{tk} + q_{vk}}{S_t + 1} \quad (3.19)$$

$$S_t = S_t + 1 \quad (3.20)$$

Note that J is not changed in this case.

- The above-mentioned process is iterated until all the input-output instances have been processed. At the end, we have J fuzzy cluster. Note that each cluster j is described as $(\ell_j(\underline{x}), \underline{w}_j)$ where $\ell_j(\underline{x})$ contains center vector $\underline{c}_j$, and width vector $\underline{b}_j$.
- We can represent cluster j by a fuzzy rule of the form in shown in Figure 3.1 with

$$\mu_{ij}(x_i) = r(x_i; c_{ij}, b_{ij}, k_{ij}) \quad (3.21)$$

for $1 \leq i \leq N$ and the conclusion is $\underline{w}_j$ for $1 \leq j \leq M$.

- Finally, we have a set of J initial fuzzy rules for the given input-output data set. With this approach, when new training data are considered, the existing clusters can be

adjusted or new cluster can be created, without the necessity of generating the whole set of rules from the scratch.

3.3 Backpropagation Training for RBP Neural Network

3.3.1 Introduction

Backpropagation is a systematic method for training multiple (three or more)-layer artificial neural networks. The illustration of this training algorithm in 1986 by Rumelhart, Hinton, and Williams [Rumelhart *et al.*, 1986] was the key step in making neural networks practical in many real-world applications. However, Rumelhart, Hinton, and Williams were not the first to develop the backpropagation algorithm. It was developed independently by Parker [Parker, 1987] in 1982 and earlier by Werbos [Werbos, 1974] in 1974 as part of his Ph.D. dissertation at Harvard University. Today, it is estimated that 80% of all applications utilize this backpropagation algorithm in one form or another. In spite of its limitations, backpropagation has dramatically expanded the range of problems to which neural network can be applied, perhaps because it has a strong mathematical foundation.

3.3.2 Backpropagation Learning Algorithm

After the set of J initial fuzzy rules is obtained, we improve the accuracy of these rules with neural network techniques in the phase of parameter optimization. First, a four-layer fuzzy rules-based RBP network is constructed by turning each fuzzy rule into a sigmoid-based local response unit (LRU), as shown in Figure 3.2. Then, a gradient method performing the steepest descent on a surface in the network parameter space is used. The goal of this phase is to adjust both the premise and consequent parameters so as to minimize the mean squared error

$$E = \frac{1}{P} \sum_{v=1}^P E_v \quad (3.22)$$

where $E_v = \frac{1}{2} \sum_{k=1}^M (e_{vk})^2$, $e_{vk} = y_{vk} - q_{vk}$ and $y_{vk} = O_k^{(4)}(\underline{p}_v)$ is the actual output of the v^{th} training pattern.

- The update formula for a generic weight α is

$$\Delta\alpha = -\eta_\alpha (\partial E / \partial \alpha) \quad (3.23)$$

where η_α is the learning rate for that weight. In summary, given training set T of P training patterns $T = \{(\underline{p}_v, \underline{q}_v) : v = 1, \dots, P\} = \{(p_{v1}, \dots, p_{vN}), (q_{v1}, \dots, q_{vM})\}$.

- For the sake of simplicity, the subscript v indicating the current sample will be dropped in the following derivation.
- Starting at the first layer, a forward pass is used to compute the activity levels of all the nodes in the network to obtain the current output values. Then, starting at the output layer, a backward pass is used to compute $\partial E / \partial \alpha$ for all the nodes.
- Let us start with the derivation of the square error with respect to the output weight for the 4th layer, w_{jk} that is to be adjusted. The delta rule training gives

$$\Delta w_{jk} = -\eta \left(\frac{\partial E}{\partial w_{jk}} \right) \quad (3.24)$$

where the square error E is now defined by

$$E = \frac{1}{2} e_k^2 = (y_k - q_k)^2 \quad (3.25)$$

- We can evaluate the last term of equation (3.24) using the chain rule of differentiation, which gives

$$\frac{\partial E}{\partial w_{jk}} = \frac{1}{2} \frac{\partial e_k^2}{\partial O_k^{(4)}} \frac{\partial O_k^{(4)}}{\partial w_{jk}} \quad (3.26)$$

- Each of these terms is evaluated in turn. The partial derivative of e_k^2 with respect to $O_k^{(4)}$ gives

$$\frac{\partial e_k^2}{\partial O_k^{(4)}} = 2(y_k - q_k) = 2e_k \quad (3.27)$$

- We can see, from equation (3.9), that $O_k^{(4)}$ is the average sum of the weighted inputs from the 3rd layer. Taking the partial derivative with respect to w_{jk} gives

$$\frac{\partial O_k^{(4)}}{\partial w_{jk}} = \frac{O_j^{(3)}}{\sum_{t=1}^J O_t^{(3)}} \quad (3.28)$$

- Substituting equations (3.27), and (3.28) into equation (3.26) gives

$$\frac{\partial E}{\partial w_{jk}} = \delta_k^{(4)} \frac{O_j^{(3)}}{\sum_{t=1}^J O_t^{(3)}} \quad (3.29)$$

where the error term $\delta_k^{(4)}$ is defined as

$$\delta_k^{(4)} = e_k \quad (3.30)$$

- Substituting equation (3.29) into equation (3.24) gives

$$\Delta w_{jk} = -\eta \delta_k^{(4)} \frac{O_j^{(3)}}{\sum_{t=1}^J O_t^{(3)}} \quad (3.31)$$

- And hence, the weight update equation will have the form

$$w_{jk}(t+1) = w_{jk}(t) - \eta \delta_k^{(4)} \frac{O_j^{(3)}}{\sum_{t=1}^J O_t^{(3)}} \quad (3.32)$$

- Now, let us derive the square error with respect to the weights c_{ij}, b_{ij} that is to be adjusted. The delta rule training gives

$$\Delta c_{ij} = -\eta \left(\frac{\partial E}{\partial c_{ij}} \right) \quad (3.33)$$

- Since several output errors may be involved, the total squared error E is defined by

$$E = \frac{1}{2} \sum_{k=1}^M (e_k)^2 \quad (3.34)$$

- We can evaluate the last term of equation (3.33) using the chain rule of differentiation, which gives

$$\frac{\partial E}{\partial c_{ij}} = \frac{1}{2} \sum_{k=1}^M \frac{\partial e_k^2}{\partial O_k^{(4)}} \frac{\partial O_k^{(4)}}{\partial O_j^{(3)}} \frac{\partial O_j^{(3)}}{\partial O_{ij}^{(2)}} \frac{\partial O_{ij}^{(2)}}{\partial c_{ij}} \quad (3.35)$$

- The first term is already given by equation (3.27). Taking the partial derivative of

equation (3.9) with respect to $O_j^{(3)}$ gives

$$\frac{\partial O_k^{(4)}}{\partial O_j^{(3)}} = \frac{w_{jk} - O_k^{(4)}}{\sum_{t=1}^J O_t^{(3)}} \quad (3.36)$$

Since the output of *Node j* in the third layer has the form

$$O_j^{(3)} = \sigma(K, \sum_{i=1}^N O_{ij}^{(2)} - B) \quad (3.37)$$

- Taking the partial derivative of equation (3.37) with respect to $O_{ij}^{(2)}$ gives

$$\frac{\partial O_j^{(3)}}{\partial O_{ij}^{(2)}} = K O_j^{(3)} [1 - O_j^{(3)}] \quad (3.38)$$

Since the output of *Node (i, j)* in the second layer has the form

$$O_{ij}^{(2)} = \frac{\sigma(k_{ij}, (x_i - c_{ij} + b_{ij})) - \sigma(k_{ij}, (x_i - c_{ij} - b_{ij}))}{\sigma(k_{ij}, b_{ij}) - \sigma(k_{ij}, -b_{ij})} \quad (3.39)$$

- Taking the partial derivative of equation (3.39) with respect to c_{ij} gives

$$\frac{\partial O_{ij}^{(2)}}{\partial c_{ij}} = -k_{ij} \left[\frac{\sigma_{ij}^+ (1 - \sigma_{ij}^+) - \sigma_{ij}^- (1 - \sigma_{ij}^-)}{\sigma(k_{ij}, b_{ij}) - \sigma(k_{ij}, -b_{ij})} \right] \quad (3.40)$$

- Substituting equations (3.27), (3.36), (3.38), and (3.40) into equation (3.35) gives

$$\frac{\partial E}{\partial c_{ij}} = -\delta_{ij}^{(2)} k_{ij} \left[\frac{\sigma_{ij}^+ (1 - \sigma_{ij}^+) - \sigma_{ij}^- (1 - \sigma_{ij}^-)}{\sigma(k_{ij}, b_{ij}) - \sigma(k_{ij}, -b_{ij})} \right] \quad (3.41)$$

If we define the error term $\delta_{ij}^{(2)}$ as

$$\delta_{ij}^{(2)} = \delta_j^{(3)} K O_j^{(3)} (1 - O_j^{(3)}) \quad (3.42)$$

and the error term $\delta_j^{(3)}$ as

$$\delta_j^{(3)} = \sum_{k=1}^M \delta_k^{(4)} (w_{jk} - O_k^{(4)}) / \sum_{t=1}^M O_t^{(3)} \quad (3.43)$$

- Substituting equation (3.41) into equation (3.33) gives

$$\Delta c_{ij} = \eta \delta_{ij}^{(2)} k_{ij} \left[\frac{\sigma_{ij}^+ (1 - \sigma_{ij}^+) - \sigma_{ij}^- (1 - \sigma_{ij}^-)}{\sigma(k_{ij}, b_{ij}) - \sigma(k_{ij}, -b_{ij})} \right] \quad (3.44)$$

Hence, the update equation of the center c_{ij} will take the form

$$c_{ij}(t+1) = c_{ij}(t) + \eta \delta_{ij}^{(2)} k_{ij} \left[\frac{\sigma_{ij}^+(1 - \sigma_{ij}^+) - \sigma_{ij}^-(1 - \sigma_{ij}^-)}{\sigma(k_{ij}, b_{ij}) - \sigma(k_{ij}, -b_{ij})} \right] \quad (3.45)$$

- Similarly, we can observe that

$$\Delta b_{ij} = -\eta \delta_{ij}^{(2)} k_{ij} \left[\frac{\sigma_{ij}^+(1 - \sigma_{ij}^+) + \sigma_{ij}^-(1 - \sigma_{ij}^-)}{\sigma(k_{ij}, b_{ij}) - \sigma(k_{ij}, -b_{ij})} \right] \quad (3.46)$$

Hence, the update equation of the breadth b_{ij} will take the form

$$b_{ij}(t+1) = b_{ij}(t) - \eta \delta_{ij}^{(2)} k_{ij} \left[\frac{\sigma_{ij}^+(1 - \sigma_{ij}^+) + \sigma_{ij}^-(1 - \sigma_{ij}^-)}{\sigma(k_{ij}, b_{ij}) - \sigma(k_{ij}, -b_{ij})} \right] \quad (3.47)$$

where t is the number of iteration.

The complete learning algorithm is summarized as follow:

1. Initialize the weights $\{c_{ij}, b_{ij}, k_{ij}\}_{i=1, \dots, N}^{j=1, \dots, J}$ and $\{w_{jk}\}_{j=1, \dots, J}^{k=1, \dots, M}$ with rule parameters obtained in the SCRG phase.
2. Select the next input vector $\underline{p}$ from T , propagate it through the network and determine the output $y_k = O_k^{(4)}$.
3. Compute the error terms as follows:

$$\delta_k^{(4)} = O_k^{(4)} - q_k \quad (3.48)$$

$$\delta_j^{(3)} = \sum_{k=1}^M \delta_k^{(4)} (w_{jk} - O_k^{(4)}) / \sum_{t=1}^M O_t^{(3)} \quad (3.49)$$

$$\delta_{ij}^{(2)} = \delta_j^{(3)} K O_j^{(3)} (1 - O_j^{(3)}) \quad (3.50)$$

4. Update the gradients of $\{c_{ij}, b_{ij}\}_{i=1, \dots, N}^{j=1, \dots, J}$ and $\{w_{jk}\}_{j=1, \dots, J}^{k=1, \dots, M}$ respectively according to:

$$\left(\frac{\partial E}{\partial c_{ij}} \right)^+ = -\delta_{ij}^{(2)} k_{ij} \left[\frac{\sigma_{ij}^+(1 - \sigma_{ij}^+) - \sigma_{ij}^-(1 - \sigma_{ij}^-)}{\sigma(k_{ij}, b_{ij}) - \sigma(k_{ij}, -b_{ij})} \right] \quad (3.51)$$

$$\left(\frac{\partial E}{\partial b_{ij}}\right)^+ = \delta_{ij}^{(2)} k_{ij} \left[\frac{\sigma_{ij}^+ (1 - \sigma_{ij}^+) + \sigma_{ij}^- (1 - \sigma_{ij}^-)}{\sigma(k_{ij}, b_{ij}) - \sigma(k_{ij}, -b_{ij})} \right] \quad (3.52)$$

$$\left(\frac{\partial E}{\partial w_{jk}}\right)^+ = \delta_k^{(4)} \frac{O_j^{(3)}}{\sum_{t=1}^J O_t^{(3)}} \quad (3.53)$$

5. After applying the whole training set T, Update the weights $\{c_{ij}, b_{ij}, k_{ij}\}_{i=1, \dots, N}^{j=1, \dots, J}$ and $\{w_{jk}\}_{j=1, \dots, J}^{k=1, \dots, M}$ respectively according to:

$$\Delta c_{ij} = -\eta \left(\frac{\partial E}{\partial c_{ij}} \right) \quad (3.54)$$

$$\Delta b_{ij} = -\eta \left(\frac{\partial E}{\partial b_{ij}} \right) \quad (3.55)$$

$$k_{ij} = \frac{K_o}{b_{ij}} \quad (3.56)$$

$$\Delta w_{jk} = -\eta \left(\frac{\partial E}{\partial w_{jk}} \right) \quad (3.57)$$

where η being the learning rate (i.e. the length of each gradient transition in the parameter space; by a proper selection of η the speed of convergence can be varied) and K_o is the initial steepness.

6. If $E < \varepsilon$ or maximum number of iterations reached stop else go to step 2. (where ε is the error goal)

3.4 Feature Subset Selection by Relevance

In the real world applications, the number of features is usually high which increases the complexity of the classification task. Some of these features may be irrelevant or adding noise to the problem. Choosing only the most relevant and noise-free features will increase

the classification accuracy, shorten learning time and make the final representation of the problem simpler. If they are removed from the feature set, classification accuracy will increase.

Feature subset selection is usually done by experts using domain knowledge. But in most domains where domain knowledge is not available, subset selection should be done by using data only. Using a subset of the available features will increase classification rate, shorten classification time and will also increase the comprehensibility of the acquired knowledge. In some real world applications, like medical diagnosis, finding the values of some of the features may be expensive such as expensive lab tests. [Molina *et al.*, 2002] presents an exhaustive survey for different feature selection algorithms.

3.4.1 Overview of Feature Subset Selection

Feature subset selection is an optimization problem, which is solved by searching the feature subset space. Three factors determine how good a feature subset selection algorithm is: Classification accuracy, Size of the subset, and Computational efficiency. In feature subset selection algorithms finding the optimal feature subset is a hard task. There are 2^N states in the search space (N: number of features). For large N values, evaluating all the states is computationally infeasible. Therefore, we have to use a heuristic search.

Doak [Doak, 1992] divides search algorithms into three groups: Exponential algorithms, Sequential algorithms and Randomized algorithms. Evaluation function is used to compare the feature subsets. It creates a numeric output for each state. Feature Subset Selection algorithm's goal is to optimize this function. We can classify evaluation functions in two different groups. A group that uses the classification algorithm itself for evaluation and another that use means other than classification algorithms (i.e. information from the data set).

For the representation of feature subsets, we chose binary string representation. In this representation, each subset is represented by N bits (N: number of features in the full set). Each bit represents presence (1) or absence (0) of that feature in the subset. For example, if

$N=4$, string 1001 will represent subset $\{f_1, f_4\}$. An illustrative example of a subset search space for 4 features is shown in Figure 3.3.

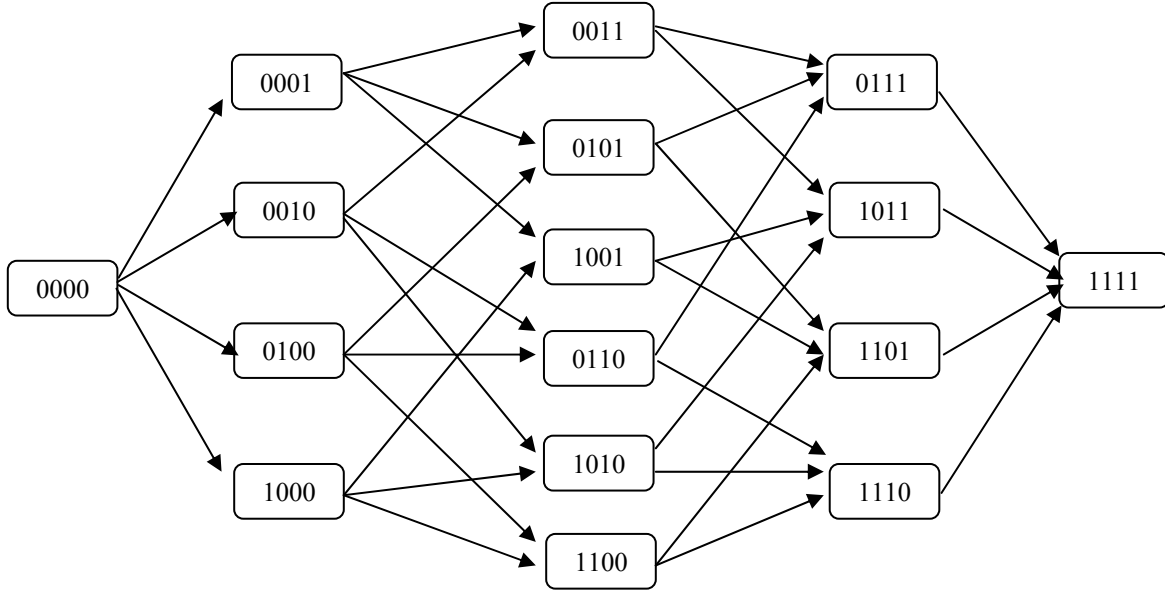


Figure 3.3. Feature Subset Selection Search Space

3.4.1.1 Search Algorithms

3.4.1.1.1 Exponential Search Algorithms

Some of the exponential search algorithms are *Exhaustive Search*, *Branch and Bound Search* [Narendra and Fukunaga, 1977] and *Beam Search*. Complexity of the exponential search algorithms is $O(2^N)$ (N : number of features). Exhaustive search evaluates every state in the search space. Exponential algorithms are computationally very expensive. Because of that, a limited search has to be used to make them computationally feasible. The limits make them less effective for real world applications.

3.4.1.1.2 Sequential Search Algorithms

Sequential search algorithms have a complexity of $O(N^2)$. They add and/or delete features to/from the current subset sequentially. They usually use hill-climbing strategy for the search.

3.4.1.1.2.1 Sequential Forward Selection (SFS)

In SFS, Miller [Miller, 1990], search starts with an empty set. First, feature subsets with only one feature are evaluated and the best feature (f^*) is selected. Then two feature combinations of f^* with the other features are tested and the best feature subset is selected. The search goes on like that by adding one more feature at each step to the subset until we do not get any more performance improvement for the system.

For example, if we have 5 features $\{f_1, f_2, f_3, f_4, f_5\}$, we first test the single feature sets. Let's assume that f_3 gives the best classification rate. Then we will test two-featured subsets $\{f_3, f_1\}$, $\{f_3, f_2\}$, $\{f_3, f_4\}$ and $\{f_3, f_5\}$. And choose the one with the best performance. If that is $\{f_3, f_4\}$ and the classification rate of that subset is better than $\{f_3\}$ then we will test three-featured subsets $\{f_3, f_4, f_1\}$, $\{f_3, f_4, f_2\}$ and $\{f_3, f_4, f_5\}$. This is continued till we get no more performance improvement. We can also continue adding features one by one till we add all the features. At the end, we can choose the subset with the best classification rate. This will find a subset with better test set accuracy but it will also increase the complexity of the search. SFS algorithm requires $N + (N - 1) + (N - 2) + \dots + 2 + 1 = (N + 1)N/2$ subset evaluations at worst case. Therefore its complexity is $O(N^2)$.

3.4.1.1.2.2 Sequential Backward Selection (SBE)

In SBE, search starts from the complete feature set. If there are N features in the set, features subsets with $(N-1)$ features are evaluated and the best performing subset is chosen. If the performance of that subset is better than the set with N features, the subset with $(N-1)$ features is taken as the basis and its subsets with $(N-2)$ features are evaluated. This goes on like this till deleting a feature does not improve performance anymore. Complexity of the algorithm is $O(N^2)$.

3.4.1.1.3 Randomized Search Algorithms

Randomized algorithms include genetic algorithms (GA) and simulated annealing search methods. In GA approach, subsets are represented by binary strings of length N (N : number of features). Each string represents a chromosome. Each chromosome is evaluated

to find its fitness value. Fitness value determines if a chromosome will survive or die. New chromosomes are created by using crossover and mutation operations on the fittest chromosomes. In crossover, two parents exchange their parts to create children. In mutation, random bits of a chromosome are changed to create a new one.

3.4.1.2 Filter Approach

In filter approach, classification algorithm is not used in feature subset selection. Subsets are evaluated by other means. For example, some methods uses exhaustive breadth first search. It tries to find the feature subset with the minimum number of features which classifies the training set sufficiently.

3.4.1.3 Wrapper Approach

In wrapper approach, classification algorithm (such as backpropagation) is used as the evaluation function. The feature selection algorithm is wrapped around the classification algorithm. For each subset, a classifier (such as a neural network) is constructed and this classifier is used for evaluating that subset. The advantage of this approach is that it improves reliability of the evaluation function. The disadvantage is that it increases the cost of the evaluation function.

3.4.2 Feature Subset Selection By Feature Relevance

In real world application areas (like medical diagnosis) not only the accuracy but also the simplicity and comprehensibility is important. By deleting unnecessary features, we cope with the high dimensionality of the real-world dataset. Therefore learning becomes easier. The thesis has utilized a new feature subset selection method that select features by using sorted features relevance. This algorithm was utilized earlier, by Boz [Boz, 2000, 2002] as part of his Ph.D. dissertation at Lehigh University, in developing an extractor that convert trained neural networks into decision trees. The algorithm is divided into three phases, Sorted Search, Neighbor Search, and Finding Final Subset by Using Cross Validation. The sorted search phase sorts the features according to their relevance to the

trained RBP network. The neighbor search phase use the subset found in the first phase as a starting point and tries to find a better subset in the immediate neighbors. The final subset is found by using cross validation which is integrated to the algorithm.

3.4.2.1 Phase 1: Sorted Search Phase

- At each step a network with a reduced set of variables is used. The most relevant feature is the one that caused the least test set classification accuracy when it was removed from the network.
- Then, sorts the features according to their relevance for the classification. Features are sorted from the most relevant one (with the lowest accuracy) to the least relevant one.
- Then, a network is constructed by using the best feature (the most relevant one).
- The classification accuracy of the network on the test dataset is saved for that subset.
- Next, the best two features are tested, followed by the best three features and it goes like that till the best N features (N: numbers of features) are tested. For example, If the sorted list is like $\{f^1, f^2, \dots, f^N\}$. The method tests the subsets $\{f^1\}$, $\{f^1, f^2\}$, $\{f^1, f^2, f^3\}$, ..., $\{f^1, f^2, \dots, f^N\}$. We find the subset with the best test set accuracy and this subset will be the starting subset for the second search phase.

Sorted search phase can also be used by itself. It will be computationally more efficient because it tests at most N states (N: number of features). The danger is that if there are highly relevant random features or if none of the features are relevant this phase by itself may fail to find a good subset. If it is known that problem has nonrandom relevant features this phase alone will give reasonably good results by testing very few states.

3.4.2.2 Phase 2: Neighbor Search Phase

- In Neighbor Search Phase the best subset from the sorted search phase is assigned to the best state and to the current state. All the immediate neighbor states of the current state will be tested. For example, If the current state is [100110], then its neighbors are

[000110], [110110], [101110], [100010], [100100], [100111]. Each neighboring state has only 1 bit different from the current state.

- If a neighbor state is better than the best state (goodness measure is explained below), it is assigned to the best state. After testing all the neighbor states, if none of them is better than the best state, algorithm stops. Other stopping criteria are explained below in the rules list. If the best state has changed, then best state is assigned to current state and its neighbors are tested.
- This goes on till stopping criteria is met or there are no untested states around the current state. Algorithm keeps track of the previously tested states and does not test them again.
- To compare two states, choose the subset with better classification accuracy or if the classification accuracy is equal, choose the one with fewer features. If both classification ratios and the number of features are equal then choose the more relevant subset. Relevancy of a subset was calculated by using the ranking of each feature. Features are ranked by the end of the first phase. For example if we have 4 features and the features are ranked like {F4, F1, F3, F2}, (from most relevant to least), relevance. Then, the relevance of the state [1010] will be 5 ($3*1 + 2*1$) and the relevance of state [1001] will be 7 ($3*1 + 4*1$). Therefore, state [1001] will be more relevant than state [1010].
- If accuracy of the best subset at any point is 100%, there is no need to test subsets with higher number of features than the current subset.
- If more than one of the neighboring subsets are better than the best subset and if they have equal number of features, choose the more relevant subset. If that does not give any improvement (after testing its neighbors) go back to the previous state and test the next relevant subset.
- If there is only one feature in the best subset and if the accuracy is 100% stop the search.

3.4.2.3 Phase 3: Finding Final Subset Phase

The final best feature subset can be found by the following steps:

- In each fold, we find the best subset. (as mentioned above)

- For each feature, we find in how many folds that feature is a member of its best subset.
- Then, we find the average-times-in-best-subset value (total of times-in-best-subset values of all the features divided by the number of features).
- For the final feature subset, we choose the feature that appeared in subsets more than or equal the average-times-in-best-subset value.
- To test the final subset, we use the cross validation test sets in each fold. Then, we find the average of these test results.
- For comparing the results we also tested best feature subset at each fold on the test set of that fold.

An outline of the feature subset selection algorithm is given in Figure 3.4. This algorithm searches at most number of states equal to the number of features. So it will give reasonably good results by testing very few states. Complexity of the algorithm is $O(N)$.

```

// Sorted Search Phase
visitedSubSetList= emptySet; sortedList = emptySet;
N = numFeats(fullFeatureSet);
for (i=0; i < N; i++) {
    currentSubSet = fullFeatureSet – featurei
    Construct an RBP Network by using currentSubSet
    Test the RBP Network by using test set
    Find the classification accuracy (acci) of the test set
    Add the pair (featurei, acci) to the sortedList
    Add currentSubSet to the visitedSubSetList
}
sort the sortedList in ascending order according to test accuracy
(Now the sortedList is sorted from the most relevant feature to the least)
bestAcc = -1;
currentSubSet = emptySet; bestSubSet = emptySet;
for (i=0; i < N; i++) {
    Add the next most relevant feature from sortedList to the currentSubSet

```

```

Construct an RBP Network by using currentSubSet
Test the RBP Network by using test set
Find the classification accuracy (currentAcc) of the test set
if ((currentAcc >= bestAcc) {
    bestAcc = currentAcc;
    bestSubSet = currentSubSet;
}
Put currentSubSet into the visitedSubSetList
}
// Neighbor Search Phase
neighborList = emptySet;
currentSubSet = bestSubSet
Get the immediaste neighbors of the currentSubSet
While(Not STOP) {
If (all neighbors of the currentSubSet have already been visisted)
    STOP
for (i=0; i < N; i++) {
    if ( bestAcc == 100 AND numFeats(bestSubSet) == 1)
        STOP
    neighborSubSet = ith neighbor of the currentSubSet
    if(NOT (bestAcc == 100 AND
        (numFeats (currentSubSe). < numFeats (neighborSubSet)))){
        if(neighborSubSet is not in visitedSubSetList) {
            Put currentSubSet into the visitedSubSetList
            Construct an RBP Network by using currentSubSet
            Test the RBP Network by using test set
            Find the classification accuracy (acci) of the test set
            if ((acc > bestAcc) OR ((( acc == bestAcc) AND
                (numFeats(neighboSubSetr) < numFeats(bestSubSet))) OR
                ((acc == bestAcc) AND
                (numFeats(neighboSubSetr) == numFeats( bestSubSet)) AND
                (neighboSubSet is more relevant than bestSubSet)))) {
                bestAcc = currentAcc;
                bestSubSet = currentSubSet;

```

```

        }
    }
} //for
If (None of the neighbors is better than bestSubSet)
    STOP
} //while
Return bestSubSet

```

Figure 3.4. *Feature Subset Selection by Relevance Algorithm*

EVALUATION OF FRULEX APPROACH

This chapter presents the results of applying the proposed approach on a number of real-world case studies to evaluate the effectiveness of the different parts of the approach in fuzzy rules extraction for classification tasks. It provides a number of textual and graphical representations for the extracted fuzzy classifiers. Finally, it evaluates the proposed approach according to the evaluation criteria defined in Chapter 2.

4.1 Description of Case Studies

The experiments reported here used real-world case studies. The real-world case studies were obtained from the machine learning data repository at the University of California at Irvine, [Mertz and Murphy, 1992]. Table 4.1 presents a description of the case studies.

Table 4.1. *Description of Case Studies*

Case Study	Size	No. of Attributes	No. of Classes	Continuous Data	Discrete Data	Missing Data
Iris Flower Classification	150	4	3	✓	✗	✗
Wisconsin Breast Cancer	699	9	2	✗	✓	✓
Cleveland heart disease	303	13	2	✓	✓	✓
Pima Indians diabetes	768	8	2	✓	✗	✗

A variety of methods including Leave-One-Out Nearest Neighbor (LOONN), Cross Validation Nearest (XVNN), RULEX, Full-RE, FSM, NEFCLASS, Castellano's approach and C4.5 were chosen to provide comparative results for the proposed approach. The nearest neighbor methods were chosen because they are traditional statistical classifiers.

FSM, NEFCLASS and Castellano's approach were chosen because they are efficient neuro-fuzzy approaches, which are applied in the same domains.

The k-fold cross validation is part of our approach and it is used for finding the final feature subset in the simplification phase. K is user definable. User is also able to choose how many partitions of the dataset will be used for the training set, test set and cross validation set. The reporting experiments used 10(8-1-1) fold cross validation, that is, 8 of them for training (training set), 1 for testing (test set) and 1 for testing the final feature subsets (cross validation set).

4.2 Case Study 1: Iris Flower Classification Dataset

4.2.1 Description of Case Study

The classification problem of the Iris Flower data set [Mertz and Murphy, 1992] consists of classifying three species of iris flowers, namely, setosa, versicolor, and virginica. The dataset contains 150 instances, with 50 of each class. Each instance is described by four leaf attributes, namely, sepal length, sepal width, petal length, and petal width (See Table 4.2 and Table 4.3).

Table 4.2. Case Study 1: Classes

ID	Class
1	Setosa
2	Versicolor
3	Virginica

Table 4.3. Case Study 1: Features and Feature values

ID	Feature	Feature values
F1	Sepal length	[4.3, 7.9]
F2	Sepal width	[2.0, 4.4]
F3	Petal length	[1.0, 6.9]
F4	Petal width	[0.1, 2.5]

The performance of the extracted fuzzy classifier was measured by 10(8-1-1) fold cross-validation. This means that the whole dataset was divided into ten equally sized groups (each group consists of 15 samples randomly drawn from the three classes). One group was used as a test set to test the fuzzy classifier, another group used as a cross

validation test set to test the final feature subset, while the classifier was trained with the remaining 8 groups.

4.2.2 Initialization Phase

The SCRG method, described in Chapter 3, is used to determine the initial centers and widths of the membership functions of the input features. Table 4.4 summaries the results after applying the SCRG phase in the ten runs. (We have $B=4$, $K_0=1.0$, $K=1.0$, $\sigma_0=0.05$, $\rho = 0.0001$, and $\tau = 0.001$)

Table 4.4. Case Study 1: Results of the 10-fold cross validation after initialization

Iris Flower			After Initialization Phase					
			Training Set		Test Set		Average	
Run	Rules	Features	Acc.	Misclass.	Acc.	Misclass.	Acc.	Misclass.
1	3	4	94.17	7	93.33	1	93.75	4
2	3	4	95.00	6	93.33	1	94.17	3.5
3	3	4	94.17	7	100.00	0	97.09	3.5
4	3	4	95.83	5	93.33	1	94.58	3
5	3	4	95.83	5	93.33	1	94.58	3
6	3	4	95.83	5	93.33	1	94.58	3
7	3	4	95.83	5	93.33	1	94.58	3
8	3	4	94.17	7	100.00	0	97.09	3.5
9	3	4	95.00	6	100.00	0	97.50	3
10	3	4	94.17	7	86.67	2	90.42	4.5
avg.	3.00	4.00	95.00	6.00	94.67	0.80	94.83	3.4

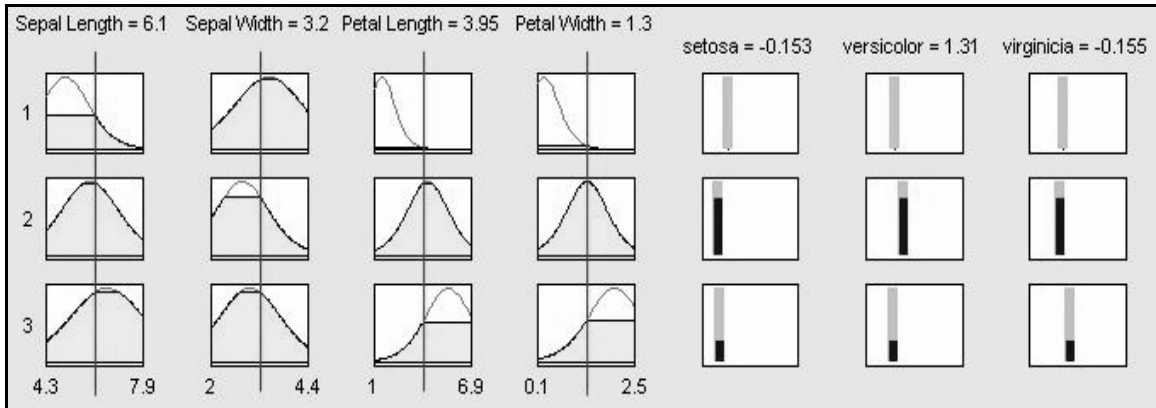
4.2.3 Optimization Phase

The backpropagation gradient descent learning method (Chapter 3) is used to optimize the FKB extracted in phase one. A Network with 4 inputs and 3 outputs, corresponding to the 3 classes, was constructed. Table 4.5 summaries the results obtained for this phase, after 100 epochs for the ten runs. (We have $\varepsilon = 0.01$, and $\eta = 1.0$)

Table 4.5. Case Study 1: Results of the 10-fold cross validation after optimization

Iris Flower			After Optimization Phase					
			Training Set		Test Set		Average	
Run	Rules	Features	Acc.	Misclass.	Acc.	Misclass.	Acc.	Misclass.
1	3	4	95.00	6	93.33	1	94.17	3.5
2	3	4	95.00	6	93.33	1	94.17	3.5
3	3	4	94.17	7	100.00	0	97.09	3.5
4	3	4	95.83	5	93.33	1	94.58	3
5	3	4	95.83	5	93.33	1	94.58	3
6	3	4	95.83	5	93.33	1	94.58	3
7	3	4	95.83	5	93.33	1	94.58	3
8	3	4	94.17	7	100.00	0	97.09	3.5
9	3	4	95.00	6	100.00	0	97.50	3
10	3	4	95.00	6	93.33	1	94.17	3.5
avg.	3.00	4.00	95.17	5.80	95.33	0.70	95.25	3.25

For the last run of the 10 trials, Figure 4.1 shows the graphical representation of the FKB obtained, after the optimization phase. (Using *MATLAB Fuzzy Toolbox*)

**Figure 4.1.** Case Study 1: Graphical representation of FRB obtained after optimization

4.2.4 Simplification Phase

Feature Subset Selection by Relevance method, described in Chapter 3, is used to simplify the FRB extracted in phase one. Table 4.6 and Table 4.7 have summarized the results obtained for this phase for the ten trials.

Table 4.6. *Case Study 1: Results of 10-fold cross validation after sorted and neighbor search*

Iris Flower			After Sorted Search & Neighbor Search Phase						
			Best Feature Set	Training Set		Test Set		Average	
Run	Rules	Features		Acc.	Mis.	Acc.	Mis.	Acc.	Mis.
1	3	1	F4	95.83	5	100.00	0	97.92	2.5
2	3	1	F4	93.33	8	93.33	1	93.33	4.5
3	3	1	F4	95.00	6	100.00	0	97.50	3
4	3	1	F4	96.67	4	93.33	1	95.00	2.5
5	3	1	F4	97.50	3	93.33	1	95.42	2
6	3	3	F1,F2,F3	90.00	12	100.00	0	95.00	6
7	3	1	F4	96.67	4	93.33	1	95.00	2.5
8	3	1	F4	95.00	6	100.00	0	97.50	3
9	3	1	F4	95.00	6	100.00	0	97.50	3
10	3	1	F4	93.33	8	100.00	0	96.67	4
avg.	3.00	1.2	F4,F3	94.83	6.20	97.33	0.40	96.08	3.30

Table 4.7. *Case Study 1: Results of the 10-fold cross validation after simplification*

Iris Flower			After Simplification Phase						
			Final Feature Set	Training Set		XV Test Set		Average	
Run	Rules	Features		Acc.	Mis.	Acc.	Mis.	Acc.	Mis.
1	3	2	F3,F4	95.83	5	100.00	0	97.92	2.5
2	3	2	F3,F4	96.67	4	93.33	1	95.00	2.5
3	3	2	F3,F4	95	6	100.00	0	97.50	3
4	3	2	F3,F4	95.83	5	93.33	1	94.58	3
5	3	2	F3,F4	97.5	3	93.33	1	95.42	2
6	3	2	F3,F4	97.5	3	93.33	1	95.42	2
7	3	2	F3,F4	95.83	5	93.33	1	94.58	3
8	3	2	F3,F4	95	6	100.00	0	97.50	3
9	3	2	F3,F4	96.67	4	100.00	0	98.34	2
10	3	2	F3,F4	95.83	5	93.33	1	94.58	3
avg.	3.00	2	F3,F4	96.17	4.60	96.00	0.60	96.08	2.60

For the first run of the ten trials, Figure 4.2 shows the performance of the networks constructed by the successive removal of input features, Figure 4.3 shows the performance of the networks constructed by the successive addition of the relevant features, and Figure 4.4 and Figure 4.5 show the graphical and textual representation of the obtained FKB.

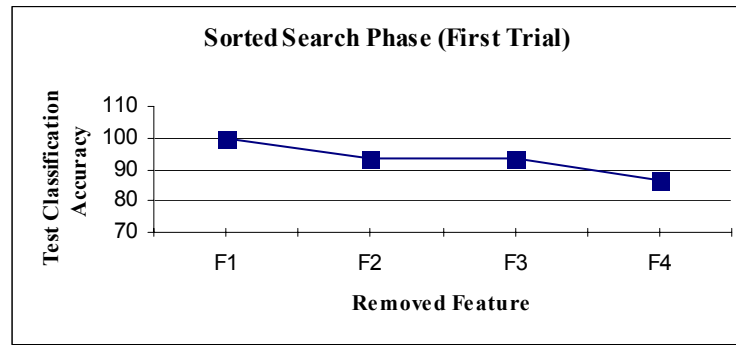


Figure 4.2. Case Study 1: Performance of RBPN during removal of input features

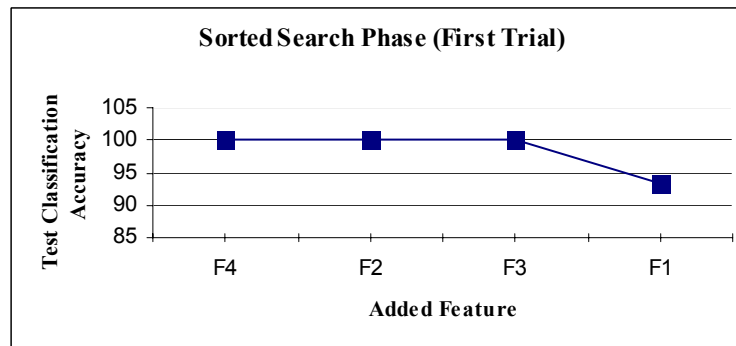


Figure 4.3. Case Study 1: Performance of the RBPN with different features

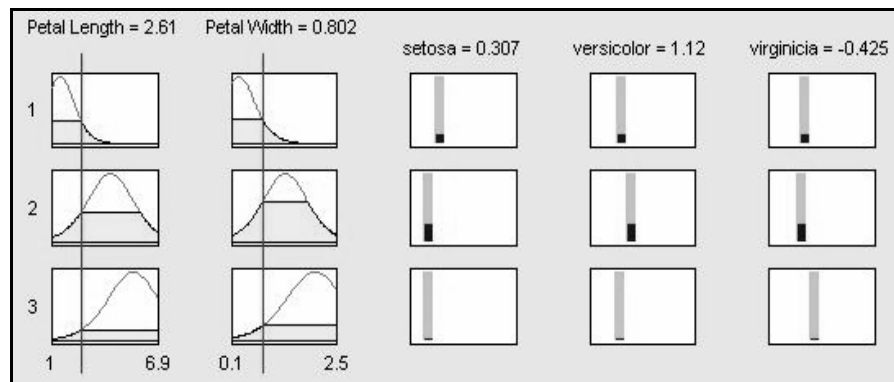


Figure 4.4. Case Study 1: Graphical representation of the FRB obtained after simplification

Rule 1:	<i>IF ('Petal Length' IS in3mf1) AND ('Petal Width' IS in4mf1), THEN ('setosa' IS out1mf1) AND ('versicolor' IS out2mf1) AND ('versicolor' IS out3mf1)</i>		
Rule 2:	<i>IF ('Petal Length' IS in3mf2) AND ('Petal Width' IS in4mf2), THEN ('setosa' IS out1mf2) AND ('versicolor' IS out2mf2) AND ('versicolor' IS out3mf2)</i>		
Rule 3:	<i>IF ('Petal Length' IS in3mf3) AND ('Petal Width' IS in4mf3), THEN ('setosa' IS out1mf3) AND ('versicolor' IS out2mf3) AND ('versicolor' IS out3mf3)</i>		
Where:	$\begin{aligned} in3mf1 &= ridgemf(x3; 0.4759, 1.4600, 2.1014) \\ in3mf2 &= ridgemf(x3; 0.7697, 4.2325, 1.2992) \\ in3mf3 &= ridgemf(x3; 0.8636, 5.5025, 1.1579) \\ in4mf1 &= ridgemf(x4; 0.2354, 0.2475, 4.2473) \\ in4mf2 &= ridgemf(x4; 0.3107, 1.3175, 3.2189) \\ in4mf3 &= ridgemf(x4; 0.4024, 2.0025, 2.4852) \\ out1mf1 &= 1.388 & out1mf2 &= -0.1760 & out1mf3 &= -0.0918 \\ out2mf1 &= -0.2546 & out2mf2 &= 2.0533 & out2mf3 &= -0.7655 \\ out3mf1 &= -0.1334 & out3mf2 &= -0.8773 & out3mf3 &= 1.8573 \end{aligned}$		

Figure 4.5. Case Study 1: Textual representation of the FRB obtained after simplification

4.2.5 Analysis of Results

The ten-fold cross validation results are summarized in Table 4.8 and Figure 4.6. To evaluate the effectiveness of classification and rule extraction, the proposed approach was compared with other statistical, neural and rule-based classifiers developed for the same dataset, as shown in Table 4.9, Table 4.10 and Table 4.11.

Table 4.8. Case Study 1: Summary of Classification results of FRULEX

Iris Flower		Train	Test	Average
Phase 1	Misclassified	6.0	0.8	3.4
	Accuracy	95 %	94.67 %	94.83
Phase 2	Misclassified	5.8	0.7	3.25
	Accuracy	95.17 %	95.33 %	95.25 %
Phase 3	Misclassified	4.6	0.6	2.6
	Accuracy	96.17 %	96 %	96.08 %

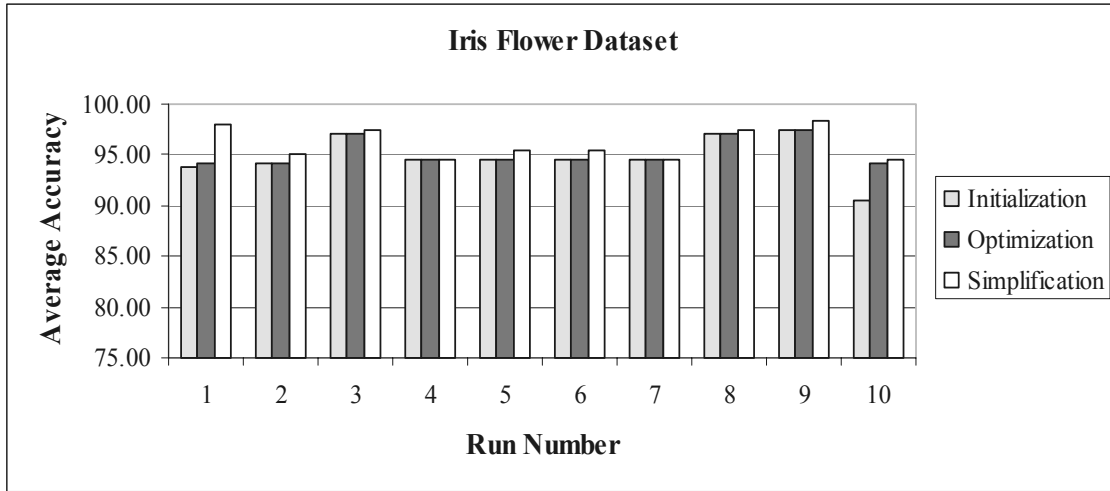


Figure 4.6. Case Study 1: Summary of Classification results of FRULEX

Table 4.9. Case Study 1: Statistical and Neural Classifiers

Method	Classification Accuracy	Reference
LOONN	95.3%	[Andrews and Geva, 1994]
XVNN	96%	[Andrews and Geva, 1994]
RBF network	97.36%	[Ster <i>et al.</i> , 1996]

- LOONN, XVNN and RBF network have achieved accuracy 95.3%, 96% and 97.36% respectively. However, they are black-boxes as they do not provide any explanation to their decisions and have not any human-readable representation to their hidden knowledge. Reasoning with logical rules is more acceptable to human users than recommendations given by black box systems, because such reasoning is comprehensible, provides explanations, and may be validated, increasing confidence in the system.

Table 4.10. Case Study 1: Crisp Rule-Based Classifiers

Method	Classification Accuracy	Extracted Rules	Antecedents Per rule	Reference
Full-RE	97.33%	3 crisp rules	1 to 2	[Taha and Ghosh, 1996a]
NeuroRule	98%	3 crisp rules	1	[Taha and Ghosh, 1996a]
KT	97.33%	5 crisp rules	1 to 4	[Taha and Ghosh, 1996a]
RULEX	94.0%	5 crisp rules	3	[Andrews <i>et al.</i> , 1995]

- Full-RE has achieved a high accuracy (97.33%) and has extracted three crisp rules with a maximum of two conditions per rule.
- NeuroRule has achieved a high accuracy (98%) and has extracted three crisp rules with one condition per rule.
- KT has achieved a high accuracy (97.33%) and has extracted five crisp rules with a maximum of four conditions per rule.
- RULEX has achieved accuracy (94.0%) using RBP network but it does not allow network to produce overlapping local response units. If the local response units are allowed to overlap and an input pattern fill in the region of overlap is presented, more than one unit will show significant activation and the pattern will be classified by the network, but when the individual units are decompiled into rules, these rules may not account for the patterns that lie in the region of overlap. Avoid overlapping leads to suboptimal solutions.
- The crisp rule-based classifiers can achieve higher accuracy. However, providing a black-and-white picture where the user needs additional information since only one class label is identified as the correct one. For medical diagnosis, physicians may wish to quantify “how severe the disease is” with numbers in $[0, 1]$.

Table 4.11. *Case Study 1: Fuzzy Rule-Based Classifiers*

Method	Classification Accuracy	Extracted Rules	Conditions Per rule	Reference
NEFCLASS	96.7%	7 fuzzy rules	4	[Nauck et al., 1996]
NEFCLASS	96.7%	4 fuzzy rules	1 to 2	[Nauck et al., 1999]
<i>FRULEX</i>	96%	3 fuzzy rules	2	Proposed Approach

- Fuzzy rule-based classifiers provide a good platform to deal with uncertain, noisy, imprecise or incomplete information. They provide a gray picture where the user can gain further information. For medical diagnosis, physicians can quantify “how severe the disease is”. For pattern classification, user can quantify “how typical this pattern is”.
- The NEFCLASS method has also been applied to this data [Nauck et al., 1996]. The system was initialized with fuzzy clustering method and used trapezoidal membership

functions per input feature. Using 7 rules gave 96.7% correct answers, showing the usefulness of prior knowledge from initial clustering. It should be noted that our approach achieves high accuracy (96.0%) on the test set with an average of 2 input variables and 3 fuzzy rules with respect to the 4 features and 7 fuzzy rules used by NEFCLASS, thus resulting in a more simple and interpretable fuzzy classifier.

4.3 Case Study 2: Wisconsin Breast Cancer Dataset

4.3.1 Description of Case Study

The Wisconsin breast cancer dataset (*WBCD*) [Mertz and Murphy, 1992] contains 699 instances, with 458 benign (65.5%) and 241 (34.5%) malignant cases (see Table 4.12). Nine features with integer value in the range are used for each instance (See Table 4.13). For 16 instances one attribute is missing (it was replaced by an average value).

Table 4.12. Case Study 2: Classes

ID	Class
1	Benign
2	Malignant

Table 4.13. Case Study 2: Features and Feature values

ID	Feature	Feature values
F1	Clump thickness	[1, 10]
F2	Uniformity of cell size	[1, 10]
F3	Uniformity of cell shape	[1, 10]
F4	Marginal adhesion	[1, 10]
F5	Single epithelial cell size	[1, 10]
F6	Bare nuclei	[1, 10]
F7	Bland chromatin	[1, 10]
F8	Normal nucleoli	[1, 10]
F9	Mitoses	[1, 10]

To estimate the performance of the FKB extracted by the proposed approach, 10-fold cross-validation was carried out. The whole dataset was divided into 10 equally sized groups (a group consists of 70 samples randomly drawn from the two classes). One group was used as a test set to test the fuzzy classifier, another group used as a cross validation

test set to test the final feature subset, while the classifier was trained with the remaining 8 groups.

4.3.2 Initialization Phase

The SCRG method, described in Chapter 3, is used to determine the initial centers and widths of the membership functions of the input features. Table 4.14 summaries the results obtained after applying the SCRG phase for the ten trails. (We have $B=9$, $K_o=1.0$, $K=1.0$, $\sigma_o=0.05$, $\rho = 0.0001$, and $\tau = 0.01$)

Table 4.14. Case Study 2: Results of the 10-fold cross validation after initialization

WBCD			After Initialization Phase					
			Training Set		Test Set		Average	
Run	Rules	Features	Acc.	Misclass.	Acc.	Misclass.	Acc.	Misclass.
1	2	9	96.96	17	94.37	4	95.67	10.5
2	2	9	96.43	20	97.14	2	96.79	11
3	2	9	96.60	19	95.71	3	96.16	11
4	2	9	96.96	17	91.43	6	94.20	11.5
5	2	9	96.24	21	98.57	1	97.41	11
6	2	9	96.24	21	97.14	2	96.69	11.5
7	2	9	96.96	17	98.57	1	97.77	9
8	2	9	96.60	19	98.57	1	97.59	10
9	2	9	96.43	20	97.10	2	96.77	11
10	2	9	96.96	17	97.10	2	97.03	9.5
avg.	2.00	9	96.64	18.80	96.57	2.40	96.60	10.6

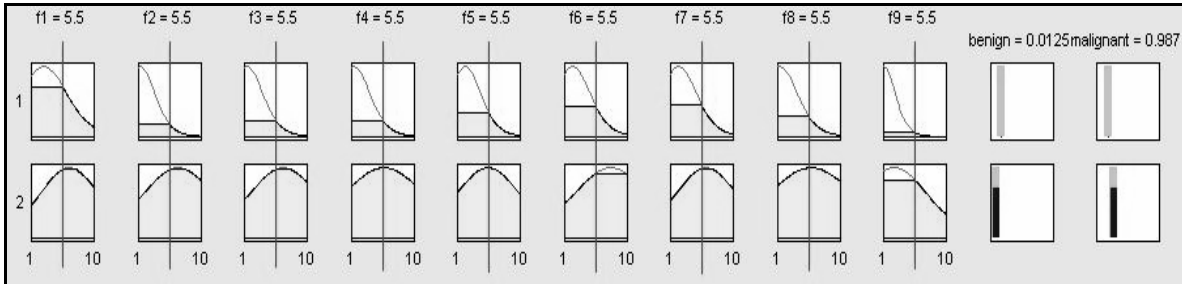
4.3.3 Optimization Phase

The gradient-descent backpropagation learning method, described in Chapter 3, is used to optimize the FRB extracted in phase one. A network with 9 inputs and 2 outputs, corresponding to the two classes, was constructed. Table 4.15 summaries the results obtained after 100 epochs for the ten trials. (We have $\varepsilon = 0.01$, and $\eta = 1.0$)

Table 4.15. Case Study 2: Results of the 10-fold cross validation after optimization

WBCD			After Optimization Phase					
			Training Set		Test Set		Average	
Run	Rules	Features	Acc.	Misclass.	Acc.	Misclass.	Acc.	Misclass.
1	2	9	96.96	17	94.37	4	95.67	10.5
2	2	9	96.25	21	98.57	1	97.41	11
3	2	9	96.25	21	98.57	1	97.41	11
4	2	9	97.14	16	91.43	6	94.29	11
5	2	9	96.42	20	98.57	1	97.50	10.5
6	2	9	96.42	20	97.14	2	96.78	11
7	2	9	97.14	16	98.57	1	97.86	8.5
8	2	9	96.60	19	98.57	1	97.59	10
9	2	9	96.25	21	97.10	2	96.68	11.5
10	2	9	96.96	17	97.10	2	97.03	9.5
avg.	2.00	9	96.64	18.80	97.00	2.10	96.82	10.45

For the sixth run of the ten trials, Figure 4.7 shows the graphical representation of the FRB obtained. (Using *MATLAB Fuzzy Toolbox*)

**Figure 4.7.** Case Study 2: Graphical representation of the FRB obtained after optimization

4.3.4 Simplification Phase

Feature Subset Selection by Relevance method, described in Chapter 3, is used to simplify the FRB extracted in phase one. Table 4.16 and Table 4.17 have summarized the results obtained after this phase for the ten trials.

Table 4.16. *Case Study 2: Results of 10-fold cross validation after sorted and neighbor search*

WBCD			After Sorted Search & Neighbor Search Phases						
Run	Rules	Features	Best Feature Set	Training Set		Test Set		Average	
				Acc.	Mis.	Acc.	Mis.	Acc.	Mis.
1	2	3	{F3,F8,F9}	95.17	27	95.77	3	95.47	15
2	2	6	{F1,F2,F3,F6,F7,F8}	96.79	18	98.57	1	97.68	9.5
3	2	3	{F1,F2,F3}	94.28	32	97.14	2	95.71	17
4	2	2	{F3,F5}	94.10	33	92.86	5	93.48	19
5	2	2	{F1,F6}	93.92	34	100.00	0	96.96	17
6	2	4	{F1,F3,F6,F7}	95.53	25	97.14	2	96.34	13.5
7	2	6	{F1,F2,F3,F5,F6,F9}	96.24	21	100.00	0	98.12	10.5
8	2	2	{F1,F2}	94.10	33	98.57	1	96.34	17
9	2	1	{F2}	93.04	39	98.55	1	95.80	20
10	2	2	{F2,F4}	93.56	36	98.55	1	96.06	18.5
avg.	2.00	3.1	{F1,F2,F3,F6}	94.67	29.80	97.72	1.60	96.19	15.70

Table 4.17. *Case Study 2: Results of the 10-fold cross validation after simplification*

WBCD			After Simplification Phase						
Run	Rules	Features	Final Feature Set	Training Set		XV Test Set		Average	
				Acc.	Mis.	Acc.	Mis.	Acc.	Mis.
1	2	4	{F1,F2,F3,F6}	96.96	17	92.96	5	94.96	11
2	2	4	{F1,F2,F3,F6}	95.89	23	95.71	3	95.80	13
3	2	4	{F1,F2,F3,F6}	96.42	20	97.14	2	96.78	11
4	2	4	{F1,F2,F3,F6}	97.32	15	91.43	6	94.38	10.5
5	2	4	{F1,F2,F3,F6}	96.78	18	97.14	2	96.96	10
6	2	4	{F1,F2,F3,F6}	96.78	18	97.14	2	96.96	10
7	2	4	{F1,F2,F3,F6}	97.32	15	97.14	2	97.23	8.5
8	2	4	{F1,F2,F3,F6}	96.42	20	98.57	1	97.50	10.5
9	2	4	{F1,F2,F3,F6}	95.89	23	98.55	1	97.22	12
10	2	4	{F1,F2,F3,F6}	96.96	17	97.10	2	97.03	9.5
avg.	2.00	4	{F1,F2,F3,F6}	96.67	18.60	96.29	2.60	96.48	10.60

For the sixth run of the ten trials, Figure 4.8 shows the performance of the networks constructed by the successive removal of input features, Figure 4.9 shows the performance of the networks constructed by the successive addition of the relevant features, and Figure 4.10 and Figure 4.11 show the graphical and textual representation of the FKB obtained, respectively. (*Using MATLAB Fuzzy Toolbox*)

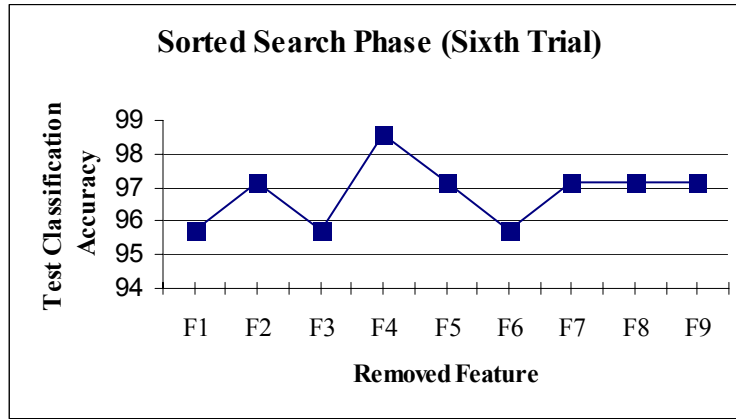


Figure 4.8. Case Study 2: Performance of RBPN during removal of input features

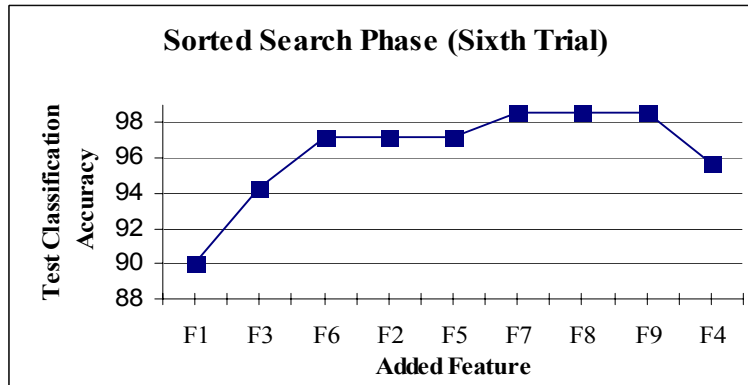


Figure 4.9. Case Study 2: Performance of the RBPN with different features

Rule 1:	IF ('Climp thickness' IS in1mf1) AND ('Uniformity of cell size' IS in2mf2) AND ('Uniformity of cell shape' IS in3mf1) AND ('Bar nuclei' IS in6mf1), THEN ('benign' IS out1mf1) AND ('malignant' IS out2mf1)
Rule 2:	IF ('Climp thickness' IS in1mf2) AND ('Uniformity of cell size' IS in2mf2) AND ('Uniformity of cell shape' IS in3mf2) AND ('Bar nuclei' IS in6mf2), THEN ('benign' IS out1mf2) AND ('malignant' IS out2mf2)
Where:	$\text{in1mf1} = \text{ridgemf}(x1; 2.0201, 2.8123, 0.4950)$ $\text{in1mf2} = \text{ridgemf}(x1; 2.6591, 6.4326, 0.3761)$
	$\text{in2mf1} = \text{ridgemf}(x2; 1.2876, 1.2703, 0.7766)$ $\text{in2mf2} = \text{ridgemf}(x2; 3.1604, 6.6579, 0.3164)$

$$\begin{aligned}
in3mf1 &= \text{ridgemf}(x3; 1.4049, 1.3904, 0.7118) \\
in3mf2 &= \text{ridgemf}(x3; 3.0527, 6.5595, 0.3276) \\
in6mf1 &= \text{ridgemf}(x6; 1.5428, 2.1656, 0.6482) \\
in6mf2 &= \text{ridgemf}(x6; 3.3604, 7.6497, 0.2976) \\
out1mf1 &= 1.1440 & , out1mf2 &= 0.0125 \\
out2mf1 &= -0.1440 & , out2mf2 &= 0.9875
\end{aligned}$$

Figure 4.10. Case Study 2: Textual Representation of the FRB obtained after simplification

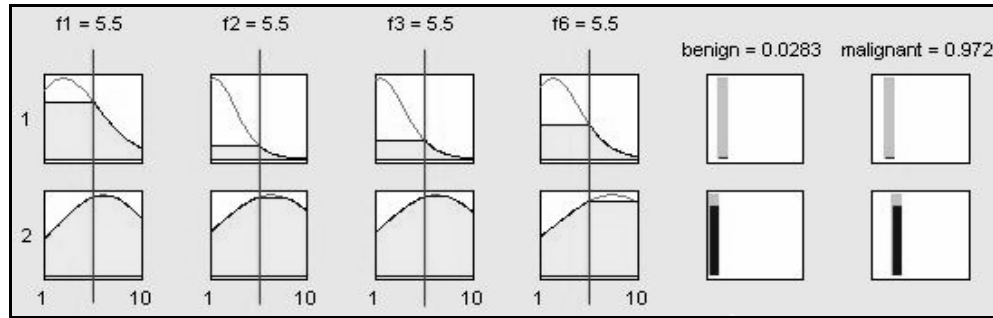


Figure 4.11. Case Study 2: Graphical representation of the FRB obtained after simplification

4.3.5 Analysis of Results

The ten-fold cross validation results are summarized in Table 4.18 and Figure 4.12. To evaluate the effectiveness of classification and rule extraction, the proposed approach was compared with other statistical, neural and rule-based classifiers developed for the same dataset, as shown in Table 4.19, Table 4.20 and Table 4.21.

Table 4.18. Case Study 2: Summary of Classification results of FRULEX

	WBCD	Train	Test	Average
Phase 1	Misclassified	18.8	2.4	10.6
	Accuracy	96.64 %	96.57 %	96.6 %
Phase 2	Misclassified	18.8	2.1	10.45
	Accuracy	96.64 %	97 %	96.82 %
Phase 3	Misclassified	18.6	2.6	10.6
	Accuracy	96.67 %	96.29 %	96.48 %

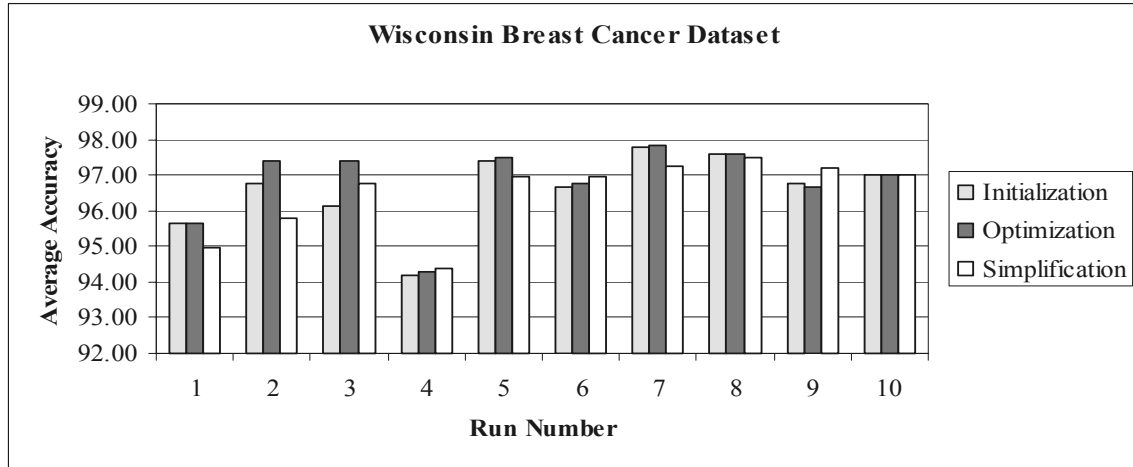


Figure 4.12. Case Study 2: Summary of Classification results of FRULEX

Table 4.19. Case Study 2: Statistical and Neural Classifiers

Method	Classification Accuracy	Reference
LOONN	95.6%	[Andrews and Geva, 1994]
XVNN	95.3%	[Andrews and Geva, 1994]
RBF	96.7%	[Ster <i>et al.</i> , 1996]

- LOONN, XVNN and RBF network have achieved accuracy **95.6%**, **95.3%** and **96.7%** respectively. However, they are black-boxes as they do not provide any explanation to their decisions and have not any human-readable representation to their hidden knowledge. Reasoning with logical rules is more acceptable to human users than recommendations given by black box systems, because such reasoning is comprehensible, provides explanations, and may be validated, increasing confidence in the system.

Table 4.20. Case Study 2: Crisp Rule-Based Classifiers

Method	Classification Accuracy	Extracted Rules	Conditions Per rule	Reference
Full-RE	96.19%	5 crisp rules	2	[Taha and Ghosh, 1996a]
NeuroRule	97.21%	4 crisp rules	4	[Taha and Ghosh, 1996a]
C4.5	97.21%	7 crisp rules	3	[Taha and Ghosh, 1996a]
SSV	96.3%	3 crisp rules	9	[Duch <i>et al.</i> , 2001]
RULEX	94.4%	5 crisp rules	4-5	[Andrews <i>et al.</i> , 1995]

- Full-RE has achieved a high accuracy (**96.19%**) and has extracted five crisp rules with a maximum of two conditions per rule.
- NeuroRule has achieved a high accuracy (**97.21%**) and has extracted three crisp rules with one condition per rule.
- RULEX has achieved a high accuracy (**94.4%**) and has extracted five crisp rules with a maximum of five conditions per rule.
- The crisp rule-based classifiers can achieve higher accuracy. However, providing a black-and-white picture where the user needs additional information since only one class label is identified as the correct one. For medical diagnosis, physicians may wish to quantify “how severe the disease is”.

Table 4.21. Case Study 2: Fuzzy Rule-Based Classifiers

Method	Classification Accuracy	Extracted Rules	Conditions Per rule	Reference
Castellano's Method	96.08%	4 fuzzy rules	4	[Castellano <i>et al.</i> , 2000]
FSM	96.5%	12 fuzzy rules	9	[Duch <i>et al.</i> , 2001]
NEFCLASS	96.2%	4 fuzzy rules	8	[Nauck <i>et al.</i> , 1996]
NEFCLASS	95.06%	2 fuzzy rules	5-6	[Nauck <i>et al.</i> , 1999]
<i>FRULEX</i>	96.48 %	2 fuzzy rules	4	Proposed Approach

- Fuzzy rule-based classifiers provide a good platform to deal with uncertain, noisy, imprecise or incomplete information. They provide a gray picture where the user can gain further information. For medical diagnosis, physicians can quantify “how severe the disease is”. For pattern classification, user can quantify “how typical this pattern is”.
- The NEFCLASS method has also been applied to this data [Nauck *et al.*, 1996], removing 16 instances with missing values. The system was initialized with fuzzy clustering method and used trapezoidal membership functions per input feature. Using 4 rules and the “best per class” rule learning (that can be viewed as a kind of pruning strategy), NEFCLASS achieves 8 errors on the training set (**97.66%** correct) and 18 errors on the test set (**94.72%** correct) and 26 errors (**96.2%** correct) on the whole set, showing the usefulness of prior knowledge from initial clustering. It should be noted that in our approach higher accuracy (**96.29%**) on the test set (generalization ability) is

achieved with an average of 4 input variables and 2 fuzzy rules with respect to the 8 features and 4 fuzzy rules used by NEFCLASS, thus resulting in a more simple and interpretable fuzzy classifier. Also, our results come from the application of procedures that do not require human intervention unlike NEFCLASS.

- FSM method has generated 12 fuzzy rules with Gaussian membership functions, providing **97.8%** on the training and **96.5%** on the test set part in 10-fold cross validation tests. FSM pursue accuracy as ultimate goal and take no care about the interpretability of the extracted knowledge.

4.4 Case Study 3: Cleveland Heart Disease Dataset

4.4.1 Description of Case Study

The Cleveland heart disease dataset [Mertz and Murphy, 1992] (collected at Cleveland Clinic Foundation by R. Detrano) contains 303 instances, with 164 healthy (54.1%) instances, the rest are heart disease instances of various severity (See Table 4.22). While the database has 76 raw attributes, only 13 of them are actually used in machine learning tests, including six continuous features and four discrete values (See Table 4.23).

Table 4.22. Case Study 3: Classes

ID	Class
1	Healthy
2	Heart disease

Table 4.23. Case Study 3: Features and Feature values

ID	Feature	Feature values
F1	Age	Continuous
F2	Sex	0,1 (male, female)
F3	Chest pain type	0,1,2,3 (typical angina, atypical angina, non angina, asymptomatic angina)
F4	Resting blood pressure	Continuous
F5	Serum cholesterol	continuous
F6	Fasting blood sugar	0,1 (yes, no)
F7	Resting ECG results	{0,1,2}
F8	Maximum heart rate	Continuous
F9	Exercise induced angina	0,1 (yes, no)

F10	Peak depression	Continuous
F11	Slope of ST segment	0,1,2 (up sloping, flat, down sloping)
F12	Number of major vessels	0,1,2,3
F13	Thal	3,6,7 (normal, fixed defect, reversible defect)

To estimate the performance of the FKB extracted by the proposed approach, we carried out a 10-fold cross-validation. The whole dataset was divided into 10 equally sized parts (a part consists of 30 samples randomly drawn from the two classes). One part was used as a test set to test the fuzzy classifier, another part used as a cross validation test set to test the final feature subset, while the classifier was trained with the remaining 8 parts.

4.4.2 Initialization Phase

The SCRG method, described in Chapter 3, is used to determine the initial centers and widths of the membership functions of the input features. Table 4.24 summaries the results after applying the SCRG phase for the ten trials. ($B=13$, $K_o=1.0$, $K=1.0$, $\sigma_o=0.05$, $\rho = 0.0001$, and $\tau=0.01$)

Table 4.24. Case Study 3: Results of 10-fold cross validation after initialization

Heart Disease			After Initialization Phase					
			Training Set		Test Set		Average	
Run	Rules	Features	Acc.	Misclass.	Acc.	Misclass.	Acc.	Misclass.
1	2	13	85.60	35	77.42	7	81.51	21
2	2	13	84.30	38	74.19	8	79.25	23
3	2	13	82.64	42	83.87	5	83.26	23.5
4	2	13	81.82	44	93.55	2	87.69	23
5	2	13	83.13	41	73.33	8	78.23	24.5
6	2	13	83.13	41	73.33	8	78.23	24.5
7	2	13	81.82	44	90.00	3	85.91	23.5
8	2	13	82.64	42	90.00	3	86.32	22.5
9	2	13	84.30	38	76.67	7	80.49	22.5
10	2	13	85.60	35	82.76	5	84.18	20
avg.	2.00	13	83.50	40.00	81.51	5.60	82.51	22.8

4.4.3 Optimization Phase

The backpropagation gradient descent learning method, described in Chapter 3, is used to optimize the FRB extracted in phase one. A network with 13 inputs and 2 outputs,

corresponding to the two classes, was constructed. Table 4.25 summaries the results obtained after 100 epochs for the ten trials. ($\epsilon = 0.01$, and $\eta = 1.0$)

Table 4.25. Case Study 3: Results of 10-fold cross validation after optimization

Heart Disease			After Optimization Phase					
			Training Set		Test Set		Average	
Run	Rules	Features	Acc.	Misclass.	Acc.	Misclass.	Acc.	Misclass.
1	2	13	85.60	35	80.65	6	83.13	20.5
2	2	13	86.36	33	83.87	5	85.12	19
3	2	13	83.47	40	83.87	5	83.67	22.5
4	2	13	83.06	41	93.55	2	88.31	21.5
5	2	13	86.01	34	76.67	7	81.34	20.5
6	2	13	86.01	34	76.67	7	81.34	20.5
7	2	13	83.06	41	86.67	4	84.87	22.5
8	2	13	83.47	40	86.67	4	85.07	22
9	2	13	86.36	33	76.67	7	81.52	20
10	2	13	85.60	35	86.21	4	85.91	19.5
avg.	2.00	13	84.90	36.60	83.15	5.10	84.03	20.85

For the tenth run of the ten trials, Figure 4.13 shows the graphical representation of the FKB obtained, after optimization phase. (Using *MATLAB Fuzzy Toolbox*)

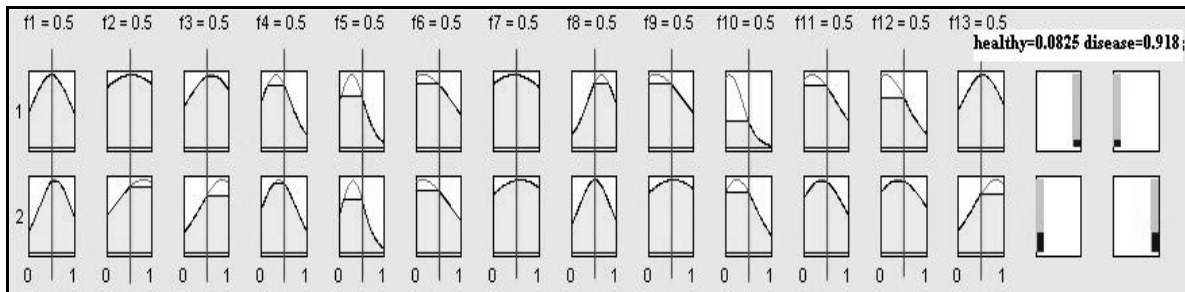


Figure 4.13. Case Study 3: Graphical representation of the FRB obtained after optimization

4.4.4 Simplification Phase

Feature Subset Selection by Relevance method, described in Chapter 3, is used to simplify the FKB extracted in phase one. Table 4.26 and Table 4.27 have summarized the results obtained after this phase for the ten trials.

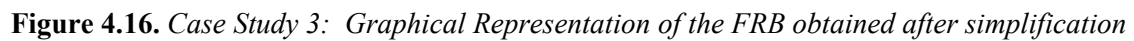
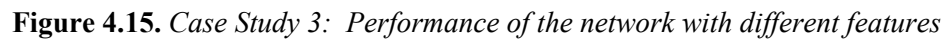
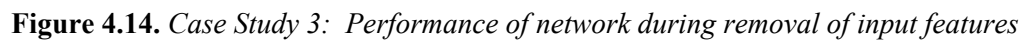
Table 4.26. *Case Study 3: Results of 10-fold cross validation after sorted and Neighbor Search*

Heart Disease			After Sorted Search & Neighbor Search Phases						
Run	Rules	Feat.	Best Feature Set	Training Set		Test Set		Average	
				Acc.	Mis.	Acc.	Mis.	Acc.	Mis.
1	2	8	{F1,F2,F3,F5,F6,F7,F10,F12}	81.48	45	87.10	4	84.29	24.5
2	2	2	{F3,F13}	76.86	56	87.10	4	81.98	30
3	2	6	{F2,F3,F4,F6,F10,F12}	78.51	52	93.55	2	86.03	27
4	2	4	{F3,F11,F12,F13}	82.64	42	96.77	1	89.71	21.5
5	2	3	{F8,F10,F12}	76.54	57	90.00	3	83.27	30
6	2	4	{F2,F3,F9,F11}	76.54	57	86.67	4	81.61	30.5
7	2	11	{F1,F2,F3,F4,F5,F7,F9,F10,F11,F12,F13}	82.64	42	83.33	5	82.99	23.5
8	2	3	{F3,F8,F12}	77.69	54	93.33	2	85.51	28
9	2	4	{F2,F6,F9,F13}	75.62	59	83.33	5	79.48	32
10	2	3	{F3,F9,F12}	78.19	53	89.66	3	83.93	28
avg.	2.00	4.8	{F2,F3,F9,F10,F12,F13}	78.67	51.70	89.08	3.30	83.88	27.5

Table 4.27. *Case Study 3: Results of 10-fold cross validation after simplification*

Heart Disease			After Simplification Phase						
Run	Rules	Feat.	Final Feature Set	Training Set		XV Test Set		Average	
				Acc.	Mis.	Acc.	Mis.	Acc.	Mis.
1	2	6	{F2,F3,F9,F10,F12,F13}	82.72	42	83.87	5	83.30	23.5
2	2	6	{F2,F3,F9,F10,F12,F13}	83.88	39	77.42	7	80.65	23
3	2	6	{F2,F3,F9,F10,F12,F13}	81.82	44	83.87	5	82.85	24.5
4	2	6	{F2,F3,F9,F10,F12,F13}	83.06	41	90.32	3	86.69	22
5	2	6	{F2,F3,F9,F10,F12,F13}	83.13	41	76.67	7	79.90	24
6	2	6	{F2,F3,F9,F10,F12,F13}	83.13	41	73.33	8	78.23	24.5
7	2	6	{F2,F3,F9,F10,F12,F13}	83.06	41	73.33	8	78.20	24.5
8	2	6	{F2,F3,F9,F10,F12,F13}	81.82	44	93.33	2	87.58	23
9	2	6	{F2,F3,F9,F10,F12,F13}	83.88	39	80	6	81.94	22.5
10	2	6	{F2,F3,F9,F10,F12,F13}	82.72	42	86.21	4	84.47	23
avg.	2.00	6	{F2,F3,F9,F10,F12,F13}	82.92	41.40	81.84	5.50	82.38	23.45

For the tenth run of the ten trials, Figure 4.14 shows the performance of the networks constructed by the successive removal of input features, Figure 4.15 shows the performance of the networks constructed by the successive addition of the relevant features, and Figure 4.16 and Figure 4.17 show the graphical and textual representation of the FKB obtained, respectively.



Rule 1:	$IF (F2 \text{ IS } in2mf1) \text{ AND } (F3 \text{ IS } in3mf1) \text{ AND } (F9 \text{ IS } in9mf1)$ $AND (F10 \text{ IS } in10mf1) AND (F12 \text{ IS } in12mf1) \text{ AND } (F13 \text{ IS } in13mf1),$ $THEN ('healthy' \text{ IS } out1mf1) \text{ AND } ('disease' \text{ IS } out2mf1)$
Rule 2:	$IF (F2 \text{ IS } in2mf2) \text{ AND } (F3 \text{ IS } in3mf2) \text{ AND } (F9 \text{ IS } in9mf2)$ $AND (F10 \text{ IS } in10mf2) AND (F12 \text{ IS } in12mf2) \text{ AND } (F13 \text{ IS } in13mf2),$ $THEN ('healthy' \text{ IS } out1mf2) \text{ AND } ('disease' \text{ IS } out2mf2)$
Where:	$in2mf1 = \text{ridgemf}(x2; 0.5501, 0.5420, 1.8177)$ $in2mf2 = \text{ridgemf}(x2; 0.4347, 0.8214, 2.3004)$ $in3mf1 = \text{ridgemf}(x3; 0.3534, 0.6133, 2.8296)$ $in3mf2 = \text{ridgemf}(x3; 0.3235, 0.8691, 3.0914)$ $in9mf1 = \text{ridgemf}(x9; 0.4183, 0.1603, 2.3906)$ $in9mf2 = \text{ridgemf}(x9; 0.5494, 0.5536, 1.8203)$ $in10mf1 = \text{ridgemf}(x10; 0.1716, 0.0871, 5.8272)$ $in10mf2 = \text{ridgemf}(x10; 0.2595, 0.2447, 3.8531)$ $in12mf1 = \text{ridgemf}(x12; 0.2781, 0.1094, 3.5956)$ $in12mf2 = \text{ridgemf}(x12; 0.3904, 0.3809, 2.5614)$ $in13mf1 = \text{ridgemf}(x13; 0.2777, 0.5358, 3.6013)$ $in13mf2 = \text{ridgemf}(x13; 0.3148, 0.8241, 3.1764)$ $out1mf1 = 1.4717$, $out1mf2 = -0.3595$ $out2mf1 = -0.4717$, $out2mf2 = 1.3595$

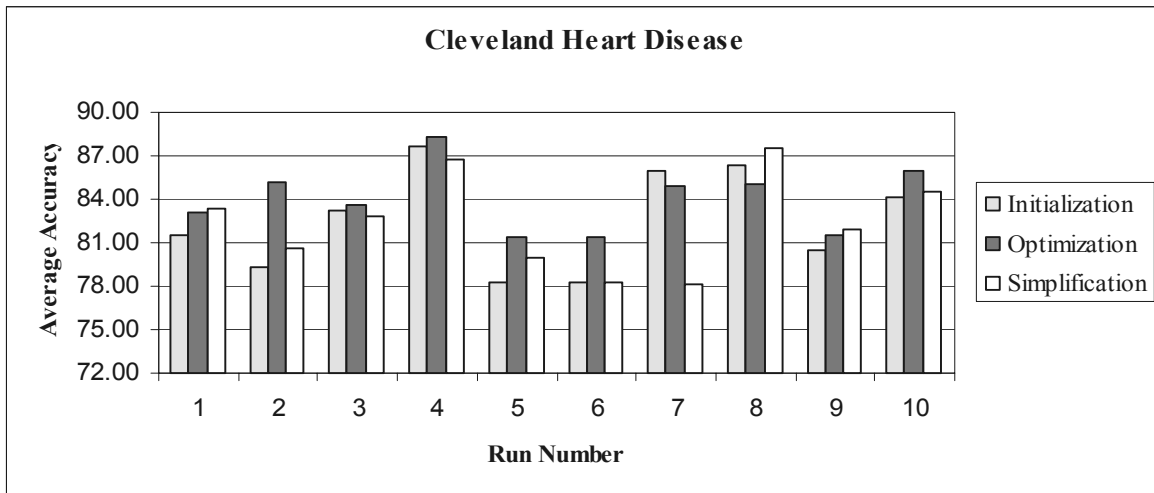
Figure 4.17. Case Study 3: Textual representation of the FRB obtained after simplification

4.4.5 Analysis of Results

The ten-fold cross validation results are summarized in Table 4.28 and Figure 4.21. To evaluate the effectiveness of such results, they were compared with other statistical, neural and rule-based classifiers developed for the same dataset, as shown in Table 4.29, Table 4.30 and Table 4.31.

Table 4.28. *Case Study 3: Summary of Classification results of FRULEX*

	Heart	Train	Test	Average
Phase 1	Misclassified	40	5.6	22.8
	Accuracy	83.5 %	81.51 %	82.51 %
Phase 2	Misclassified	36.6	5.1	20.85
	Accuracy	84.9 %	83.15 %	84.03 %
Phase 3	Misclassified	41.4	5.5	23.45
	Accuracy	82.92 %	81.84 %	82.38 %

**Figure 4.18.** *Case Study 3: Summary of Classification results of FRULEX***Table 4.29.** *Case Study 3: Statistical and Neural Classifiers*

Method	Classification Accuracy	Reference
LOONN	76.2%	[Andrews and Geva, 1994]
XVNN	76.2%	[Andrews and Geva, 1994]
RBP	81.3%	[Ster <i>et al.</i> , 1996]

- Leave-One-Out Nearest Neighbor, Cross Validation Nearest Neighbor methods and RBF network trained using BP learning have achieved accuracy **76.2%**, **76.2%** and **81.3%**, respectively. They are considered black-boxes as they do not provide any explanation to their decisions and have not any human-readable representation to their hidden knowledge.

Table 4.30. *Case Study 3: Crisp Rule-Based Classifiers*

Method	Classification Accuracy	Extracted Rules	Conditions Per Rule	Reference
SSV	81.8%	3 crisp rules	13	[Duch <i>et al.</i> , 2001]
RULEX	80.2%	3 crisp rules	5	[Andrews <i>et al.</i> , 1995]

- RULEX has achieved a high accuracy (**80.2%**) and has extracted three crisp rules with five conditions per rule, using RBP network but it does not allow the network to produce overlapping local response units. Avoid overlapping leads to suboptimal solutions.
- The crisp rule-based classifiers provide a black-and-white picture where the user needs additional information since only one class label is identified as the correct one. For medical diagnosis, physicians may wish to quantify “how severe the disease is”.

Table 4.31. *Case Study 3: Fuzzy Rule-Based Classifiers*

Method	Classification Accuracy	Extracted Rules	Conditions Per rule	Reference
FSM	82.0%	27 fuzzy rules	13	[Duch <i>et al.</i> , 2001]
<i>FRULEX</i>	81.84%	2 fuzzy rules	6	Proposed Approach

- Fuzzy rule-based classifiers provide a good platform to deal with uncertain, noisy, imprecise or incomplete information. They provide a gray picture where the user can gain further information. For medical diagnosis, physicians can quantify “how severe the disease is”. For pattern classification, user can quantify “how typical this pattern is”.
- FSM method with Gaussian functions generates 27 fuzzy rules and achieves in the ten-fold cross validation 93.4% accuracy on the training part and only **82.0%** on the test part. It should be noted that in our approach high accuracy (81.84%) on the test set (generalization ability) is achieved with an average of 6 input variables and 2 fuzzy rules with respect to the 13 features and 27 fuzzy rules used by FSM, thus resulting in a more simple and interpretable FKB. FSM pursue accuracy as ultimate goal and take no care about the interpretability of the extracted knowledge.

4.5 Case Study 4: Pima Indians Diabetes Dataset

4.5.1 Description of Case Study

The “Pima Indians diabetes” dataset is stored in the UCI repository [Mertz and Murphy, 1992] and is frequently used as benchmark case study. All patients were females at least 21 years old, of Pima Indian heritage. The data contains two classes, eight attributes, 768 instances, 500 (65.1%) healthy and 268 (34.9%) diabetes cases (See Table 4.32 and Table 4.33).

Table 4.32. Case Study 4: Classes

ID	Class
1	Healthy
2	Diabetes

Table 4.33. Case Study 4: Features and Feature values

ID	Feature	Feature values
F1	Number of times pregnant	Discrete
F2	Plasma glucose concentration	Continuous
F3	Diastolic blood pressure (mm Hg)	Continuous
F4	Triceps skin fold thickness (mm)	Continuous
F5	2-Hour serum insulin (mu U/ml)	Continuous
F6	Body mass index (weight in kg/(height in m) ²)	Continuous
F7	Diabetes pedigree function	Continuous
F8	Age	Discrete

To estimate the performance of the FKB extracted by the proposed approach, we carried out a 10-fold cross-validation. The whole dataset was divided into 10 equally sized groups (a group consists of 76 samples randomly drawn from the two classes). One part was used as a test set to test the fuzzy classifier, another part used as a cross validation test set to test the final feature subset, while classifier was trained with the remaining 8 parts.

4.5.2 Initialization Phase

The SCRG method, described in Chapter 3, is used to determine the initial centers and widths of the membership functions of the input features. Table 4.34 summaries the results after applying the SCRG phase for the ten trials. ($B=13$, $K_o=1.0$, $K=1.0$, $\sigma_o=0.05$, $\rho = 0.0001$, and $\tau=0.01$)

Table 4.34. *Case Study 4: Results of the 10-fold cross validation after initialization*

Diabetes			After Initialization Phase					
			Training Set		Test Set		Average	
Run	Rules	Features	Acc.	Misclass.	Acc.	Misclass.	Acc.	Misclass.
1	2	8	71.06	178	64.94	27	68.00	102.5
2	2	8	72.20	171	72.73	21	72.47	96
3	2	8	69.71	186	70.13	23	69.92	104.5
4	2	8	71.01	178	62.34	29	66.68	103.5
5	2	8	72.64	168	77.92	17	75.28	92.5
6	2	8	72.64	168	63.64	28	68.14	98
7	2	8	71.01	178	76.62	18	73.82	98
8	2	8	69.71	186	77.92	17	73.82	101.5
9	2	8	72.20	171	68.42	24	70.31	97.5
10	2	8	71.06	178	72.37	21	71.72	99.5
avg.	2.00	8.00	71.32	176.20	70.70	22.50	71.01	99.4

4.5.3 Optimization Phase

The backpropagation gradient descent learning method, described in Chapter 3, is used to optimize the fuzzy rule base extracted in phase one. Network with 8 inputs and 2 outputs, corresponding to the two classes, was constructed. Table 4.35 summaries the results obtained after 100 epochs for the ten trials. ($\epsilon=0.01$, and $\eta=1.0$)

Table 4.35. *Case Study 4: Results of the 10-fold cross validation after optimization*

Diabetes			After Optimization Phase					
			Training Set		Test Set		Average	
Run	Rules	Features	Acc.	Misclass.	Acc.	Misclass.	Acc.	Misclass.
1	2	8	76.75	143	72.73	21	74.74	82
2	2	8	74.96	154	75.32	19	75.14	86.5
3	2	8	75.57	150	67.53	25	71.55	87.5
4	2	8	75.90	148	66.23	26	71.07	87
5	2	8	74.43	157	80.52	15	77.48	86
6	2	8	74.43	157	67.53	25	70.98	91
7	2	8	75.90	148	80.52	15	78.21	81.5
8	2	8	75.57	150	79.22	16	77.40	83
9	2	8	74.96	154	78.95	16	76.96	85
10	2	8	76.75	143	78.95	16	77.85	79.5
avg.	2.00	8.00	75.52	150.40	74.75	19.40	75.14	84.90

For the third run of the 10 trials, Figure 4.19 shows the graphical representation of the FKB obtained, after optimization phase. (Using *MATLAB Fuzzy Toolbox*)

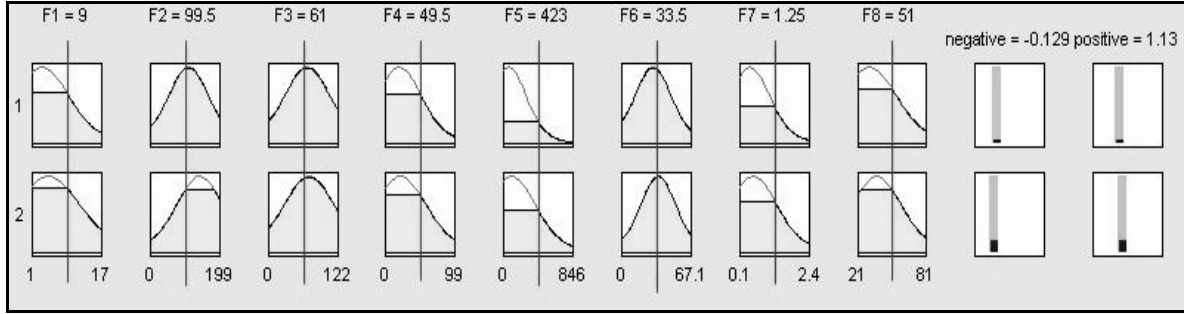


Figure 4.19. Case Study 4: Graphical representation of the FRB obtained after optimization

4.5.4 Simplification Phase

Feature Subset Selection by Relevance method, described in Chapter 3, is used to simplify the FKB extracted in phase one. Table 4.36 and Table 4.37 have summarized the results obtained after this phase, for the ten trials.

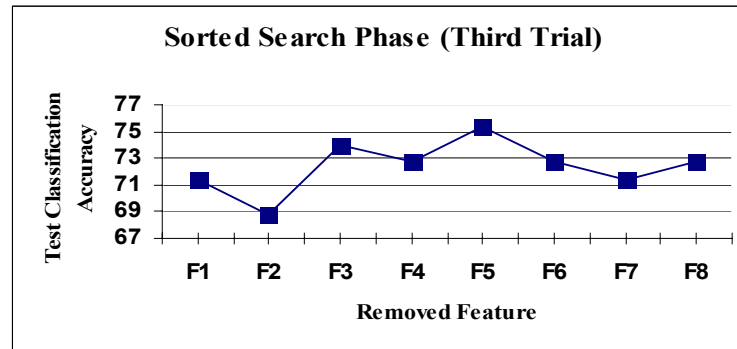
Table 4.36. Case Study 4: Results of 10-fold cross validation after sorted and neighbor search

Diabetes			After Sorted Search & Neighbor Search Phases						
			Best Feature Set	Training Set		Test Set		Whole Set	
Run	Rules	Feat.		Acc.	Mis.	Acc.	Mis.	Acc.	Mis.
1	2	5	{F2,F5,F6,F7,F8}	76.91	142	77.92	17	77.42	79.5
2	2	4	{F1,F2,F3,F7}	76.75	143	77.92	17	77.34	80
3	2	4	{F1,F2,F6,F7}	78.01	135	71.43	22	74.72	78.5
4	2	5	{F2,F3,F4,F5,F6}	75.57	150	76.62	18	76.10	84
5	2	5	{F1,F2,F6,F7,F8}	75.73	149	80.52	15	78.13	82
6	2	2	{F2,F6}	75.08	153	76.62	18	75.85	85.5
7	2	6	{F1,F2,F3,F5,F6,F7}	76.87	142	80.52	15	78.70	78.5
8	2	2	{F1,F2}	75.41	151	81.82	14	78.62	82.5
9	2	7	{F1,F2,F3,F4,F5,F6,F7}	77.56	138	78.95	16	78.26	77
10	2	6	{F1,F2,F4,F6,F7,F8}	76.26	146	82.89	13	79.58	79.5
avg.	2.00	4.60	{F1,F2,F6,F7}	76.42	144.90	78.52	16.50	77.47	80.70

Table 4.37. *Case Study 4: Results of the 10-fold cross validation after simplification*

Diabetes			After Simplification Phase						
Run	Rules	Feat.	Final Feature Set	Training Set		XV Test Set		Average	
				Acc.	Mis.	Acc.	Mis.	Acc.	Mis.
1	2	4	{F1,F2,F6,F7}	77.07	141	72.73	21	74.90	81
2	2	4	{F1,F2,F6,F7}	77.24	140	79.22	16	78.23	78
3	2	4	{F1,F2,F6,F7}	78.01	135	71.43	22	74.72	78.5
4	2	4	{F1,F2,F6,F7}	77.52	138	71.43	22	74.48	80
5	2	4	{F1,F2,F6,F7}	77.04	141	83.12	13	80.08	77
6	2	4	{F1,F2,F6,F7}	77.04	141	75.32	19	76.18	80
7	2	4	{F1,F2,F6,F7}	77.52	138	77.92	17	77.72	77.5
8	2	4	{F1,F2,F6,F7}	78.01	135	77.92	17	77.97	76
9	2	4	{F1,F2,F6,F7}	77.24	140	78.95	16	78.10	78
10	2	4	{F1,F2,F6,F7}	77.07	141	80.26	15	78.67	78
avg.	2.00	4	{F1,F2,F6,F7}	77.38	139.00	76.83	17.80	77.10	78.40

For the tenth run of the ten trails, Figure 4.20 shows the performance of the networks constructed by the successive removal of input features, Figure 4.21 shows the performance of the networks constructed by the successive addition of the relevant features, and Figure 4.22 and Figure 4.23 show the graphical and textual representation of the FKB obtained, respectively.

**Figure 4.20.** *Case Study 4: Performance of RBPN during removal of input features*

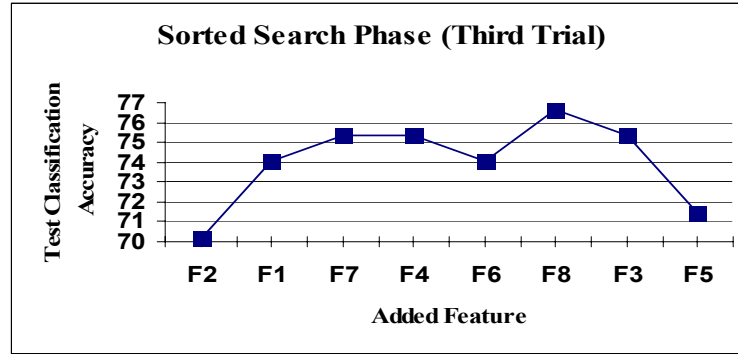


Figure 4.21. Case Study 4: Performance of the RBPN with different features

Rule 1:	<i>IF ('Times Pregnant' IS in1mf1) AND ('Plasma Glucose Conc' IS in2mf1)</i> <i>('Body Mass Index' IS in6mf1) AND ('Diabetes Pedigree' IS in7mf1),</i> <i>THEN 'negative' IS out1mf1 AND 'positive' IS out2mf1</i>	
Rule 2:	<i>IF ('Times Pregnant' IS in1mf2) AND ('Plasma Glucose Conc' IS in2mf2)</i> <i>('Body Mass Index' IS in6mf2) AND ('Diabetes Pedigree' IS in7mf2),</i> <i>THEN 'negative' IS out1mf1 AND 'positive' IS out2mf1</i>	
Where:	$\begin{aligned} in1mf1 &= ridgemf(x1; 3.8313 \ 3.3350 \ 0.2610) \\ in1mf2 &= ridgemf(x1; 4.5835 \ 4.7664 \ 0.2182) \\ in2mf1 &= ridgemf(x2; 36.0640 \ 109.2075 \ 0.0277) \\ in2mf2 &= ridgemf(x2; 41.9281 \ 140.6682 \ 0.0238) \\ in6mf1 &= ridgemf(x6; 11.0978 \ 30.1630 \ 0.0901) \\ in6mf2 &= ridgemf(x6; 10.6734 \ 35.1860 \ 0.0937) \\ in7mf1 &= ridgemf(x7; 0.4069 \ 0.4388 \ 2.4577) \\ in7mf2 &= ridgemf(x7; 0.4924 \ 0.5405 \ 2.0308) \\ out1mf1 &= 2.0346 & , out1mf2 &= -0.6664 \\ out2mf1 &= -1.0346 & , out2mf2 &= 1.6664 \end{aligned}$	

Figure 4.22. Case Study 4: Textual representation of the FRB obtained after simplification

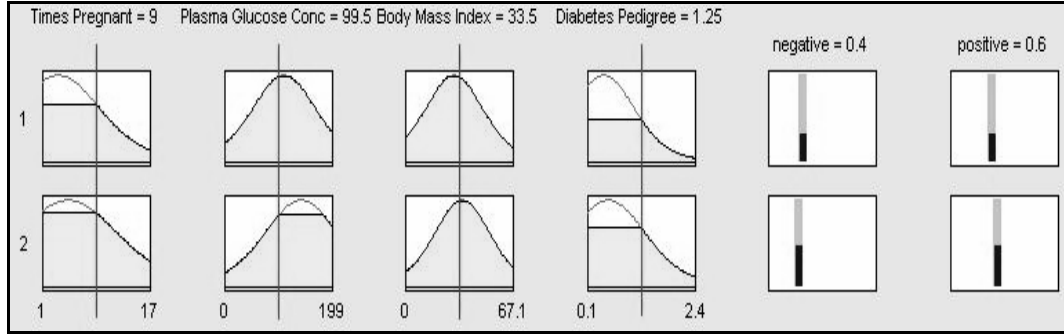


Figure 4.23. Case Study 4: Graphical representation of the FRB obtained after simplification

4.5.5 Analysis of Results

The ten-fold cross validation results are summarized in Table 4.38 and Figure 4.24. To evaluate the effectiveness of such results, they were compared with other statistical, neural and rule-based classifiers developed for the same dataset, as shown in Table 4.39, Table 4.40 and Table 4.41.

Table 4.38. Case Study 4: Summary of Classification results of FRULEX

Diabetes		Train	Test	Average
Phase 1	Misclassified	176.2	22.5	99.4
	Accuracy	71.32 %	70.7 %	71.01 %
Phase 2	Misclassified	150.4	19.4	84.9
	Accuracy	75.52 %	74.75 %	75.14 %
Phase 3	Misclassified	139.00	17.80	78.40
	Accuracy	77.38 %	76.83 %	77.10 %

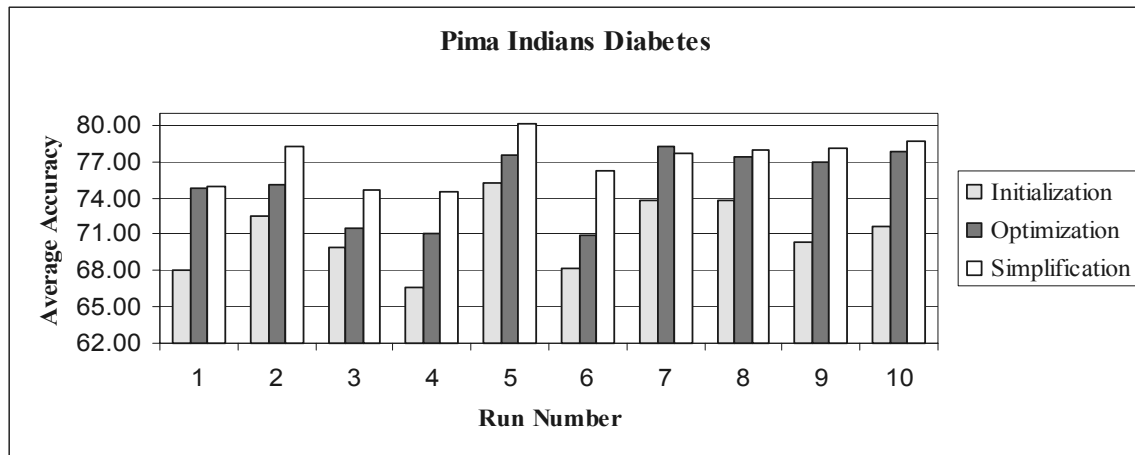


Figure 4.24. Case Study 4: Summary of Classification results of FRULEX

Table 4.39. *Case Study 4: Statistical and Neural Classifiers*

Method	Classification Accuracy	Reference
LOONN	70.4 %	[Andrews and Geva, 1994]
XVNN	70.7 %	[Andrews and Geva, 1994]
RBF +BP	75.7%	[Ster <i>et al.</i> , 1996]

- LOONN, XVNN methods and RBF network trained using BP learning have achieved accuracy **70.4%**, **70.7%** and **75.7%**, respectively. They are considered black-boxes as they do not provide any explanation to their decisions and have not any human-readable representation to their hidden knowledge.

Table 4.40. *Case Study 4: Crisp Rule-Based Classifiers*

Method	Classification Accuracy	Extracted Rules	Conditions Per rule	Reference
RULEX	72.6 %	5 crisp rules	5	[Andrews <i>et al.</i> , 1995]

- RULEX has achieved a high accuracy (**72.6%**) and has extracted five crisp rules with five conditions per rule, using RBP network but it does not allow the network to produce overlapping local response units. Avoid overlapping leads to suboptimal solutions.
- The crisp rule-based classifiers provide a black-and-white picture where the user needs additional information since only one class label is identified as the correct one. For medical diagnosis, physicians may wish to quantify “how severe the disease is”.
- The optimization of the crisp rule-based classifiers is difficult since only non-gradient based optimization methods may be used.

Table 4.41. *Case Study 4: Fuzzy Rule-Based Classifiers*

Method	Classification Accuracy	Extracted Rules	Conditions Per rule	Reference
FSM	73.8 %	50 fuzzy rules	8	[Duch <i>et al.</i> , 2001]
<i>FRULEX</i>	76.83%	2 fuzzy rules	4	Proposed Approach

- Fuzzy rule-based classifiers provide a good platform to deal with uncertain, noisy, imprecise or incomplete information. They provide a gray picture where the user can

gain further information. For medical diagnosis, physicians can quantify “how severe the disease is”. For pattern classification, user can quantify “how typical this pattern is”.

- FSM method with Gaussian functions generates 50 rules and achieves in the ten-fold cross validation 85.3% accuracy on the training part and only 73.8% on the test part. It should be noted that in our approach higher accuracy (76.83%) on the test set (generalization ability) is achieved with an average of 4 input variables and 2 fuzzy rules with respect to the 8 features and 50 fuzzy rules used by FSM, thus resulting in a more simple and interpretable FKB. FSM pursue accuracy as ultimate goal and take no care about the interpretability of the extracted knowledge.

4.6 Evaluation

This section presents the evaluation of the proposed approach according to the evaluation criteria mentioned previously in section 2.4.1.

4.6.1 Rule Format

FRULEX extracts fuzzy rules. In the directly extracted fuzzy system, each fuzzy rule contains an antecedent condition for each input dimension as well as a consequent, which describes the output class covered by that rule.

4.6.2 Complexity of the Approach

FRULEX, unlike other decomposition algorithms, does not rely on any form of search to extract rules. The initialization module is linear in the number of fuzzy clusters (or fuzzy rules) and the number of training patterns, $O(J.P)$. The optimization module is linear in the number of iterations, the number of fuzzy rules, and the number of training patterns, $O(I.J.P)$. The simplification module is linear in the number of features, $O(N)$. Therefore, *FRULEX* is computationally efficient.

4.6.3 Quality of the Extracted Rules

As stated previously, the essential function of rule extraction algorithms such as *FRULEX* is to provide an explanation facility for the trained network. The rule quality

criteria provide insight into the degree of trust that can be placed in the explanation. Rule quality is assessed according to the accuracy, fidelity and comprehensibility of the extracted rules.

4.6.3.1 *Comprehensibility*

In general, comprehensibility is inversely related to the number of rules and to the number of antecedents per rule. The RBPN is based on a greedy algorithm. Hence, its solutions are achieved with relatively small numbers of training iterations and are typically compact, i.e. the trained network contains a small number of local response units. Given that *FRULEX* converts each local response unit into a single fuzzy rule, therefore the extracted rule set contains, at most, the same number of rules as the number of local response units in the trained network.

4.6.3.2 *Accuracy*

During training phase, local response units will grow, shrink, and/or move to form a more accurate representation of the knowledge encoded in the training data.

4.6.3.3 *Fidelity*

Fidelity is closely related to accuracy and the factors that affect accuracy also affect the fidelity of the rule sets. In general, the rule sets extracted by *FRULEX* display an extremely high degree of fidelity with the networks from which they were drawn.

4.6.4 *Portability of the Approach*

FRULEX is non-portable having been specifically designed to work with RBPN, which is a local function network. This means that it cannot be used as a general-purpose device for providing an explanation component for existing, trained neural networks. *FRULEX* is also applicable to a broad variety of problem domains in the fields of pattern classification and medical diagnosis. (Including domains with continuous, discrete, or missing values)

4.6.5 Translucency of the Approach

FRULEX is a decompositional approach, as fuzzy rules are extracted at the level of the hidden layer units. Each local response unit is treated in isolation with the output weights being converted directly into a fuzzy rule.

4.6.6 Consistency of the Approach

FRULEX is a consistent algorithm because it always generates different fuzzy systems with the same accuracy from any given training run.

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

Rule extraction methods should not be judged only on the basis of the accuracy of the rules but also on their simplicity and their comprehensibility. Comprehensibility of knowledge extracted from data is a very attractive feature for a neuro-fuzzy approach, since it establishes a bridge between the symbolic reasoning paradigm, that provides explicit knowledge representation, and the sub-symbolic paradigm, where systems like neural networks discover automatically knowledge from data. For complex and high-dimensional classification tasks, data-driven extraction of classifiers has to deal with a number of structural problems such as the *effective initial partitioning of the input domain* and the *selection of the relevant features*. Also, *linguistic interpretability* is an important aspect of these classifiers. Fuzzy logic helps improving the interpretability of knowledge-based classifiers through its semantics that provide insight in the classifier internal structure. A fuzzy classifier that is accurate and interpretable as well can hardly be found by a completely automatic learning process. Most of the modeling approaches pursue only accuracy as ultimate goal and take no care about the interpretability of the knowledge representation. The proposed approach aims to make a step further to solve these problems.

This thesis presents a neuro-fuzzy approach for the data-based extraction of fuzzy rule-based classifiers that is easily interpretable by human. In the first phase, an initial model is derived using a fuzzy clustering method (SCRG). A given training data set is partitioned into a set of clusters based on input-similarity and output-similarity tests. Membership functions associated with each cluster are defined according to statistical means and

variances of the data points included in the cluster. A fuzzy IF-THEN rule is extracted from each cluster to form a fuzzy rule-base from which a fuzzy neural network is constructed. In the second phase, parameters of the membership functions are refined to increase the precision of the fuzzy rule-base using an efficient gradient-descent learning method (BP). In the third phase, the extracted fuzzy rule-base is simplified using feature subset selection method, (FSS), to increase the readability and simplicity.

For structure identification step, an efficient partitioning method is used. The number of fuzzy rules extracted is determined automatically without user intervention and the membership functions match closely with the real distribution of the training data points.

For parameter identification step, the constructed knowledge-based neural network converges very rapidly because the initial weights of the network are set by the parameters of original fuzzy rules, which is built from the data in the first step.

In real world applications usually there are many features some of which may not be relevant to the problem domain. They may even be adding noise to the problem. Usually a subset of the features will speed learning process and will improve accuracy. Some of the features may also be expensive to acquire (like in medical applications). FSS is a search and optimization problem. The search space is very big even for small set of features. The number of possible states is 2^N (N: number of features). So an exhaustive search is not possible if N is not very small. Researchers developed other heuristic methods which are not computationally expensive as exhaustive search. But still they require many tests of the states in the search space. FSS method finds a starting point by sorting the features, in the beginning, by their relevancy algorithm and therefore visits fewer states than other methods. In most of the tests done, accuracy was improved when compared to the original feature set. The method used for choosing the final feature subset improves accuracy and it chooses more reliable subsets since it is using k-fold cross validation for choosing the subset. This shows that starting the search from a state chosen by using feature relevancy decreases the number of states to be tested. Also, FSS is performed automatically without user intervention.

The case studies have also been showed that it is possible to get a proper rule structure by the proposed rule initialization-optimization-simplification procedure and the obtained

fuzzy classifier accuracy is comparable to the best results reported in the literature. On the overall, the reported results indicate that *FRULEX* approach is a valid tool to automatically extract fuzzy rules from data providing a good balance between accuracy and readability.

5.2 *Future Work*

This section presents a few topics for future research in the area related to the thesis:

- **Function approximation:** We are planning to apply our approach to function approximation problems.
- **Mamdani-type fuzzy models:** We can extend our proposed approach to be applied to other types of fuzzy models, such as Mamdani-type fuzzy models.
- **Real-world problems:** We expect that the proposed approach should be considered further in respect to a wider range of real-world problems.
- **Genetic Algorithms:** The use of Genetic Algorithms (GA) instead of backpropagation learning algorithm. GA does not suffer from convergence problems with the same degree that the BP suffers.
- **Information Extraction:** We are planning to integrate *FRULEX* approach with Information Extraction (IE) techniques to deal with free text and semi-structured data. (Currently, *FRULEX* approach deals with structured data)

BIBLIOGRAPHY

- [Abdel Hady et. al., 2003] Abdel Hady, M.F. and Wahdan, M.A. (2003). Frulex – A New Approach for Fuzzy Rules Extraction Using Rapid Back Propagation Neural Networks. *Proceedings of the 38th International Conference on Statistics, Computer Sciences and Operation Research*, pp. 59-80, Cairo, Egypt.
- [Abdel Hady et. al., 2004] Abdel Hady, M.F., Wahdan, M.A. and Elmaghraby, A.S. (2004). FRULEX - Fuzzy Rules Extraction Using Rapid Back Propagation Neural Networks. *Proceedings of the 2nd International Conference on Informatics and Systems, INFOS'2004*, Cairo, Egypt.
- [Abe and Lan, 1995] Abe, S. and Lan, M.S. (1995). A Method for Fuzzy Rules Extraction Directly from Numerical Data and Its Application to Pattern Classification. *IEEE Trans. on Fuzzy Systems*, vol. 3, no.1, pp. 18-28.
- [Andrews and Geva, 1994] Andrews, R. and Geva, S. (1994). Extracting Rules from a Constrained Error Backpropagation Network. *Proceedings of the 5th Australian Conference on Neural Networks*, Brisbane, pp. 9-12.
- [Andrews and Geva, 1995] Andrews, R. and Geva, S. (1995). RULEX and CEBP Networks as the Basis for a Rule Refinement System. In *Hybrid Problems Hybrid Solutions*, pp. 1-12.
- [Andrews et al., 1995] Andrews, R., Diederich, J. and Tickle, A.B. (1995). A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks. *Knowledge-Based Systems*, vol. 8, pp. 378-389.
- [Andrews and Geva, 1999] Andrews, R. and Geva, S. (1999). On the Effects of Initializing a Neural Network with Prior Knowledge. *Proceedings of the International Conference on Neural Information Processing*, pp. 251-256, Perth, Western Australia.

-
- [Benitez *et al.*, 1997] Benitez, J. M., Castro, J. L. and Requena, I. (1997). Are Artificial Neural Networks Black Boxes?. *IEEE Trans. on Neural Networks*, vol. 8, no. 5, pp. 1156–1164.
- [Berthold and Huber, 1995] Berthold, M. and Huber, K. (1995). Building Precise Classifiers with Automatic Rule Extraction. In *Proceeding of the IEEE International Conference on Neural Networks*, Perth, Australia. vol. 3, pp. 1263-1268.
- [Boz, 2000] Boz, O. (2000). Converting a Trained neural Network to a Decision Tree. Ph.D. Thesis, Lehigh University, Bethlehem, Pennsylvania.
- [Boz, 2002] Boz, O. (2002). Feature Subset Selection by Using Sorted Feature Relevance Proc. of The 2002 Intl. Conf. on Machine Learning and Applications.
- [Bottou and Vapnik, 1992] Bottou, L. and Vapnik, V. (1992). Local Learning Algorithms. *Neural Computation*, vol. 4, pp. 888-900.
- [Castellano *et al.*, 2000a] Castellano, G. and Fanelli, A. M. (2000). Fuzzy Classifiers Acquired from Data. In Mohammadian, M. (Ed.), *New frontiers in computational intelligence and its applications*. IOS Press, pp. 31-41.
- [Castellano *et al.*, 2000b] Castellano, G. and Fanelli, A. M. (2000). Variable Selection Using Neural Network Models. *Neurocomputing*, vol. 31, no. 14, pp. 1-13.
- [Castellano *et al.*, 2002] Castellano, G., Fanelli, A. M. and Mencar, C. (2002). A Neuro-Fuzzy Network to Generate Human-Understandable Knowledge from Data. *Cognitive Systems Research*, vol. 3, pp.125-144.
- [Castro *et al.*, 2002] Castro, J. L., Mantas, C. J. and Benitez, J. M. (2002). Interpretation of Artificial Neural Networks by Means of Fuzzy Rules. *IEEE Trans. on Neural Networks*, vol. 13, no. 1, pp. 101–116.
- [Doak, 1992] Doak, J. (1992). Intrusion Detection: The Application of Feature Selection, a Comparison of Algorithms, and the Application of a Wide Area Network Analyzer. Master's thesis, University of California, Davis, Department of Computer Science.
- [Dubois and Prade, 1980] Dubois, D. and Prade, H. (1980). Fuzzy Sets and Systems: Theory and Applications. *Academic Press*, New York.

- [Duch *et al.*, 1999] Duch, W., Adamczak, R. and Grabczewski, K. (1999). Neural Optimization of Linguistic Variables and Membership Functions. *Proceedings of the 6th Internal Conference on Neural Information Processing ICONIP'99*, Perth, Australia, vol. 2, pp. 616-621.
- [Duch *et al.*, 2001] Duch, W., Adamczak, R. and Grabczewski, K. (2001). A New Methodology of Extraction, Optimization and Application of Crisp and Fuzzy Logical Rules. *IEEE Trans. on Neural Networks*, vol. 12, no. 2, pp. 277–306.
- [Farag *et al.*, 1998] Farag, W. A., Quintana, V.H. and Lambert-Torres, G. (1998). A genetic-based neuro-fuzzy approach for modeling and control of dynamical systems. *IEEE Trans. on Neural Networks*, vol.9, pp. 756-767.
- [Geva and Sitte, 1994] Geva, S. and Sittle, J. (1994). Constrained Gradient Descent. In *Proceedings of the 5th Australian Conference on Neural Computing*, Brisbane, Australia.
- [Jang, 1993] Jang J.-S. R. (1993). ANFIS: Adaptive-Network-based Fuzzy Inference System. *IEEE Trans. on Systems, Man and Cybernetics*, vol. 23, no. 3, pp. 665-683.
- [Jang and Sun, 1993] Jang, J.-S. R. and Sun, C.-T. (1993). Functional Equivalence Between Radial Basis Function Networks and Fuzzy Inference Systems. *IEEE Trans. on Neural Networks*, vol. 4, pp. 156–159.
- [Jang *et al.*, 1998] Jang, J.-S. R., Sun, C.-T. and Mizutani E. (1998). Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence. Prentice Hall, Upper Saddle River, NJ, 2nd Edition.
- [Kantardzic and Elmaghraby, 1997] Kantardzic, M.M. and Elmaghraby, A.S. (1997). Logic-Oriented Model of Artificial Neural Networks. *Info. Sciences Journal*, vol. 101, no. (1-2): pp. 85-107.
- [Kubat, 1998] Kubat, M. (1998). Decision Trees Can Initialize Radial-Basis Function Networks. *IEEE Trans. on Neural Networks*, vol. 11, no. 3, pp. 813-820.
- [Lapedes and Faber, 1987] Lapedes, A. and Faber, R. (1987). How Neural Networks Work. *Neural Information Processing Systems*, Anderson D.Z.(ed), American Institute of Physics, New York, pp. 442-456.

-
- [Lee *et al.*, 2003] Lee, S. J. and Ouyang, C. S. (2003). A Neuro-Fuzzy System Modeling with Self-Constructing Rule Generation and Hybrid SVD Based Learning. *IEEE Trans. on Fuzzy Systems*, vol.11, pp. 341-353.
- [Lin *et al.*, 1997] Lin, Y., Cunningham, G. A. and Coggeshall, S. V. (1997). Using Fuzzy Partitions to Create Fuzzy Systems from Input-output Data and Set the Initial Weights in a Fuzzy Neural Network,” *IEEE Trans. On Fuzzy Systems*, vol. 5, pp 614-621.
- [Mcculloch and Pitts, 1943] Mcculloch, W. S. and Pitts, W. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-133.
- [Miller, 1990] Miller, A. J. (1990). *Subset Selection in Regression*. Chapman and Hall.
- [Mitra and Hayashi, 2000] Mitra, S. and Hayashi, Y. (2000). Neuro-fuzzy Rule Generation: Survey in Soft Computing Framework. *IEEE Trans. on Neural Networks*, vol. 11, no. 3, pp. 748-768.
- [Molina *et al.*, 2002] Molina, L.C., Belanche, L. and Nebot, A. (2002). Feature Selection Algorithms: A Survey and Experimental Evaluation. In Proc. of the Intl. Conf. on Data Mining, Maebashi City, Japan.
- [Moody and Darken, 1989] Moody, J. and Darken, C. J. (1989). Fast Learning in Networks of Locally Tuned Processing Units. *Neural Computation*, pp. 281-294.
- [Mertz and Murphy, 1992] Mertz, C. J. and Murphy, P. M. (1992). UCI Repository of Machine Learning Databases. University of California, Department of Information and Computer Science, Irvine, CA. Available Online: <ftp://ftp.ics.uci.edu/pub/machine-learning-data-bases>
- [Narendra and Fukunaga, 1977] Narendra, P. and Fukunaga, K. (1977). A branch and bound algorithm for feature subset selection. *IEEE Trans. on Computing*, vol.26, pp. 917-922
- [Nauck *et al.*, 1996] Nauck, D., Nauck, U. and Kruse, R. (1996). Generating Classification Rules with the Neuro-Fuzzy System NEFCLASS. In *Proceedings Biennial Conference North America Fuzzy Information Processing Society. (NAFIPS'96)*, Berkeley, CA.

-
- [Nauck *et al.*, 1999]] Nauck, D. and Nauck, U. (1999). Obtaining interpretable fuzzy classification rules from medical data. *Artificial Intelligence in Medicine*, vol. 16, pp. 149-169.
- [Pal *et al.*, 1996] Pal, S.K., and Ghosh, A. 1996. Neuro-fuzzy Computing for Image Processing and Pattern Recognition. *International Journal for Systems and Science*, vol. 27, pp. 1179-1193.
- [Parker, 1987] Parker, D. (1987). Optimal Algorithms for Adaptive Networks: Second Order Back Propagation, Second Order Direct Propagation and Second Order Hebbian Learning. In *Proceedings of the IEEE First International Conference on Neural Networks*, vol. 2, San Diego, CA, pp. 593-600.
- [Rojas *et al.*, 2000] Rojas, I., Pomares, H., Ortega, J. and Prieto, A. (2000). Self-organized Fuzzy System Generation from Training Examples. *IEEE Trans. On Fuzzy Systems*, vol. 8, pp. 23-36.
- [Rumelhart *et al.*, 1986] Rumelhart, D. E., Hinton, G. R. and Williams, R. J. (1986). Learning Internal Representations by Error Propagation. In *Parallel Distributed Processing*, vol.1, D., MIT Press, Cambridge, MA.
- [Ster and Dobnikar, 1996] Ster, B. and Dobnikar, A. 1996. Neural networks in Medical Diagnosis: Comparison with other methods. In *Proceedings of the International Conference EANN'96*, pp. 427-430.
- [Taha and Ghosh, 1996a] Taha, I. and Ghosh, J. (1996a). Three Techniques for Extracting Rules from Feedforward Networks. In *Intelligent Engineering Systems Through Artificial Neural Networks*, vol. 6, pp. 23-28.
- [Taha and Ghosh, 1996b] Taha, I. and Ghosh, J. (1996b). Symbolic Interpretation of Artificial Neural Networks. Technical Report, Computer and Vision Research Center, University of Texas, Austin.
- [Takagi and Sugeno, 1983] Takagi, T. and Sugeno, M. (1983). Derivation of Fuzzy Control Rules from Human Operator's Control Actions. *Proceedings of the IFAC Symposium on Fuzzy Information, Knowledge Representation and Decision Analysis*, pp. 55-60.

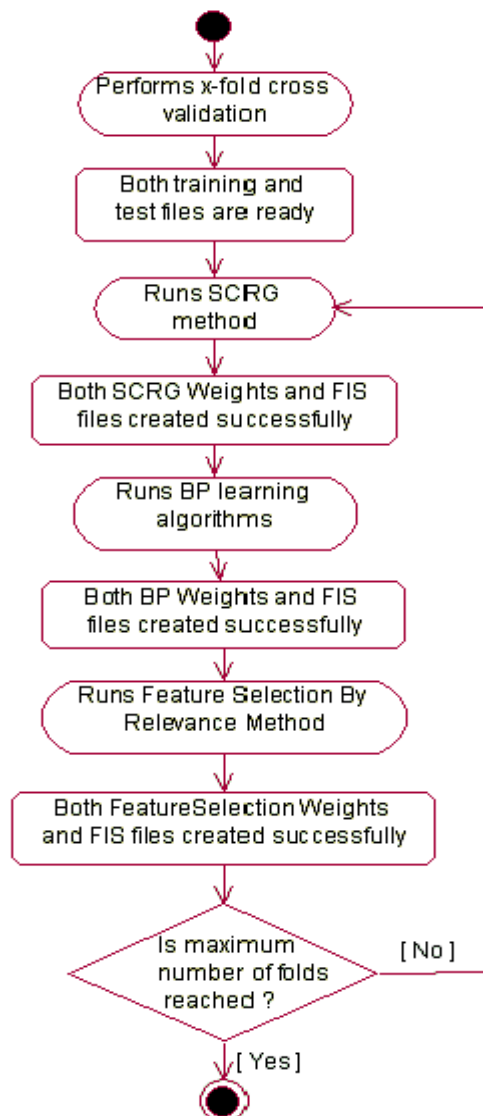
-
- [Takagi and Sugeno, 1985] Takagi, T. and Sugeno, M. (1985). Fuzzy Identification of Systems and Its Application to Modeling and Control. *IEEE Trans. on Systems, Man, and Cybernetics*, pp. 116-132.
- [Towell and Shavlik, 1993] Towell, G. and Shavlik, J. (1993). The Extraction of Refined Rules from Knowledge-based Neural Networks. *Machine Learning*, vol. 131, pp. 71-101.
- [Tresp *et al.*, 1993] Tresp, V., Hollatz, J. and Ahmed, S. (1993). Network Structuring and Training Using Rule-based Knowledge. *Advances in Neural Information Processing Systems (NIPS*6)*, pp. 871-878.
- [Werbos, 1974] Werbos, P. (1974). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Ph.D. Thesis, Harvard University, Boston, MA.
- [Wu *et al.*, 2000] Wu, S. and Er, M. J. (2000). Dynamic Fuzzy Neural Networks- a Novel approach to Function Approximation. *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 30, pp. 358-364.
- [Wu *et al.*, 2001] Wu, S., Er, M. J. and Gao, Y. (2001). A Fast Approach for Automatic Generation of Fuzzy Rules by Generalized Dynamic Fuzzy Neural Networks. *IEEE Trans. on Fuzzy Systems*, vol. 9, pp. 578-594.
- [Zadeh, 1965] Zadeh, L. A. (1965). Fuzzy Sets. *Information and Control*, vol. 8, pp. 338-353.
- [Zadeh, 1994] Zadeh, L. A. (1994). Fuzzy Logic, Neural Networks, and Soft Computing. *Communications of ACM*, vol. 37, pp. 77-84.

LIST OF ABBREVIATIONS

ANFIS	Adaptive Neural Fuzzy Inference System
ANN	Artificial Neural Network
BP	Back Propagation
FKB	Fuzzy Knowledge Base
FL	Fuzzy Logic
FNN	Fuzzy Neural Network
FRB	Fuzzy Rule Base
FSS	Feature Subset Selection
GA	Genetic Algorithm
LRU	Local Response Unit
LVQ	Learning Vector Quantization
MF	Membership Function
MSE	Mean Squared Error
NFS	Neuro-Fuzzy System
NN	Neural Network
PE	Processing Element
RBF	Radial Basis Function
RBPN	Rapid Back Propagation Network
RecBF	Rectangular Basis Function
SCRG	Self Constructing Rule Generator

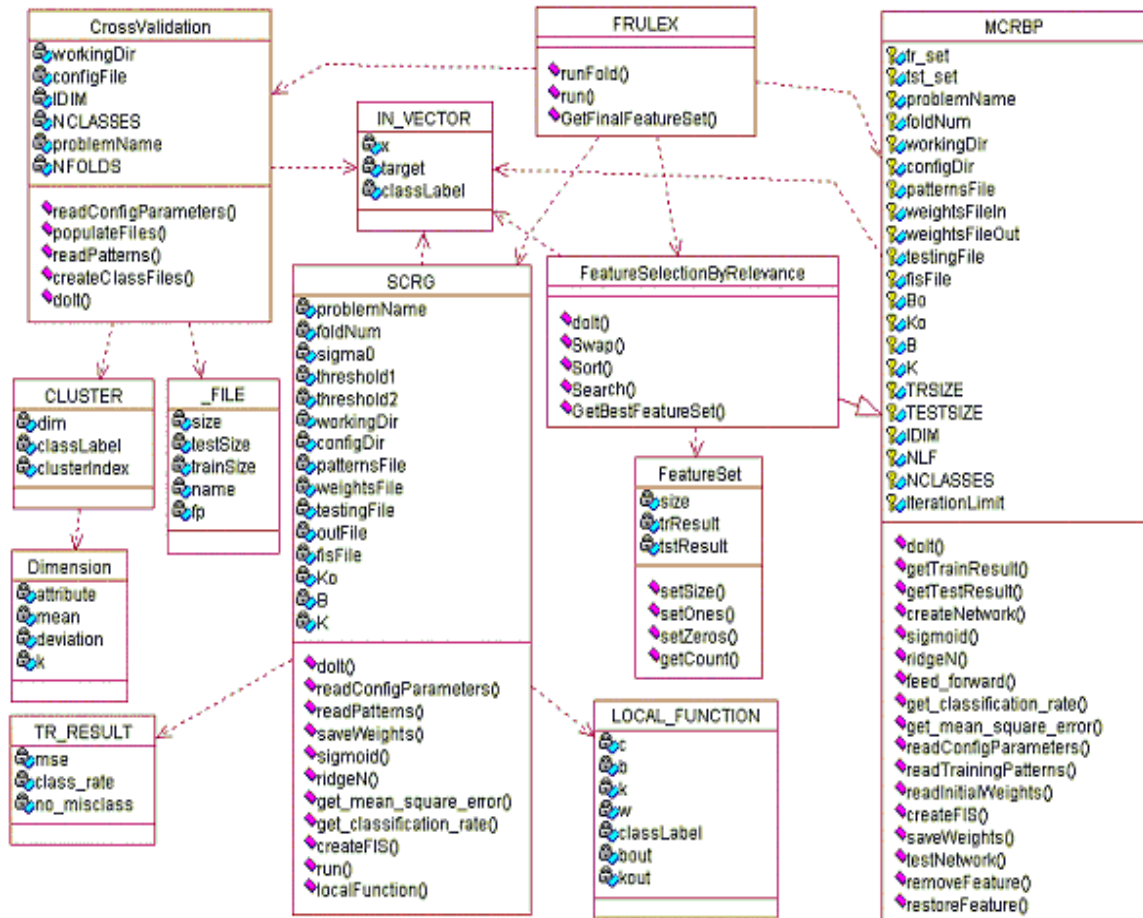
FRULEX FLOWCHART

The figure below shows the flow chart that illustrates the main functions performed by the *FRULEX* approach, drawn using *Rational™ Rose*.



FRULEX CLASS DIAGRAM

The figure shows the class diagram of the C++ implementation of the *FRULEX* approach, drawn using *Rational™ Rose*.



الملخص

ان الإستعمال المتزايد للشبكات العصبية خلال السنوات الماضية، قد جعل عملية استخراج القواعد منهم قضية هامة. في هذه الرسالة، نَقَدُّم طريقه جديده لإستخراج القواعد الضبابيه من بيانات العددية، والتي يتم استخدامها في مجال تصنيف النماذج والتشخيص الطبي. تدمج الطريقه المقترحه بين مميزات نظرية المنطق الضبابية، والشبكات العصبية. كما تستعمل الطريقه المقترحه نوع خاص من الشبكات العصبية، الذي يستطيع معالجة كلا من المعرفة الكميّة (العدديه) والنوعيّه (اللغويه). يمكن أن تعتبر الشبكة المستخدمه كنظام إستدلال ضبابي تكيفي بالقابلية لتعليم القواعد الضبابية من البيانات، وكشبكات عصبية مجهزة بالمعنى اللغوي. تستخرج القواعد الضبابية في ثلاثة مراحل: المرحلة الابتدائية، مرحلة التحسين، واخيرا مرحلة تبسيط النموذج الضبابي. في المرحلة الأولى، تقسم مجموعة البيانات آليا الي مجموعة من العناقيد مستندة الى إختبارات تشابه المدخلات و تشابه المخرجات. تربط داله عضويه بكلّ عنقود و تعرف الداله طبعا للوسط الحسابي والتباين الإحصائي للنقاط الواقعة في كلّ عنقود. ثمّ، تستخرج قاعده ضبابيه من كلّ عنقود مشكّلة في النهاية نموذج ضبابي. في المرحلة الثانية، يستخدم النموذج الضبابي المستخرج في المرحلة الأولى كنقطة البداية لبناء شبكه عصبيه ثمّ يتم تحسين معاملات النموذج الضبابي عن طريق تحليل عقد الشبكة التي درّبت عن خلال طريقة الانتشار الخلفي. عادة ما تحتوي التطبيقات الخاصه بالتصنيف على العديد من المدخلات و هذا بالطبع يزيد تعقيد مهمّة التصنيف. إختيار مجموعه جزئيه من المدخلات قد يزيد دقة ويخفّض تعقيد عملية اكتساب المعرفة. في المرحلة الثالثه، يتم استخدام طريقه تعتمد على ترتيب المدخلات من حيث الأهميّة وذلك لتقليص عدد الشروط في القواعد الضبابيه المستخرجه. يتم تقييم الطريقه المقترحه من خلال تطبيقها على عدد من مجموعات البيانات المشهورة و ذلك وفقا لمعايير التقييم المعروفه. كما يتم مقارنة النتائج بنتائج عدد من الطرق الأخرى المستخدمه في نفس المجال البحثي.

جامعة القاهرة
معهد الدراسات والبحوث الإحصائية
قسم علوم الحاسب و المعلومات

طريقة جديدة لإستخراج قواعد مبهمه
باستخدام
الشبكات العصبية الاصطناعية

إعداد

محمد فاروق عبد الهادى محمد
معيد بمعهد الدراسات والبحوث الإحصائية

تحت إشراف

أ. د. / عادل المغربى د. / مرفت غيث د. / محمود وهدان
معهد الدراسات والبحوث الإحصائية معهد الدراسات والبحوث الإحصائية وزارة الاتصالات و المعلومات

قدمت هذه الرسالة استكمالاً لمتطلبات درجة الماجستير فى علوم الحاسب, قسم علوم
الحاسب و المعلومات, معهد الدراسات والبحوث الإحصائية – جامعة القاهرة.