University of Alberta

COMPUTATIONAL INTELLIGENCE-BASED TECHNIQUES IN THE CONSTRUCTION AND REDUCTION OF RULE-BASED SYSTEMS

by

Kuwen Li

A thesis submitted to the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering

Fall 2013

Edmonton, Alberta

Abstract

This dissertation focuses on applying Computational Intelligence, a consortium of the technologies of fuzzy sets, neurocomputing and evolutionary computing, to the design and analysis of fuzzy rule-based systems (FRBS). We discuss two methods to construct FRBS, where the crux of the method is to seamlessly be based on the fuzzy neural network (FNN) and fuzzy c-means (FCM) clustering respectively. The rule set for the FRBS derived from the above two methods commonly consists of quite a number of rules and including all the attributes from the problem inputs. It becomes necessary and intuitive to reduce the dimensionality (number of input attributes in the rule) of the rules in the rule set. Also, some rules in the rule set might be conflicting with others. To make the FRBS more concise, the less important rules could be removed from the rule set. So after finishing the construction of FRBS, the rule complexity reduction algorithms are applied.

The key results of this study include:

- Construction of the FRBS with the aid of FNNs where the network is developed through genetic optimization.

- Reduction of complexity in terms of dimensionality and quantity (viz. the number of rules) by configuring pruning thresholds for AND neurons and OR neurons. The optimal values of the thresholds are determined in a way one strikes a sound balance between the interpretability of the rules and the accuracy associated with the reduced (simplified) rules. To develop the model optimal against these two competing objectives, multi-objective optimization is considered.

- Application of FRBS constructed with the use of FNN to well-known datasets and a real-world application such as deployment of wireless sensor networks.
- Construction of FRBS involving mechanism of information granulation (fuzzy clustering) and local linear models and studies on their complexity management through reduction of condition space and a relational expansion of fuzzy clusters.

Acknowledgements

I would like to express my deep gratitude to my supervisors, Dr. Witold Pedrycz and Dr. Marek Reformat, for their precious advice, encouragement, continuous support and guidance throughout my study.

Thanks to Dr. Petr Musilek, and Dr. Vicky Zhao for the invaluable comments and encouragement.

Thanks to my wife, Ying, for giving me all the support that I needed in my study and to my son, HongXuan, who has missed lots of fun during this period, and to my baby boy, Michael, for giving me plenty of pleasure and joy recently.

Table of Contents

List of Tables

List of Figures

Chapter 1 Motivation and Objectives

Nowadays, we are facing more and more serious threaten of information lost. Rapidly and correctly extracting useful information from huge amount of data has attracted a great deal of attention. Fuzzy rule-based systems can efficiently analyze data and come down to present the knowledge learnt from data in gracefully interpretable structures, so it has been widely adopted in data modeling, pattern recognition, and data mining, etc. Most of the real-world data sets comprise a large number of input attributes. Quite often some of those input attributes could be irrelevant and in this way negatively impact the design of a model. We are also faced with noisy data inputs. To simplify the structure and enhance the interpretability of the fuzzy rule-based system, a reduction of rule complexity becomes necessary. In our study, we proposed two methods to construct fuzzy rule-based systems and put forward related rule reduction algorithms.

1.1 Introduction

In our research, we take advantage of a number of fundamental technologies of Computational Intelligence. This environment embraces fuzzy logic (and fuzzy rule-based architectures as a highly visible algorithmic environment), evolutionary computing and neural networks. Here we offer a concise introduction to the area and highlight the main facets of these technologies.

- *Evolutionary computation*

Evolutionary computation is the subfield in computational intelligence. Evolutionary computation is a population-based optimization technique. The population has an initial state, and evolves during the iterations. Each individual in the population is the probable solution to the problem. The information pieces changed in the individual by carrying out recombination or mutations are generally done in stochastic way. Evolutionary computation is widely used to solve multi-dimensional problems, where it is sometimes more efficient and economical than conventional search algorithms.

- *Fuzzy Set and Fuzzy Logic*

Fuzzy set and fuzzy logic were introduced by Zadeh in 1965. Each item in a fuzzy set will not be either belonging to or excluded from a set. A real number positioned in-between 0 and 1 is assigned to every item to quantify the degree it is related to the set. After the emergence of fuzzy notions, the applications in this area have been extensively employed in numerous fields, ranging from pattern recognition, data modeling, data mining, data classification, data clustering, control engineering to fuzzy expert system. In control engineering, fuzzy sets give rise to a notion of fuzzy controllers. The essence of Fuzzy logic control is a non-conventional control type based on fuzzy rules that are combined with fuzzy logic. We may view fuzzy controllers as examples of real-time expert system being applied to nonlinear control problems.

- *Fuzzy neural networks*

Neural networks can work well if the problem is presented by a relatively sufficient amount of observed samples. These observations are used to train the black box. As to advantages, no prior knowledge of the problem needs to be given. Taking disadvantages into consideration, however, it is not straightforward to extract comprehensible rules from the neural network's structure.

On the contrary, a fuzzy system tries to present the knowledge learnt from the samples in linguistic rules. Furthermore, the input and output variables have to be described linguistically. If the knowledge is incomplete, wrong or contradictory, the fuzzy system can employ the tuning approach to interpret the data. Since there is no formal approach to it, the tuning is performed in a heuristic way. This activity could be usually very time-consuming and error-prone.

Fuzzy neural network comes as the result of the combination of fuzzy logic and neural networks. They capitalize on well-articulated capabilities of neural networks and the inherent transparency of knowledge representation of fuzzy rule-based systems.

- *Fuzzy clustering*

Clustering is the process of dividing data into clusters so that items located in the same cluster are similar to a significant extent, and items positioned in different clusters are highly distinct. Depending on the nature of the data and the purpose for which clustering is being used, different measures of similarity may be used to place items into classes. Some examples of measures that can be utilized as in clustering include distance, connectivity, and density. Fuzzy clustering is an extension to Boolean clustering in the sense that we admit a notion of partial membership of a given element to the cluster. The corresponding membership grades describe strength of the association (belongingness) of the data to the cluster thereby offering a more detailed insight into the structure of the data. In particular, through fuzzy clustering we can identify elements of a borderline nature. A well known representative of the class of algorithms of fuzzy clustering is the Fuzzy C- Means (FCM) algorithm.

- *Fuzzy rule-based system*

The rule based system is created by utilizing a working memory and a set of rules. Rule-based systems are a relatively simple model that can be utilized to almost any problem. The rule-based system itself uses a simple technique: It starts with a rule-base, which contains all the appropriate knowledge encoded into If-Then rules, and a working memory. When applying the rule-based system, the system will first go through the conditions in IF clause for all the rules in order to determine a subset of the rules whose conditions are satisfied based on given input information in the working memory. For each rule inside the generated subset, the action, defined in THEN clause, will be executed or fired. These actions can be anything the rule-based system is designed to react, including modifying the working memory, making changes to the rule-base, terminating the induction process, etc. The rule-based system continues to work in the loop of firing rules and performing actions until there are no available rules whose conditions are satisfied or a predefined rule, whose action indicates the termination of the induction process, is triggered.

Fuzzy rule-based systems (FRBS) have the ability to provide an accurate solution even there are inaccurate data. In addition, FRBS have the ability to make use of knowledge from experts and extract knowledge from raw data. The knowledge can be expressed in the form of rules to provide a solution for the problem. It is much easier for the end-users, even non-technical users to understand the solution which is written in rules similar to natural language.

- ***Rule base system complexity reduction***

The rule set for the FRBS typically consists of quite a number of rules and including all the attributes (or variables) from the problem inputs. When a rule has too many conditions, it becomes less interpretable. Moreover, for most of the datasets and real-world applications, just a small set of input variables has strategic influence on the output variable. Based on this understanding, it becomes necessary and intuitive to reduce the dimensionality (number of input attributes in the rule) of the rules in the rule set. At the same time, some rules in the rule set might be conflicting with others. For a good FRBS, those conflicting rules must be identified and properly resolved to avoid confusion. On the other hand, some rules could have less impact on the output variable. To make the FRBS more concise, the less important rules could be removed from the rule set if the accuracy of FRBS does not drop too much. So after finishing the construction of FRBS, the rule complexity reduction algorithms are needed.

1.2 Motivation and objectives

As a result of the massive data generated and gathered day by day, we are facing more and more information in daily life. Most of the information we confront is not directly useful to our decision. It becomes critically important how to efficiently extract implicit potentially useful information from the large amount of data. To do this, many statistical techniques have been put forward to set up the data models from the given information. Those statistical models could give reliable accuracy in predicting the future information in same problem domain. Statistical techniques work best when some previous knowledge is known beforehand, e.g. which input variables are more important. Those statistical methods can be classified into two groups, linear regression and nonlinear regression methods. The linear regression is easy to construct and understand. But it becomes difficult to find the appropriate parameters for the linear models when the relationship among input variables is complex and when the number of input variables is large. Nonlinear regression method is powerful in representing the accurate models of complex data. However, those models can be very difficult to interpret. FRBS can professionally process the given training information and present the knowledge learnt from the data in rules, which are easy for people to understand. Also, FRBS has strong ability to process the data entries with a multifaceted relationship and missing information for some variables. FRBS have been widely adopted in data modeling, pattern recognition and data mining area. In many cases, the FRBS trained with the given training information contains more than enough rules in the resulting rule set and the rules in the rule set have some less significant input attributes. Those unnecessary rules and the less constructive input attributes will produce side effects on the prediction of future data and the difficulty in interpretation. The reduction of the complexity becomes more and more prominent in the research in FRBS. This research is intended to efficiently extract precise rules from the dataset and real world control systems.

Our focus is put on the construction of FRBS and the reduction of complexity for the generated FRBS.

The main goal of our research is applying fuzzy logic into the rule-based system construction. We proposed two ways to generate rules for the rule based systems. One is based on fuzzy neural networks. The other is derived from fuzzy c-means clustering algorithm. Both algorithms have the ability to reduce the rule quantity and complexity.

The main objectives in the studies are described as below:

- Construction of the FRBS with FNNs. The FNN was constructed by the genetic algorithm (GA).

- Reduction of complexity, of both dimensionality and quantity (number of rules) of rule-based systems by a pruning process based on thresholds for AND neurons and OR neurons. The selection of optimal thresholds for AND neurons and OR neurons is based on the balance between simplicity of rules and accuracy of FRBS. To gain the optimal model against these two competing objectives, multi-objective optimization is considered.

- Application of FRBS constructed based on FNN to well-known datasets and a real-world application – deployment of wireless sensor networks.

- Construction of FRBS based on fuzzy clustering: the linear approximation functions are defined on clusters in order to evaluate the model accuracy.

- Complexity reduction of FRBS. We have already reduced the quantity of rules when deciding the optimal cluster numbers for a given problem. After the clusters for a given dataset are found, we focus on complexity reduction, i.e., reduction of a number of variables in each rule. Two methods for reduction of variables' number are proposed in this research. One method is based on an optimal ratio for how many input variables should be keep. Another approach is determine isolated input variables in each rule.

- Application of FRBS constructed based on fuzzy clustering and optimized with the two complexity reduction methods to the datasets from the well-known data repositories.

1.3 Outline

The first part of this dissertation is introduction and literature review. Chapters 2-5 provide an extended introduction to the research area and deliver a thorough review on the design of fuzzy rule-based systems. Chapter 2 elaborates on the fundamental concepts and applications of Evolutionary Computing. Three widely used population-based optimization algorithms, namely genetic algorithm (GA), Differential Evolution (DE), and particle swarm optimization (PSO) are discussed and compared. In the sequel, multi-objective genetic algorithms (MOGA) are discussed. The multi-objective optimization applied in this research is implemented with MOGA. The introduction to fuzzy logic and fuzzy neural networks is done in chapter 3. In the same chapter, a detailed literature review of fuzzy rule-based systems (FRBSs) and rule-based system complexity reduction are also covered. In Chapter 4, we look into the fundamentals and practice of fuzzy clustering is presented. The real-world control problem, the deployment of wireless sensor nodes, is tackled in Chapter 5. Some pertinent material on wireless sensor network (WSN) is provided as well. The literature review on the current research progress in general WSN area and the deployment of sensor nodes is presented in detail.

The second part of the dissertation, starting from Chapter 6, is focused on the novel approaches in development of fuzzy rule-based systems with fuzzy neural networks and fuzzy c-means clustering ensuing algorithmic facets of the rule complexity reduction. Chapter 6 demonstrates how to construct the FRBS using FNNs and shows their applications in system modeling and model simplification.

In Chapter 7, we applied the fuzzy rule-based system using FNN, introduced in Chapter 6, to the deployment of the WSN nodes to gain optimal coverage and maximum lifetime of the network. Chapter 8 presents a way of construction of FRBS with the use of fuzzy clustering. We introduce two conceptually different ways to reduce complexity of fuzzy rule-based models in this chapter. Wireless sensor networks (WSNs) become hot research topic in recent decades. The deployment of sensor nodes in WSNs is an important aspect for the quality of WSNs. Conclusions and future work are presented in Chapter 9.

Chapter 2 Evolutionary Computing and Population-based Optimization

In this chapter, we introduce the essence of evolutionary computing and population-based optimization algorithms. Three typical evolutionary computing algorithms, genetic algorithm (GA), differential evolution (DE) and particle swarm optimization (PSO) are discussed in detail. The performance for GA, DE, and PSO is verified with some widely used synthetic functions. The comparison of the three algorithms is done. Multi-objective optimization and the population-based multi-objective optimization method are discussed next.

2.1 Introduction to evolutionary computing and population-based optimization

Generally speaking, Evolutionary Computation [2_2, 2_24] is derived from some principles of Charles R. Darwin’s theory of natural selection. It has been extensively used in solving optimization problems since the 1960s [2_35]. Evolutionary computation focuses on the simulation to evolution process in natural systems. In natural systems, the evolution process involves the change in the biological hierarchy of cells, organs, individuals, and populations. Evolutionary computation takes advantage of the associated optimization mechanisms governed by the mechanisms of evolution. It makes use of biotic evolution methods, such as the reproduction, mutation and recombination, to generate new individuals. A searching strategy is needed to locate optimal candidates to get involve into those biotic evolution methods. The searching strategy simulates the natural rule of competing to survive. With evolutionary computing methods, we are able to easily solve those problems that are difficult or even unable to achieve satisfactory solutions with traditionally popular optimization approaches. Evolutionary computation can be realized with population-based optimization methods.

Population-based optimization begins the searching for the solution coming through a set of random solution candidates. These candidates evolve during generations until the optimal solution shows up or some stopping criteria have been satisfied. Population-based optimization algorithm is typically based on the principle that a population of individuals is processed to find a solution to a problem. The algorithm will change the values in the candidate solution within predefined ranges. The algorithm mimics the natural process of how an individual tries to learn from the best one in the population to gain better chance to locate the optimal solution and how a population evolves through generations by keeping track of the better fitting (showing less gap to the optimal solution) individuals. Genetic Algorithm (GA), Differential Evolution (DE) and Particle Swarm Optimization (PSO) are three most utilized representatives for population-based optimization algorithms. There is one function or one set of functions evaluating the performance of each candidate solution, so as to figure out how close the current candidate is to the optimal solution. The function fulfilling this task in PSO and GA is called the fitness function. The performance measurement in GA and PSO is the higher, the better, so the fitness function should be monotonically increasing. DE uses the similar technique but measures in the reverse way. The objective function in DE is called as the energy function, which indicates the energy needed for the candidate to reach the optimal solution, so the energy is the less, the better. The energy function is a monotonically decreasing one.

The searching space for GA, DE and PSO can be generalized as an n-dimensional normalized hypercube $[0,1]^n$ meaning that each vector, representing a candidate solution, is described as $\boldsymbol{x}_k \in [0,1]^n$, $k=1, 2, \ldots, N$, where N is the population size.

2.2 Genetic Algorithms (GA)

Genetic Algorithm (GA) was introduced by Holland in 1975 [2_16, 2_28]. In essence, GA is a biologically inspired searching method exploiting the principles of natural selection. In this sense, all possible solutions are treated as data values in the form of vector, called chromosomes. Each value in the chromosome is named as a gene. Based on the value type of the genes, GA can be classified into two types: binary coded GA (BCGA) and real coded GA (RCGA) [2_15]. The genes in BCGA are all binary values or integer values, while the genes in RCGA are continuous real numbers. The chromosomes are evaluated based on their performance in solving the problem and ranked by problem oriented objective function, called fitness function. The output of the fitness function indicates the fitness of the chromosome in term of the performance. Fitness presents how close the probable solution is to the optimal solution. The fitness values from each evaluation are used by a chromosome selection function, which adopts some selection mechanism to pick more competitive candidates out. Selected candidates are then used to form the intermediate population. Then, some genetic operators, such as crossover and mutation, are employed to work on intermediate population to form a new population. Crossover works on two candidates from the intermediate population with some probability p_c. Simple one point or two point crossover can be used on BCGA and RCGA. Many variations for crossover operator have been explored for RCGA, for example, arithmetical crossover, BLX-α crossover and linear crossover etc. [2_15]. For mutation, all single genes in the vector are able to be altered to a new value with the probability p_m. For BCGA, a random new integer value with the range will be used to replace the old value. For RCGA, more complex algorithms could also be used to generate the new value, such as non-uniform, discrete modal and continuous modal mutation etc. [2_16]. After mutation, the new candidate will be added to the new population. For different problems, we need to find an appropriate fitness function for GA to utilize. Typical fitness functions are: the reverse of the sum of the training data errors for data modelling, correct classification rate for data classification.

2.3 Differential Evolution (DE)

Differential Evolution (DE) [2_8, 2_9, 2_30] is a population based stochastic function optimizer. It is a scheme for doing optimization with trial parameter vectors. DE adds the weighted difference between two chromosomes to another selected chromosome to generate a new chromosome. Each individual in the population is evaluated by an objective function, called energy function [2_23]. It calculates the energy value for each chromosome. This value indicates the energy consumption for current individual to travel to the optimal position. Thus, the energy is the less the better. Some substantial improvements on DE also apply the crossover GA operator on the new candidate and the existing candidate. Since mutation is the main genetic operator to DE, quite a number of mutation operators have been put forward [2_21, 2_25]. DE has been widely applied to training neural networks [2_34], distributed computation [2_42], system design [2_36], function optimization [2_38, 2_37], image classification [2_22], and electromagnetic [2_27].

2.4 Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) [2_17, 2_37, 2_41, 2_1, 2_11] is also a population based stochastic optimization technique, which was inspired by social behaviour of bird flocking. PSO is initialized with a population of random solution candidates and searches for optimums by changing those candidates through generations. The candidates in PSO do not need to compete to survive. PSO just uses the overall best candidate and the individually best solution to direct the mutation of each candidate [2_43]. The candidate, or chromosome, in PSO is called a particle. Each particle has two related vectors to store both current position and current velocity. The current position contains same information as the chromosome in GA and DE. There is no selection stage for PSO and each particle remains in the population. The update to the particles is done by optimizing the values in current position vector using the values in velocity vector.

At the same time, not like GA or DE, the particle also has a small memory to keep track of its own previous best position, which is the one yielding the highest value of the fitness function found so far [2_4,2_18]. Normally, the objective function for PSO should be monotonic increasing function. The implementation of PSO could use its own way to find the local best particle and global best particle based on the chosen objective function. PSO is proven to be fast in convergence on complex problems [2_29, 2_6]. PSO can also combine with fuzzy logic [2_33]. PSO has been adopted in many research areas, such as neural network training [2_12, 2_13, 2_40], electromagnetic [2_26, 2_5], function minimization or maximization [2_20, 2_32, 2_31].

2.5 Comparison of GA, DE and PSO

All three algorithms are representing evolutionary computing techniques and they are all searching for the optimal solution among a population of candidates. Here is the main difference among the three algorithms.

- ✓ GA and DE are classified as the evolutionary algorithms, while PSO is treated as a swarm intelligence algorithm.

- ✓ Traditional DE only applies simple arithmetic operations among selected candidates, while GA needs the help of genetic operators, like crossover and mutation. Thus, traditional DE is more straightforward and easier to implement comparing with GA.

- ✓ DE and GA both simulate the natural selective approach to maintain the population. The candidate produces the offspring. The offspring has to compete to survive, which means that the offspring having more potential to become a solution will have more chances to be chosen to form the next generation. Candidates in the new population will undergo fine tuning to ensure the diversity and excellence of the population.

- ✓ PSO makes use of a collective searching method among generations. In each generation, candidates exchange their individual best findings and keep a record of their own local best findings. The searching process includes adjustment on two aspects, which are the social communication result (the overall best candidate performance) and the local record (the individual best performance). Each candidate adjusts its behaviour according to behaviour of the leader (the best candidate) in the whole population and its own historical best of the performance.
- ✓ Candidates in PSO do not need to compete to survive and they just improve themselves among iterations, while candidates in DE and GA undergo the selection process to form the new population or take part in the creation of the new population.
- ✓ Particles in PSO have a local memory to keep track of the local history information, while the chromosomes in GA and DE do not have the local memory.
- ✓ GA and DE only keep track of the current state of the candidates. PSO needs to set aside storage for current state (position) and the speed.
- ✓ GA and DE could make use of a variety of genetic operators to fine tune individuals in the new population to maintain both diversity and maturity of the population. PSO only modifies the existing candidates using simple mathematical formulae over their position and velocity based on their own local best position and global best position discovered by the whole society.

2.6 Multi-objective Optimization

In this section, the multi-objective optimization technique is introduced. Pareto approach is defined and other methods used to solve multi-objective optimization problems are also presented.

Optimization can be defined as the search for the best possible solution(s) to a given problem. Real-world problems often entail the optimization of multi-objectives. If these objectives are conflicting, then no best solution exists, but a set of moral compromise solutions.

A multi-objective optimization problem is often a problem to formulate a design in which there are more than one criteria or design objectives [2_39]. Almost every real-world problem involves simultaneous optimization of several incommensurable and often competing objectives. While in single-objective optimization the optimal solution is usually clearly defined, this does not hold for multi-objective optimization problems. If the objectives are opposing, then the problem becomes finding the best feasible design which still satisfies the opposing objectives. An optimum design problem must then be solved, with multi-objectives and constraints being taken into consideration. Instead of a single optimum, there is rather a set of alternative trade-offs. This type of problem is known as a multi-objective, multi-criteria, or a vector optimization problem.

Definitions of a multi-objective problem:

A general multi-objective problem (MOP) includes a set of *n* parameters (decision variables) $\mathbf{x} = \{x_1, x_2, ..., x_n\} \in \mathbf{X}$, where $\mathbf{X} \subseteq \mathbf{R}^n$ is the n-dimensional decision space or solution space, a set of *k* objective functions $f_1(\mathbf{x}), f_2(\mathbf{x})..., f_k(\mathbf{x})$. Objective functions are functions of the decision variables. The optimization goal is to $\min_{x \in X}\{f_1(\mathbf{x}), f_2(\mathbf{x})..., f_k(\mathbf{x})\} where \mathbf{X} \subseteq \mathbf{R^n}$ or $\max_{x \in X}\{f_1(\mathbf{x}), f_2(\mathbf{x})..., f_k(\mathbf{x})\} where \mathbf{X} \subseteq \mathbf{R^n}$ according to different problems. In some cases, these objective functions have a set of *m* constraints functions, which define the boundary of the problem. In the following discussion, we will consider maximizing optimization problem. For each feasible set of decision parameters, we could call it a solution.

In the sections below, we will provide a short survey of several multi-objective optimization methods. In our study, we choose Pareto approach to realize multi-objective optimization.

2.6.1 Pareto approach

Pareto approach is named after Vilfredo Pareto, an Italian economist who used the concept in his studies of economic efficiency and income distribution. Given a set of solutions to the problem, a partial ordering can be found by the principle of dominance: A solution is clearly better than (dominating) another solution, if it is better or equal in all objectives, but at least better in one objective. Using this principle, the set of best compromise solutions is generated by removing all solutions that are dominated by at least one other solution. The remaining solutions are all of equal quality (indifferent). A mutual comparison of always two solutions shows that each one is always better and worse in at least one objective. This set of indifferent solutions is referred to as the Pareto set. Starting from a Pareto solution, one objective can only be improved at the expense of at least one other objective [2_3].

Definition 1 A solution $\mathbf{a} \in \mathbf{X}$ is dominating a solution $\mathbf{b} \in \mathbf{X} (\mathbf{a} \succ \mathbf{b})$ if and only if it is superior or equal in all objectives and at least superior in one objective. This can be expressed as $\mathbf{a} \succeq \mathbf{b}$, if $\forall i \in \{1,2,...,m\}: f_i(\mathbf{a}) \geq f_i(\mathbf{b})$ and $\exists j \in \{1,2,...,m\}: f_j(\mathbf{a}) > f_j(\mathbf{b})$.

When we define a weaker condition for definition 1, we get the definition of weak dominating relationship between two solutions.

Definition 2 A solution $\mathbf{a} \in \mathbf{X}$ is weakly dominating a solution $\mathbf{b} \in \mathbf{X}$ ($\mathbf{a} \succeq \mathbf{b}$) if and only if it is superior or equal in all objectives. This can be expressed as $\mathbf{a} \succeq \mathbf{b}, if \forall i \in \{1,2,...,m\}: f_i(\mathbf{a}) \geq f_i(\mathbf{b})$.

Now we could define the indifferent relationship between two solutions.

Definition 3 The solution $\mathbf{a} \in \mathbf{X}$ is indifferent to a solution $\mathbf{b} \in \mathbf{X}$ ($\mathbf{a} \sim \mathbf{b}$), if and only if neither solution is dominating the other one ($\neg(\mathbf{a} \succeq b) \wedge \neg(b \succeq a)$).

Definition 4 A solution $\mathbf{a} \in \mathbf{X}$ is said to be nondominated regarding a set $\mathbf{A} \subseteq \mathbf{X}$, if and only if $\neg\exists \mathbf{b} \in \mathbf{X} : \mathbf{b} \succ \mathbf{a}$. Here, $\mathbf{X}_f \subseteq \mathbf{X}$ represents the set containing all the feasible solutions we got right now. If and only if the solution $\mathbf{a} \in \mathbf{X}$ is nondominated regarding the set $\mathbf{X}_f$, **x** is said to be Pareto optimal.

When no a priori preference is defined among the objectives, dominance is the only way to determine, if one solution performs better than the other. Furthermore, the best solutions to a multi-objective problem are the nondominated subset among all feasible solutions. These solutions are denoted as the Pareto optimal set, the corresponding objective vectors form the Pareto-optimal front or surface.

Definition 5 Let $\mathbf{A} \subseteq \mathbf{X}_f$. The function *p(A)* gives the set of nondominated decision vectors in ***A***: p(***A***)={ $\mathbf{b} \in \mathbf{A}$ | b is nondominated regarding ***A***}.

The set *p(**A**)* is the nondominated set regarding ***A*** and the result set of objective vectors ***f****(p(**A**))* is the nondominated front (surface) regarding ***A***. In the sequel, if ***A***=$\boldsymbol{X}_f$, the $p(\boldsymbol{X}_f)$ is called the Pareto-optimal set and the set $\boldsymbol{f}(p(\boldsymbol{X}_f))$ is denoted as the Pareto optimal front.

There have been quite a number of implementations for Pareto approach, such as the Pareto archived evolution strategy (PAES) [2_19], the Pareto envelope-based selection algorithm (PESA) [2_7], the strength Pareto evolutionary algorithm 2 (SPEA2) [2_45], Pareto trained with multi objective genetic algorithm (MOGA) [2_10]. In our research, we are using similar MOGA approach to realize Pareto optimization.

2.6.2 Weighted Sum Approach

Weighted sum is a method of secularization of vector functions [2_14].

Definition 6 For a function $F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x})..., f_k(\mathbf{x}))$, which is a vector of target objects and a vector $\mathbf{w} = (w_1, w_2, ..., w_k)$, so that $\sum_{i=1}^{k} w_i = 1$, define

$$Fw(\mathbf{x}) = \sum_{i=1}^{k} w_i \bullet f_i(\mathbf{x})$$

We have transferred the original multi-objective optimization problem into the problem of finding the optimal result for function $Fw(\mathbf{x})$ with a suitable vector ***w***. In this way, we could deal with the original problem using single-objective optimization methods.

A normal way to assign the weight vector is considering the importance of the objectives. The more important objective will get a higher weight, while less important objective will get lower weight. But, usually not all the objectives have the same value range, so these objectives should be normalized before being weighted.

The advantages of using weighted sum are:

a) multi-objective function is reduced to a single-objective function;

b) traditional optimization methods can now be applied.

But the disadvantages are also very obvious. Since the results of solving an optimization problem using weighted sum method can vary significantly as the weighting coefficients change, and since very little is usually known about how to choose these coefficients, a necessary approach is to solve the same problem for many different values of ***w***. But in this case, we are still confronted with the decision of having to choose the most appropriate solution based on our intuition.

2.6.3 The ε-constraint Approach

This method is based on minimization of one (the most preferred or primary) objective function, and considering the other objectives as constraints bound by some allowable levels ε_i. Hence, a single objective minimization is carried out for the most relevant objective function, say $\boldsymbol{f_r}$, subject to additional constraints on the other objective functions. The method could be formulated as follows:

$f_r(\mathbf{x}^*) = \min f_r(\mathbf{x})$, subjects to additional constraints of the form $f_i(\mathbf{x}) \le \varepsilon_i$ for $i=1, 2,..., k$ and $i \neq r$, where ε_I are assumed values of the objective functions which we wish not exceed.

This method is also known as trade-off method, because of its main concept of trading a value of one objective function for a value of another function [2_44]. A problem with this technique is that the obtained new feasible solution set might be empty. This will happen if the lower bounds are not chosen appropriately. In order to avoid this situation, a suitable range for values for the ε_i has to be known beforehand. In other word, problem knowledge may be required for this approach, but the knowledge may not be available sometimes.

2.7 Summary

The three optimization algorithms are all good at searching for the best solution for complex problems. In our following research, we are using GA as the main evolutionary tool. For the multi-objective optimizations, Pareto is proven to be able to provide a set of solutions to meet different trade-off expectation on the competing goals.

Reference

[2_1] P. J. Angeline, “Using selection to improve particle swarm optimization,” in IEEE International Conference on Evolutionary Computation Proceedings, pp. 84-89, 1998 .

[2_2] B.V. Babu. "Evolutionary Computation - At a Glance." NEXUS, Annual Magazine of Engineering Technology Association, BITS, Pilani, pages 3-7, 2001.

[2_3] D. E. Bell, R. L. Keeney, and H. Raiffa (Eds), “Conflicting objectives in decision,” John Wiley & Sons, 1977.

[2_4] D. W. Boeringer and D. H. Werner, “Particle swarm optimization versus genetic algorithms for phased array synthesis,” IEEE Transactions on Antennas and Propagation, 52 (3), pp. 771-779, Mar. 2004.

[2_5] G. Ciuprina, D. Ioan, and I. Munteanu, “Use of intelligent-particle swarm optimization in electromagnetics,” IEEE Transactions on Magnetics 38 (2), pp. 1037-1040, Mar.2002.

[2_6] M. Clerc and J. Kennedy, “The particle swarm - explosion, stability, and convergence in a multidimensional complex space,” IEEE Transactions on Evolutionary Computation 6 (1), pp. 58-73, Feb. 2002.

[2_7] D. W. Corne, J. D. Knowles, and M. J. Oates, “The Pareto envelope-based selection algorithm for multiobjective optimization,” in Proc. Parallel Problem Solving Nature VI Conf., pp. 839–848, 2000.

[2_8] S. Das, and A. Konar, “Two-Dimensional IIR Filter Design with Modern Search Heuristics: A Comparative Study,” International Journal of Computational Intelligence and Applications, vol. 6, no. 3, pp. 329-355, 2006.

[2_9] S. Das, and A. Konar, “A Swarm Intelligence Approach to the Synthesis of Two- Dimensional IIR Filters,” Engineering Applications of Artificial Intelligence, vol. 20, no. 8, pp. 1086-1096, 2007.

[2_10] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” IEEE Trans. Evol. Comput., vol. 6, pp. 182–197, 2002.

[2_11] R. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” in 1995 Proceeding of the Sixth International Symposium on Micro Machine and Human Science (MHS '95), pp. 39-43, 1995.

[2_12] R. Eberhard and X. Hu, "Human Trmor Analysis Using Particle Swarm Optimization," In Proc. of the Congress on Evolutionary Computation, Washington D.C. USA, pp:1927,1930, 1999.

[2_13] A.P. Engelbrecht, "A New Pruning Heuristic Based on Variance Analysis of Sensitivity Information," IEEE Trans. on Neural Networks, 12(6), pp:1386-1399, 2001.

[2_14] J. Guddat, F. G. Vasquez, K. Tammer and K. Wendler, “Multiobjective and Stochastic Optimization Based on Parametric Optimization.” Akademie-Verlag, Berlin, 1985.

[2_15] F. HerrerA, M. Lozano and J.L. Verdegay, "Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis," Artificial Intelligence Review 12: 265–319, 1998. Kluwer Academic Publishers.

[2_16] J. Holland, "Adaptation in Natural and Artificial Systems," University of Michigan Press, Ann Arbor, 1975.

[2_17] J. Kennedy, and R. Eberhart, “Particle Swarm Optimization,” Proceedings of IEEE International Conference on Neural Networks, pp. 1942-1948, 2000.

[2_18] J. Kennedy and R. Eberhart, "Swarm Intelligence." Morgan Kaufmann: San Francisco, CA, 2001.

[2_19] J. D. Knowles and D.W. Corne, “Approximating the nondominated front using the Pareto archived evolution strategy,” Evol. Comput., vol. 8, no.2, pp. 149–172, 2000.

[2_20] C. Li and S. Yang, “An adaptive learning particle swarm optimizer for function optimization,” in Proc. Congr. Evol. Comput., pp. 381–388, 2009.

[2_21] F. Neri and V. Tirronen, “Recent advances in differential evolution: A review and experimental analysis,” Artif. Intell. Rev., vol. 33, nos. 1–2, pp. 61–106, 2010.

[2_22] M. Omran, A. Engelbrecht, and A. Salman, “Differential evolution methods for unsupervised image classification,” in Proc. IEEE Int. Conf. Evol. Comput., vol. 2, pp. 966–973, 2005.

[2_23] K. Price. "An introduction to differential evolution. New Ideas in Optimization," McGraw-Hill, London, pages 79–108, 1999.

[2_24] K. Price and R. Storn. "Differential Evolution: Numerical Optimization Made Easy." Dr. Dobb's Journal, pages 18-24, April 1997.

[2_25] K. Price, R. Storn, and J. Lampinen, "Differential Evolution: A Practical Approach to Global Optimization." Berlin, Germany: Springer, 2005.

[2_26] J. Robinson and Y. Rahmat-Samii, “Particle Swarm Optimization in Electromagnetics,” IEEE Transactions on Antennas and Propagation, 52 (2), pp. 397-407, Feb. 2004.

[2_27] P. Rocca, G. Oliveri,A. Massa, "Differential Evolution as Applied to Electromagnetics," IEEE Antennas and Propagation Magazine, Vol 53 , Issue 1, pp: 38 - 49, 2011.

[2_28] H.P. Schefel, “Evolution strategie and numeric Optimization,” Technische University Berlin, 1975.

[2_29] L. Schoofs and B. Naudts, “Swarm intelligence on the binary constraint satisfaction problem,” in 2002 Proceedings of Congress on Evolutionary Computation (CEC 02), vol. 2, pp. 1444-1449, 2002.

[2_30] Y. Shi, and R. Eberhart, “ Parameter Selection in Particle Swarm Optimization,” Lecture Notes in Computer Science, Evolutionary Programming VII, Springer, pp. 591-600, 1998.

[2_31] Y. Shi and R.C. Eberhard, "A Modified Particle Swarm Optimizer," In IEEE World Congress on Computational Intelligence, pp:68-73, 1998.

[2_32] Y. Shi and R.C. Eberhard, "Empirical Study of Particle Swarm Optimization," In Proc. of the Congress on Evolutionary Computation, Washington D.C. USA, pp:1945-1949, 1999.

[2_33] Y. Shi and R. Eberhart, “Fuzzy adaptive particle swarm optimization,” in Proc. Congr. Evol. Comput., vol. 1, pp. 101–106, 2001.

[2_34] A. Slowik, "Application of an Adaptive Differential Evolution Algorithm With Multiple Trial Vectors to Artificial Neural Network Training," IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 58, NO. 8, pp:3160-3167, 2011.

[2_35] R. Stefoff, "Charles Darwin and the evolution revolution," Oxford: Oxford University Press, 1996.

[2_36] R. Storn. "System Design by Constraint Adaptation and Differential Evolution." IEEE Transactions on Evolutionary Computation, 3(1):22 – 34, 1999.

[2_37] R. Storn, and K. Price, “Differential Evolution - A Simple and Efficient Heuristic for Global Optimization Over Continuous Spaces,” Journal of Global Optimization, vol. 11, no. 4, 1997, pp. 341-359.

[2_38] R. Storn. "On the Usage of Differential Evolution for Function Optimization." NAFIPS 1996, Berkeley, pages 519 – 523, 1996.

[2_39] D. A. Van Veldhuizen and G. B. Lamont, “Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art,” Evolutionary Computation, 8(2):125-147, 2000.

[2_40] F. van den Bergh, "Particle Swarm Weight Initialization in Multi-layer Perceptron Artificial Neural Networks," In Defvelopment and Practice of Artificial Intelligence Techniques, Durban, South Africa, pp:41-45, 1999.

[2_41] G. Venter, and J. Sobieszczanski-Sobieski, “Particle Swarm Optimization,” AIAA Journal, vol. 41, no. 8, pp. 1583-1589, 2003.

[2_42] M. Weber, F. Neri, and V. Tirronen, “Distributed differential evolution with explorative-exploitative population families,” Genet. Programming Evolvable Mach., vol. 10, no. 4, pp. 343–371, 2009.
[2_43] X. Yao, Y. Liu, and G. Lin, “Evolutionary Programming Made Faster,” IEEE Transactions on Evolutionary Computation, vol. 3, 1999, pp. 82-102.
[2_44] E. Zitzler, “Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications”, PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.
[2_45] E. Zitzler, M. Laumanns, and L. Thiele, “SPEA2: Improving the strength Pareto evolutionary algorithm,” Computer Engineering and Networks Laboratory (TIK), Swiss Federal Inst. Technology (ETH), Zurich, Switzerland, 103, 2001.

Chapter 3 Fuzzy Rule Based Systems and Rule Complexity Reduction

In this chapter, we presented the introduction and literature review for the fuzzy rule based system (FRBS) and the rule complexity reduction. The introduction of fuzzy set and fuzzy logic is presented first. Later, artificial neural network and fuzzy neural network (FNN) and the pruning method for FNN are discussed. Detail literature review on FRBS is shown next. Lastly, the rule complexity reduction is discussed. Another method in constructing FRBS, fuzzy clustering, will be deliberated about in next chapter.

3.1 Fuzzy Set and Fuzzy Logic

Fuzzy sets were introduced by L. A. Zadeh [3_59] in 1965. Instead of using 1 or 0 to indicate whether an object belongs to a set or not, fuzzy set uses a value in the unit interval *[0, 1]* to indicate the relationship between an object and the set. This value is determined by a predefined function, called a membership function. Generally speaking, a fuzzy set ***F*** defined in the domain of ***X*** is characterized by a membership function $\mu_F(x), x \in X$ which gives out the values in the range *[0, 1]*. There are different implementations for defining a membership function for the fuzzy set. In many practical usages, fuzzy sets can be represented by the following parameterized functions, such as triangular, trapezoidal, Gaussian functions (shown in Figure 3.1). For more forms of the membership functions, see reference [3_37].

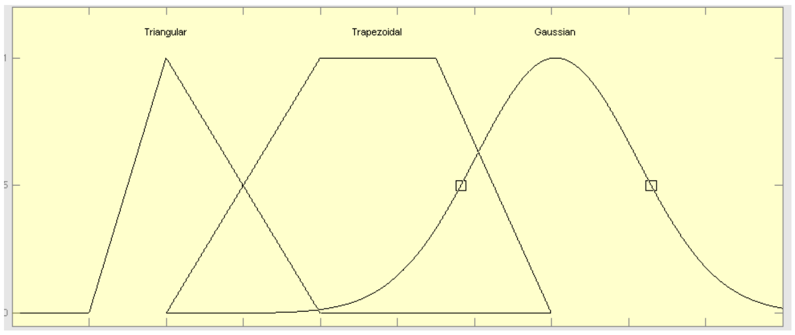


Figure 3.1 Triangular, Trapezoidal and Gaussian functions

For the specific domain, we could define as many fuzzy sets as we need, but it would become more difficult to explain and understand when the number of fuzzy sets grows larger. It is preferable if the number of the fuzzy sets defined in a universe does not exceed 7. Usually we assign linguistic meanings to membership functions, such as low, medium and high for three membership functions of temperature. Refs [3_37, 3_38] provide more information on fuzzy sets and membership functions.

With fuzzy logic, we treat original truth values in two-valued or many-valued logic as a linguistic characterization of numerical truth-values. "Thus, fuzzy logic concerns the principles of approximate reasoning [3_38, 3_36]." By applying the theory of fuzzy logic in solving modeling problems, we are able to generate fuzzy models to represent knowledge. This process is known as fuzzy modeling. According to [3_39], fuzzy modeling is primarily concerned with building models at a certain level of information granularity, which is conveniently quantified in terms of fuzzy sets. Rule-based models have assumed a dominant position in the plethora of fuzzy models. With the continuously growing diversity of modeling rules, one can project a significant development along these lines. Rule-based fuzzy models exploit the calculus of rule-based structures and, in general, can be structured as a series of "IF-THEN" conditional statements of the form:

IF Input_1 is $\boldsymbol{A_{i1}}$ and Input_2 is $\boldsymbol{A_{i2}}$ and … and Input_n is $\boldsymbol{A_{in}}$ THEN output is Y

i=*1, 2,…, c*, where $\boldsymbol{A_{i1}}$, $\boldsymbol{A_{i2}}$, …, $\boldsymbol{A_{in}}$, and ***Y*** are fuzzy sets (linguistic labels) of the corresponding system's variables being defined.

3.2 Artificial Neural Network

An Artificial Neural Network (ANN) is an information-processing paradigm. Artificial neural networks are simulations of biological nervous systems, such as animal brains. They are composed of a large number of highly interconnected processing elements (neurons) working together to solve specific problems. An ANN is defined by the neurons and the connection between these neurons. These connections are measured by having values assigned to them (primarily floating point values), which are called weights. ANNs are trainable, that is they can learn by examples. Through a learning process, an ANN adjusts the weights connecting to neurons based on a set of training data. The neuron, as the basic element to an ANN, usually has a number of connections that send information in/out of the neuron. The neuron processes the input information and produces the output information for other neurons in the ANN, or the network output to the outer system. The typical structure of the neuron looks like that in Figure 3.2.

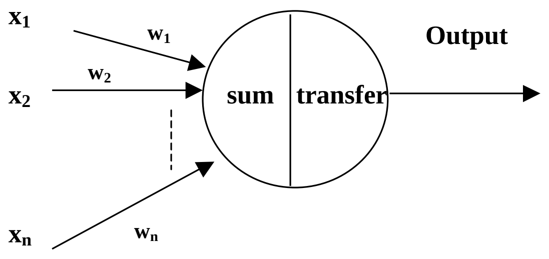


Figure 3.2 the structure of the neuron

The neuron can be considered as having two parts. The first part has responsibility for the aggregation of the input information. In Figure 3.1, the sum is calculated by $sum = \sum_{i=1}^{n} x_i w_i$. If we denote $\mathbf{x}$ as the vector of the inputs and $\mathbf{w}$ as the vector of weights, that is $\mathbf{x} = [x_1, x_2, ..., x_n]$ and $\mathbf{w} = [w_1, w_2, ..., w_n]$, then the calculation of the sum can be expressed by $sum = \mathbf{x} \bullet \mathbf{w}'$. The second part of the neuron applies the transfer function (also called the activation function [3_17, 3_31]) on the sum obtained in the first part, and generates the output of this neuron. There are many selections for the transfer function, such as the sigmoid function $f(x) = \frac{1}{1+e^x}$, the linear function $f(x) = x$, or the hard limit function $f(x) = \begin{cases} 0 & x < 0 \\ 1 & x \ge 0 \end{cases}$. The topology of the neural network can be described by the number of layers and the number of neurons (or nodes) per layer. Types of layers include input, hidden, and output. The topology of a feed forward ANN, which has n inputs, 1 hidden layer containing m nodes, and just one node in the output layer, is shown in Figure 3.3.

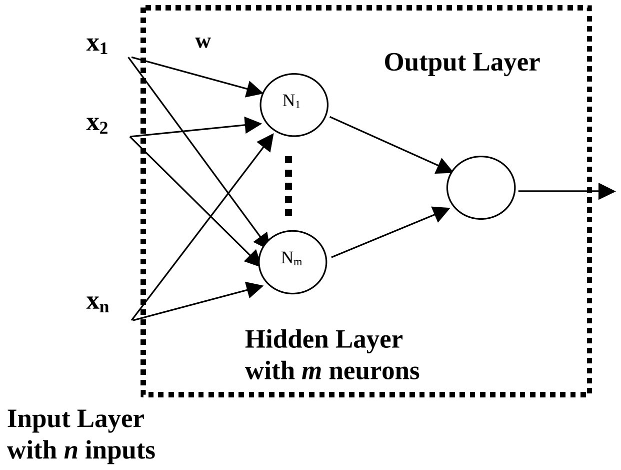


Figure 3.3 The topology of an ANN

3.3 Fuzzy Neural Networks

Fuzzy neural networks are developed based on the principle of artificial neural networks. From above discussion on ANN, one can think of neural networks as structure-free and fully distributed models. The distributivity contributes to profound learning capabilities, as the individual computing elements in the network are capable of adjusting their connections to carry out the best possible mapping. While this feature enhances learning, it makes it almost impossible to come up with a reasonable interpretation of the overall structure of the network worked out in terms of easily understood logical constructs (like “if-Then” statements, frames, etc.). [3_39] In order to make use of the profound learning capabilities of the ANN and gain the ability to interpret the knowledge in logical form, fuzzy neurons are introduced to the family of neural networks [3_36, 3_40]. There are two types of basic fuzzy neurons: *AND* and *OR*. Both types of neurons process information through the use of standard fuzzy set operations like *AND*, *OR*, and *NOT*. (See [3_38, 3_41] for detailed information on these operators.) Let us briefly recall that by an *OR* neuron we mean a fuzzy neuron that achieves a logical mapping from $[0, 1]^n$ into $[0, 1]$ in the following format

$$y = \mathrm{OR}(\mathbf{x};\mathbf{w}) = \mathop{S}_{i=1}^{n}(x_i \ t \ w_i) \tag{1}$$

where x_i denotes the i^{th} input and w_i stands for the associated weight (connection), all of them assuming values in the unit interval. The aggregation operations are implemented using *t*- and *s*-norms (recall that *t*- and *s*-norms are models of logic operators of *AND* and *OR*, respectively). For logic values of 0 and 1 (Boolean logic), s- and t-norms result in standard *AND* and *OR* operators (logic intersection and union). Formally, by a t-norm we mean a two-argument operator mapping $[0, 1]^2$ to $[0, 1]$ such that it is monotonic, associative, commutative, and comes with boundary conditions as $0 \ t \ a = 0$ and as $1 \ t \ a = a$. The same properties hold for s-norms with the exception of the boundary conditions, which are spelled out as $0 \ s \ a = a$ and as $1 \ s \ a = 1$. Given the semantics of the logic operators, we can interpret equation (1) as a logic expression endowed with a collections of weights (connections) $y = (x_1 \text{ and } w_1) \text{ or } (x_2 \text{ and } w_2) \text{ or } \ldots \text{ or } (x_n \text{ and } w_n)$. We use a convenient shorthand notation $y = OR(x; w)$ by collecting all inputs and connections into two vectors, $x = [x_1 \ x_2 \ldots \ x_n]^T$, $w= [w_1 \ w_2 \ldots \ w_n]^T$. This representation helps emphasize the character of processing and underlines the available parametric flexibility of the module residing within its connections.

The AND neuron is ruled by the expression

$$y = \mathrm{AND}(\mathbf{x};\mathbf{w}) = \mathop{\mathrm{T}}_{i=1}^{n}(x_i \; s \; w_i)$$

(2)

where in comparison with the previous construction, the order of aggregation operations has been reversed. Again, in terms of the abbreviated notation, we arrive at the expression, *y = AND(x; w)*.

3.4 Rule Based System

Rule-based system (RBS) is to make use of expert knowledge to solve real world problems. The rule-based system could be used as a way to capture and refine human expertise through the computer and facilitates problem solving [3_22, 3_18]. The inference engine and the rule base are two main parts of a rule-based system [3_50]. The inference engine is the problem solving model that processes the real world data making use of the rules in the rule base. The rule base consists of a set of *IF-THEN* rules. Those rules have the general form as "*IF A THEN B*". *A* and *B* are propositions with linguistic variables. *A* is the rule condition, usually called as the collection of the premises and *B* is called as the consequence or conclusion of the rule. Rule condition can be single or a combination of various variables and be combined using *AND* or *OR* operators. The *AND* operator indicates a union of the two conditions and the *OR* operator represents an intersection of the two conditions. To simplify the conditions, sometimes a *NOT* operator can be used before a condition to represent the negation for that condition. Above three operators are commonly used basic ones. There are some other operators applied, but all can be represented by these three basic operators. Those rules are summary of the knowledge about the domain. The rules can be obtained mainly through two ways. One is summarizing the expert knowledge. Expert knowledge is often presented as a set of rules or as the data stored in the computer [3_22]. Another way is to extract rules from data gathered from field or laboratory experiments. Sometimes, the combination of the expert knowledge and the rules generated could give more meaningful and accurate rules.

A rule-based system provides a way to capture and refine human expertise electronically and solves problems [3_18, 3_22, 3_30]. There are quite a number of advantages of rule-based systems. RBS is clear and vivid in explaining the conclusion. It is easy to improve the quality of a RBS by adding more new rules or tuning old ones. No need to modify the source code, which is proven to be difficult. RBS uses inexact reasoning and is capable of dealing with incomplete fuzzy information. This way of solving problem closes to human intelligence. The shortcoming of RBS is also obvious. The accuracy is largely depending on the correctness and completeness of rules. RBS will work well if the solid expert knowledge or well maintained rule set is available for the problem. But in some area, lack of human experts or high costs of obtaining rules from the data will make RBS work poorly. So it is an important research aspect how to obtain complete correct rule set from the data.

Rule based systems have been proven successful and powerful in various research fields. Newell and Simon [3_33] used rule based systems to model human problem solving behaviours in early 1970s. For a long time, quite a number of rule-based algorithms have been proposed and proved to be useful in solving complex decision making problems when normal algorithmic/mathematical solutions are either unknown or proven inefficient [3_4, 3_9, 3_21, 3_26]. The use-cases of RBS include demand forecasting [3_3, 3_8, 3_14, 3_54], production scheduling [3_9, 3_19, 3_53], supply chain control [3_46] and inventory control [3_34, 3_42, 3_52]. Like for demand forecasting, a paper by Armstrong [3_3] reported the progress made over the past twenty years. The paper demonstrated a series of empirical studies on forecasting methods. Armstrong summarized that no single method has performed well across all types of data, all forecast horizons and all situations. The conclusion made by the paper is that a rule-based method provides a way to put experts' domain knowledge and traditional forecasting models together to produce more accurate forecasts. Recently, more researchers have put their effort in developing rule-based systems to work together with traditional operations research techniques to produce more accessible and practical methods [3_10, 3_13, 3_47].Some other researchers [3_9, 3_10, 3_13] put forward their theories and experimental results to prove that rule-based systems is able to formulate an enterprise operation with a multiplicity of representations that integrate historical and real-time data and provide quick and realistic solutions.

3.5 Fuzzy rule based system

In order to get more concise description of the real problem, brainstorming on the actual problem using expert knowledge is commonly used, especially little or no actual data available. Fuzzy logic is important to help in analyzing practical applications which involve poor-defined and uncertain concepts [3_43]. There are two types of uncertainties, randomness and fuzziness. The concept of probability is used on randomness and the concept of possibility theory is on fuzziness [3_16, 3_43]. The important aspect about the concept of possibility theory is that this is closely matching the imprecision in human cognition [3_29]. The concept of possibility (fuzzy logic) has a particularly critical role in the management of uncertainty in rule based systems. Conventional modeling methods make use of exact accurate parameters to achieve a precise solution. However, in real situations, the accuracy provided by conventional models is not good enough to fit the changing situation. A rule-based model has the ability to capture the physics of a numerical model with a set of rules which replace the complex equation solvers with general rules that requires less precise input parameters. For conventional models, if the input data have inaccurate or uncertain information, the models will become instability or provide a wrong solution. But inaccurate or uncertain inputs are always common in real world problems. Conventional models which use physical theories and equations may be difficult to interpret in real field situations. As mentioned before, fuzzy models have the ability to provide a reasonably accurate solution even there are inaccurate data. In addition, fuzzy models have the ability to make use of knowledge from experts and extract knowledge from raw data. That knowledge can be expressed in the form of rules to provide a solution for the problem. It is much easier for the end-users, even non-technical users to understand the solution which is written in rules similar to natural language. Most of the facts and rules consist of fuzzy predicates. Fuzzy rules expand linguistic IF-THEN rules. Non-fuzzy rules have serious deficiencies in solving real world problems when the reliability of decisions can be the most important goal. The fuzzy logic assisted rules utilize fuzzy sets in the premise and the conclusion. Fuzzy rules are generalized form of the relationships of variables. The main goal in using fuzzy rules is to provide better description in the similar way as human reasoning with ideas and statements. The FRBS structure is shown in figure 3.4.

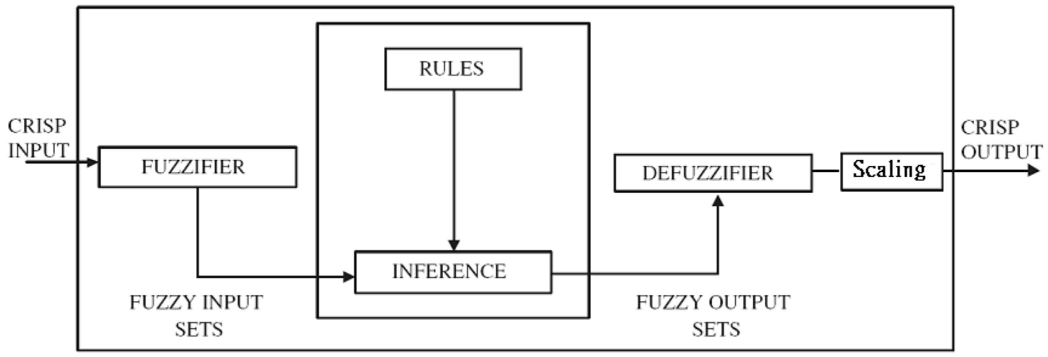


Figure 3.4 Functional structure of Fuzzy Rule Based System

The fuzzy rule-based system can be described as below.

Rule 1: IF $\underline{x}$ *is* $A_1(\underline{x})$ *THEN* $y=f_1(\underline{x})$

Rule 2: IF $\underline{x}$ *is* $A_2(\underline{x})$ *THEN* $y=f_2(\underline{x})$

...

Rule n: IF $\underline{x}$ is $A_n(\underline{x})$ THEN $y=f_n(\underline{x})$

Then the final output the fuzzy rule-based system is expressed as follows

$$y = \frac{\sum_{i=1}^{n} A_i(\underline{x}) f_i(\underline{x})}{\sum_{i=1}^{n} A_i(\underline{x})} \qquad (3)$$

3.6 Rule Complexity Reduction

The spectacular increase in the amount of data is not only found in the number of samples collected, for example over time, but also in the number of attributes. This high dimensionality of datasets leads to the phenomenon known as the curse of dimensionality where computation time is an exponential function of the number of the dimensions. It is often the case that the model contains redundant rules and/or variables. When faced with difficulties resulting from the high dimension of a space, the ideal approach is to decrease this dimension, without losing relevant information in the data. If there are large number of rules and/or attributes in each rules, it becomes more and more vague for the user to understand and difficult to put into practice. Rule complexity reduction can circumvent this problem by reducing the number of attributes in each rules and the number of rules. This can also reduce the computation time, and the resulting rule based systems take less space to store. Models with simpler rules and small number of rules are often easier to interpret. The main drawback of rule complexity reduction is the possibility of information loss. Our main goal here is to reduce the complexity of the rule-based system while keeping reasonable accuracy on the information.

In the last decade, much research has been done to search for practically feasible, but still sufficiently good predictive models for function approximation. Those models are mostly of Takagi-Sugeno-Kang (TSK) type [3_49, 3_51], where the rule consequents could be represented in the form of linear function of the inputs:

R_i: if x_1 is A_{i1} and ... x_n is A_{in} then $y_i=\mathbf{a_i}\mathbf{x}+\mathbf{b_i}$

where $\mathbf{a_i}=[a_{i1},...,a_{in}]$ and $\mathbf{b_i}=[b_{i1},...,b_{in}]$ are the vector of the linear function parameters, $\mathbf{x}=[x_1,...,x_n]^T$ is the input vector. When there are *C* rules in total in the fuzzy rule based system, the overall output for the system can be written with the weighted sum of each individual rule output: $y=\sum_{i=1}^{C}\omega_i(x)y_i$, where $\omega_i(x)$ is the activation function for i^th rule on the given input vector.

Rule complexity reduction has become an important issue in data mining when dealing with data having high number of input attributes. Cutting off the number of rules or number of attributes in each rules are two main methods to reduce rule complexity. To calculate the minimal reduction on rule complexity has been verified is an NP complete problem [3_6]. Fuzzy rule interpolation was one of the first approaches to reduce the complexity of fuzzy models [3_27, 3_28]. The main idea behind this approach is that if some rules can be estimated by the linear representation of other rules in the rule set, these rules can be eliminated from the rule set. Now the attribute reduction algorithms for rough set are another important application of the complexity reduction. It mainly includes the methods of immune mechanism [3_57] and computation intelligence algorithms, such as genetic algorithm [3_60] and Particle Swarm Optimization [3_45].

Variable importance measures for supervised learning are closely related to variable/attribute selection and they are important for improving both learning accuracy and interpretability. Tree ensembles such as random forest are widely used in measuring variable importance [3_2, 3_15]. The attribute selection method [3_20], selected relevant features in a forward phase and removed redundant variables in a backward phase.

In the construction of fuzzy rule based system, we have been witnessing a wealth of design strategies and detailed algorithms involving the technology of Evolutionary Computing and neurocomputing. Just recent developments reported in this realm can be found in a series of studies [3_56, 3_11, 3_24]. Predominantly, the development of fuzzy models is guided by the criterion of accuracy. Another fundamental criterion being at the heart of fuzzy modeling is interpretability (transparency) of resulting fuzzy models. This criterion is central to fuzzy models however its multifaceted nature requires a thorough formulation, quantification of essential aspects of interpretability and subsequently calls for advanced optimization techniques supporting the realization of the ensuing design.

The concept of interpretability of fuzzy rule-based models has been around for several decades and attracted a significant deal of attention. The transparency of fuzzy models is one of the outstanding and important features of fuzzy models. In contrast to the criterion of accuracy, whose quantification is relatively straightforward and easy to come up with performance indexes, transparency of fuzzy rules is more difficult to describe. What makes the fuzzy rule-based easier to interpret is still an open issue. It is quite subjective to assess and in one way or another invokes a factor of subjective judgment given that a human user is ultimately involved in the evaluation process. What become very much apparent are a multifaceted nature of the problem and a multitude of various approaches supported by various optimization technologies including evolutionary optimization. When it comes to the main factors worth considering when discussing a concept of interpretability, we can enumerate a list of factors that may be involved in the reduction process:

3.6.1 Reducing number of rules forming a rule base of the model

Ref. [3_5] presented a novel method to extract the generic bases of fuzzy association. Two data sets, chess and mushroom, were shown in the experiments. The results showed that the reduction of the rule number could be cutting off 99.1-99.2% of rules. In order to avoid information loss, the paper combined the rule bases got from generic basis for exact fuzzy association rules (GBEF) and generic basis for transitive fuzzy association rules (RTF) to form the final rule bases. Ref. [3_1] proposed a new post-processing approach to perform an evolutionary lateral tuning of membership functions, with the main aim of obtaining linguistic models with higher levels of accuracy while maintaining good interpretability. A new rule representation scheme using the linguistic 2-tuples representation model has been considered. The rule selection method worked with lateral tuning technique to reduce the number of rules and the accuracy of the models. The experiment on synthetic data and fuzzy control of an HVAC system showed that the advantages of the new approach.   This paper will be working with other fuzzy rule-based system constructing algorithm to tune the rules in the rule set, so the performance on this algorithm is somewhat related to the rule constructing algorithm. Similar research could also be found in ref. [3_44, 3_56].

3.6.2 Reducing number of sub conditions (input variables) forming a condition part of a given rule

In ref. [3_25], the paper applied fuzzy equivalence classes into the attribute reduction for rough set. The comparison was done to rough set crisp attribute reduction and fuzzy rough set attribute reduction. Web bookmark classification dataset was used in the experiment. The results shown that both crisp rough set attribute reduction and fuzzy rough set attribute reduction can cut off about 99% of the input attributes. The accuracy with fuzzy rough was better than crisp rough attribute reduction. In ref. [3_55], fuzzy models and decision tree models were adopted to construct the rule based systems. After the models gained, the feature selection technique was applied. The results show that the fuzzy objective function outperformed classical feature selection. Ref. [3_35] brought up a distance measure based rough set attribute reduction algorithm. The algorithm was compared with 4 existing rough set attribute reduction methods: rough set attribute reduction, tolerance rough set model, fuzzy rough feature selection and principal component analysis. The results showed that the new algorithm, although in its preliminary stage, could finish the selection process in similar time, with acceptable accuracy and reduced number of attributes. Ref. [3_7] presented a fuzzy rule-based system to construct the fuzzy rule set for online data modeling. During the construction, the new fuzzy sets can be added and the attributes used in the rules can be reduced to improve interpretability. The attribute reduction is done by removing the attributes in the rule sets that have only one fuzzy set defined. The algorithm here constructs the rule consequence and the system topology with the online data input/output, without any offline training. Ref. [3_12] was mainly focusing on selecting important features (attributes) to use in the fuzzy rules, while the paper did the feature selection at the same time of constructing the fuzzy rule based systems. The results on the machine learning datasets showed that the accuracy did not drop too much comparing to use all the features when a significant amount of features were reduced. More reading could be done in refs [3_23, 3_32, 3_58].

3.6.3 Reducing number or rules and the number of input variables,

In ref. [3_24], Iris, credit approval and wine data were used. From the result, the minimum number of rules was all equal to number of classes in the output variable. The results were compared with related references on same data and found comparable. This algorithm is combining the reduction of number of rules and number of variables. Similar research could be found in ref. [3_11]. Synthetic data derived from given functions were used in the experiments and compared with related reference. From the comparison, the algorithm in the paper could reduce the number of fuzzy sets and rules, and improving the accuracy at the same time. The rules are simplified by combining similar fuzzy sets, removing useless rules and merging similar rules. Ref. [3_48] proposed a way to construct the fuzzy rule-based system using GA to generate rules, and then reduce the system complexity by removing redundant rules, reducing attributes in rules and fuzzy set numbers. The attribute reduction is done by hiding one attribute for each rule in iteration, until the accuracy dropped too much. Reducing fuzzy sets was trying to remove uncovered ones in the rules to simplify the fuzzy rule based system.

As a result, given this diversity of possible ways of reduction of rules, it is difficult to quantify the effect of reduction. For instance, it is not always clear if it would be better to have a larger number of simple rules (whose condition parts are linear functions) or a smaller number of rules of a more complex conclusion parts (those of polynomial form).

3.7 Summary

Fuzzy rule-based system (FRBS) is a well-known tool to extract rules from the training objects and apply the rules to predict the future testing objects. Fuzzy neural network is a good way to construct the FRBS. In our research, we introduced our way to construct the FRBS and prune the system with the help of multi-objective optimization methods. After verified the proposed FRBS construction/simplification approach on commonly used datasets, we applied the similar approach to a real world control problem, wireless sensor network (WSN) deployment. FNN is used to construct the FRBS for the deployment of the sensor nodes in WSN. Then simple pruning approach and multi-objective optimization based pruning are used to reduce the complexity of the FRBS to get simpler rule set.

Reference

[3_1] R. Alcalá, J. Alcalá-Fdez, and F. Herrera, "A Proposal for the Genetic Lateral Tuning of Linguistic Fuzzy Systems and Its Interaction With Rule Selection," IEEE TRANSACTIONS ON FUZZY SYSTEMS, VOL. 15, NO. 4, pp:616-635, AUGUST 2007.

[3_2] A. Altmann, L. Tolos¸i, O. Sander, and T. Lengauer, “Permutation importance: a corrected feature importance measure,” Bioinformatics, vol. 26, no. 10, pp. 1340–1347, May 2010.

[3_3] J.S. Armstrong "Findings from evidence-based forecasting: Methods for reducing forecast error" International Journal of Forecasting 22 3 583-598, 2006.

[3_4] Y. Arzi, L. Iaroslavitz "Neural network-based adaptive production control system for a flexible manufacturing cell under a random environment" IIE Transactions 31 3 217-230, 2006.

[3_5] S. Ayouni, S. Yahia, Anne Laurent, "Extracting compact and information lossless sets of fuzzy association rules," Fuzzy Sets and Systems, Volume 183, Issue 1, Pages 1-25, 16 November 2011.

[3_6] A. L. Blum and R. L. Rivest. "Training a 3-node neural networks is NP-complete." Neural Networks 5, 117 – 127, 1992.

[3_7] A. Cara, H. Pomares, and I. Rojas, "A New Methodology for the Online Adaptation of Fuzzy Self-Structuring Controllers," IEEE TRANSACTIONS ON FUZZY SYSTEMS, VOL. 19, NO. 3, pp: 449-464, JUNE 2011.

[3_8] R. Carbone, J.S. Armstrong "Evaluation of extrapolative forecasting methods - results of a survey of academicians and practitioners" Journal of Forecasting 1 2 215-217, 1982.

[3_9] T.S. Chan and K.H. Chan. "A comprehensive survey and future trend of simulation study on FMS scheduling." Journal of Intelligent Manufacturing, 15(1):87-102, 2004.

[3_10] Y.L. Chan, Cheung, "Knowledge-based simulation and analysis of supply chain performance" International Journal of Computer Integrated Manufacturing 19 1 14-23, 2006.

[3_11] M. Chen, D.A. Linkens, "Rule-base self-generation and simplification for data-driven fuzzy models," Fuzzy Sets and Systems, Volume 142, Issue 2, Pages 243-265, 1 March 2004.

[3_12] Y. Chen, N.R. Pal, and I. Chung, "An Integrated Mechanism for Feature Selection and Fuzzy Rule Extraction for Classification," IEEE TRANSACTIONS ON FUZZY SYSTEMS, VOL. 20, NO. 4, pp:683-698, AUGUST 2012.

[3_13] C.F. Cheung, M.F. Wang, and S.K. Kwok. "Knowledge-based inventory management in production logistics: a multi-agent approach." Journal of Engineering Manufacture, 219(2):299-307, 2005.

[3_14] F. Collopy, J. S. Armstrong "Rule-based forecasting: development and validation of an expert systems approach to combining time series extrapolations" Management Science 38 10 1394-1414, 1992.

[3_15] H. Deng, G. C. Runger, and E. Tuv, "System monitoring with real-time contrasts," Journal of Quality Technology, 2010.

[3_16] D. Driankov, H. Hellendoorn and M. Reinfrank, "An introduction to Fuzzy Control", Berlin Heidelberg, Springer-Verlag 1993.

[3_17] L. Fu, "Neural Networks in Computer Intelligence," McGraw-Hill, 1994.

[3_18] J.C. Giarratan and G. Riley. "Expert Systems Principles and Programming, fourth edition," Boston: Course Technology, 2005.

[3_19] E. Gundogar, E. Gundogar "A rule-based master production scheduling system for an electro-mechanical manufacturing company" Production Planning & Control 10 5 486-492, 1999.

[3_20] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," Machine Learning, vol. 46, no. 1-3, pp.389–422, 2002.

[3_21] M.L. Fisher, J.H. Hammond, W.R. Obermeyer, and A. Raman. "Recent trends in modeling of deteriorating inventory." Harvard Business Review Article, 1994.

[3_22] F. Hayes-Roth. "Rule-based systems." Communications of the ACM, 28(9):921-932, 1985.

[3_23] Q. Hu, W. Pan, Lei Zhang, D. Zhang, etc., "Feature Selection for Monotonic Classification," IEEE TRANSACTIONS ON FUZZY SYSTEMS, VOL. 20, NO. 1, pp: 69-81, FEBRUARY 2012.

[3_24] H. Ishibuchi, T. Yamamoto, "Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining," Fuzzy Sets and Systems, Volume 141, Issue 1, Pages 59-88, 1 January 2004.

[3_25] R. Jensen, Q. Shen, "Fuzzy-rough data reduction with ant colony optimization," Fuzzy Sets and Systems, Volume 149, Issue 1, Pages 5-20, 1 January 2005.

[3_26] G. Kahn, S. Nowlan, and J. McDermott. "Strategies for knowledge acquisition." IEEE Transactions on Pattern Analysis and Machine Intelligence, (3):511-522, 1985.

[3_27] L.T. Kczy and K. Hirota, "Approximate reasoning by linear rule interpolation and general approximation," Internat. J. Approx. Reason., vol. 9, pp. 197-225, 1993.

[3_28] L.T. Kczy and K. Hirota, "Size reduction by interpolation in fuzzy rule bases," IEEE Trans. on SMC, vol. 27, pp. 14-25, 1997.

[3_29] S.L. Kohout, "Theories of Possibility," International Journal of Fuzzy Sets and Systems, Vol. 23-3, pp. 357-367, 1988.

[3_30] A. Ligęza. "Logical Foundations for Rule-based Systems." Springer-Verlag, 2006.

[3_31] C. G. Looney, "Pattern Recognition Using Neural Networks," Oxford University Press, 1997

[3_32] P. Maji and S. K. Pal, "Feature Selection Using f-Information Measures in Fuzzy Approximation Spaces," IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 22, NO. 6, pp: 854-867, JUNE 2010.

[3_33] A. Newell and H.A. Simon. "Human Problem Solving." Prentice Hall, 1972.

[3_34] M. Parlar. "Expim: a knowledge-based expert systems for production/inventory modeling." Int. J. Prod. Res., 27(1):101-108, 1989.

[3_35] N. Parthala, Q. Shen, and R. Jensen, "A Distance Measure Approach to Exploring the Rough Set Boundary Region for Attribute Reduction," IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 22, NO. 3, pp:305-317, MARCH 2010

[3_36] W. Pedrycz, "Fuzzy neural networks and neurocomputations," Fuzzy Sets and Systems, v.56 n.1, p.1-28, May 1993.

[3_37] W. Pedrycz, "Fuzzy Control and Fuzzy Systems (2nd edition)," Taunton, NY: Research Studies Press/J. Wiley, 1993

[3_38] W. Pedrycz, F. Gomide, "An Introduction to Fuzzy Sets," Analysis and Design. MIT Press, 1998.

[3_39] W. Pedrycz and M. Reformat, "Rule-Based Modeling of Nonlinear Relationships." IEEE Transactions on Fuzzy Systems, Vol.5, No.2, pp.256-269, May 1997.

[3_40] W. Pedrycz, A. Rocha, "Fuzzy-set based models of neurons and knowledge-based networks," IEEE Trans. on Fuzzy Systems, Volume 1, Issue 4, pp. 254-266, Nov. 1993.

[3_41] W. Pedrycz, A. Rocha, "Knowledge-based neural networks," IEEE Trans. on Fuzzy Systems,1, 254-266, 1993.

[3_42] E. Prater, G.V. Frazier, etc.. "Future impacts of RFID on e-supply chains in grocery retailing", Supply Chain Management: An International Journal, 10 (2), 134-142, 2005.

[3_43] J. Rothfeder, "Minds Over Matter: A New Look at AI," Harvester Press, Sussex, 1985.
[3_44] J.J. Shann, H.C. Fu, "A fuzzy neural network for rule acquiring on fuzzy control systems," Fuzzy Sets and Systems, Volume 71, Issue 3, Pages 345-357, 12 May 1995.
[3_45] Y. Shi, X. Zheng, X. Wang. "Attribute Reduction Based on Quantum-behaved Particle Swarm Optimization," [J]. Computer Engineering, 34(18):65-67,2008.
[3_46] E.A. Silver "Operations-research in inventory management - a review and critique" Operations Research 29 4 628-645, 1981.
[3_47] S.F. Smith. "Knowledge-based production management: approaches, results and prospects." Production Planning and Control, 3(4):350-380, 1992.
[3_48] D.G. Stavrakoudis, G.N. Galidaki, I.Z. Gitas, and J.B. Theocharis, "A Genetic Fuzzy-Rule-Based Classifier for Land Cover Classification From Hyperspectral Imagery," IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, VOL. 50, NO. 1, pp:130-148, JANUARY 2012
[3_49] M. Sugeno and G. T. Kang, “Structure identification of fuzzy model,” Fuzzy Sets and Systems, vol. 28, pp. 15-33, 1988.
[3_50] P.H. Sydenham and R. Thorn, edited by, “Handbook of Measuring System Design”, John Wiley & Sons, Ltd. ISBN: 0-470-02143-8.p-910-912, 2005.
[3_51] T. Takagi and M. Sugeno, “Fuzzy identification of systems and its applications to modeling and control,” IEEE Trans. on SMC, vol. 15, pp. 116-132, 1985.
[3_52] I.P. Tatsiopoulos, N.D. Mekras "An expert system for the selection of production planning and control software packages" Production Planning & Control 10 5 414-425, 1999.
[3_53] G. Tuncel, "A heuristic rule-based approach for dynamic scheduling of flexible manufacturing systems", Multiprocessor Scheduling: Theory and Applications, Itech Education and Publishing, Vienna, Austria., 2007.
[3_54] M. Vellasco, M. Pacheco, L. Neto "Electric load forecasting: evaluating the novel hierarchical neuro-fuzzy BSP model," International Journal of Electrical Power & Energy Systems 26 2 131-142, 2004.
[3_55] S. M. Vieira, J. M.C. Sousa, Uzay Kaymak, "Fuzzy criteria for feature selection Original Research Article," Fuzzy Sets and Systems, Volume 189, Issue 1, Pages 1-18, 2012.
[3_56] N. Xiong, L. Litz, "Reduction of fuzzy control rules by means of premise learning – method and case study," Fuzzy Sets and Systems, Volume 132, Issue 2, Pages 217-231, 2002.
[3_57] Z.f. Xu, G. Chun. "Reduction of Rough Set Attribute Based on Immune Mechanism," J. computer engineering, 33(23):51-53,2007.
[3_58] Y. Yao, J. Mi, Z. Li, "Attribute reduction based on generalized fuzzy evidence theory in fuzzy decision systems," Fuzzy Sets and Systems, Volume 170, Issue 1, Pages 64-75, 2011.
[3_59] L. A. Zadeh, "Fuzzy sets." Information and Control 8: 338-353, 1965.
[3_60] M. Zhao, K. Luo, X. Liao. "Rough set attribute reduction algorithm based on Immune Genetic Algorithm," J. computer engineering and applications, 43(23):171-173,2007.

Chapter 4 Fuzzy clustering

Clustering is the way in analyzing the sample data by assigning those data into a number of groups. With given method of calculation of the similarity, all the data within same group are more similar to each other than to the rest data in other groups. Those groups are called clusters. Fuzzy logic has also been applied in clustering research area. Fuzzy C-means (FCM) is a widely used fuzzy clustering algorithm to be adopted in our research. Fuzzy rule based system (FRBS) using linear function approximation with FCM to model the data is introduced next. Then the complexity reduction on FCM based FRBS is discussed.

4.1 Data Clustering

Clustering provides insight understanding to the data by dividing the data objects into groups (or clusters) of objects [4_26], so the objects within same cluster are more similar to each other and more dissimilar to the rest objects in other clusters. It has been an important research topic in a wide variety of application area for a long time, such as information mining [4_44, 4_46], market research [4_8], psychology and social science [4_30], bioinformatics and so on [4_14,4_29]. Clustering partitions the input space of the data into *C* regions calculated with the similarity/dissimilarity measures. The value of *C* is the number of clusters or partitions. It could be defined before partitioning or remain unknown until the partitioning process is done. Clustering algorithm is a branch in pattern recognition algorithm. Clustering approach could be carried out with little or without supervision. Generally, clustering is categorized into unsupervised learning approach. Unsupervised learning approach is useful to find something of interest for those data with unlabeled samples. There has been quite a number of different clustering implementation. It is not easy to tell which one is the best. No one approach is universally applicable. According to ref [4_33, 4_42], the clustering algorithms could be classified into two categories: hierarchic vs. non-hierarchic methods. The hierarchic methods breaks data objects into clusters that have nested structures. The structure of the methods is much like decision trees. The objects can be put into proper cluster by travelling along the structure. Non-hierarchic methods refer to those cannot be divided into nested clusters, also called partition clustering methods. Many real world problems are unable to be clustered using hierarchic methods, thus more efforts from researchers have been put into non-hierarchic methods. Non-hierarchic methods produce separate clusters by iteratively applying optimization algorithms on the data based on a clustering criterion function or an objective function [4_36]. Based on the resulting clusters relationship, non-hierarchic methods can be further grouped into overlapping methods or non-overlapping methods. Typical methods in this category include crisp c-means for non-overlapping methods, fuzzy c-means for overlapping methods. Crisp c-means algorithm classifies the objects in the data into Boolean condition based clusters. Every individual object in the data will only belong to one cluster. The degree for each object for a specific cluster will be either 1 (belonging to) or 0 (not belonging to). Fuzzy c-means algorithms assigns a real value to the object on each cluster to indicate the degree of the membership of the object related to the cluster. The degree value on the cluster indicates how well the object fit in the cluster. The degree values of each object belonging to the clusters are stored in a matrix, usually called partition matrix. If the total number of objects to be processed is N, the size of the partition matrix will be $C\times N$. The partition matrix could be written in the form $U=\{u_{in} | n=1..N, i=1..C\}$, u_{in} is the degree of the n^{th} object to i^{th} cluster. For crisp clustering, u_{in} is either 0 or 1. As for fuzzy c-means while $0\le u_{in}\le 1$, it indicates that the object belongs to all the clusters, just with different fuzzy membership grades [4_24]. For a single object, the grades on all clusters come to 1, that is $\sum_{j=1}^{C} u_{jn}=1, n=1..N$. Crisp clustering algorithms were discussed in ref. [4_13, 4_27]. Fuzzy c-means brings fuzzy logic concepts into the data clustering technique so as to improve the interpretability [4_36].

4.2 Fuzzy C-Means (FCM)

FCM was developed by Dunn in 1973 [4_12] and later improved by Bezdek in 1981 [4_3]. Since its inception, it has been widely applied to unsupervised data analysis [4_49, 4_26]. For example, fuzzy clustering has been used to pattern recognition [4_3], medical database exploration [4_2] and analysis of traffic data [4_35] and directing traffic control [4_39]., Clustering realized within this setting is realized by minimizing a certain objective function [4_3, 4_12]. There has been a great deal of researchers also tried to use evolutionary computation techniques to train FCM. Ref. [4_32, 4_10] applied differential evolution algorithms into FCM. In [4_31], a variable length chromosome GA was used to find optimal partition matrix and prototypes. In [4_18], the authors begin by establishing that calculus-based optimization methods, like FCM, often get stuck at local minima. To address this problem, the authors propose a GA-based fuzzy clustering method that promises good initial centers and avoids local minima. The algorithm will terminate when the maximum number of allowed generations is reached. The authors observed that a major drawback of their algorithm is its computational complexity. When testing the Magnetic Resonance dataset, the authors had to divide the dataset into subsamples and run tests on each subsample separately so that the algorithm would converge under a day's time. This excessive complexity can be attributed to the feature-level operations involved as opposed to the general gene-level. Same group of authors put forward some continuing work in [4_19]. The authors acknowledge that FCM suffers from being extremely sensitive to the initialization process. Depending on the initial cluster centers selected the algorithm can yield arbitrary results, including convergence to local minima. To solve this limitation, the authors propose a genetic guided algorithm that avoids local minima and minimizes the dependency on initialization. Clustering based on kernel methods are proven to be robust [4_34]. FCM has been actively used in tree classification [4_4], processing large dataset [4_21, 4_16], clustering microarray [4_11], feature discrimination [4_15], image segmentation [4_52, 4_51] and kernel based clustering on image segmentation [4_6, 4_48]. FCM has been a popular research method for years. Some improvements for FCM have been put forward recently, such as single pass clustering [4_25], combining with support vector machine technique [4_50], using norm distance [4_20], hybrid with LP norms [4_37], speedup of the process in FCM was discussed in [4_23,4_41]. Zhang et al. [4_53] introduced a new kernel-induced distance measure for the original data space into the objective function of FCM (KFCM) to replace the conventional measures. Ref. [4_28] proposed a robust FCM to improve the reliability. Non linear/linear component analysis was done in [4_38, 4_40, 4_45].

Many researchers have proposed various improved FCM algorithms. Ahmed et al. [4_1] proposed FCM_S, which modified the objective function of FCM by introducing the spatial neighbourhood term. One drawback of FCM_S is that the spatial neighbourhood term is computed in each iteration step, which is very time-consuming. To reduce the computational complexity of FCM_S, Chen and Zhang [4_7] proposed two variants, FCM_S1 and FCM_S2, which replace the neighbourhood term of FCM_S by introducing the extra mean-filtered image and median-filtered image, respectively. The mean-filtered image and median-filtered image can be computed in advance, so the computational costs can be reduced. To speed up the image segmentation process, Szilagyi et al. [4_43] proposed the enhanced FCM (EnFCM), which form a linearly-weighted sum image from both the local neighbourhood average gray level of each pixel and original image, and then clustering is performed on the basis of the gray level histogram of summed image. Thus, the computational time of EnFCM is very small. Cai et al. [4_5] proposed the fast generalized FCM (FGFCM) algorithm. This method introduces a local similarity measure that combines both spatial and gray level information to form a non-linearly weighted sum image. Clustering is performed on the basis of the gray level histogram of the summed image. Thus, its computational time, similar to EnFCM, is also very small. However, these algorithms do not directly apply on the original image. They need some new parameters to control the trade-off between robustness to noise and effectiveness of preserving the details. The selection of these parameters is not an easy task, and has to be made by experience or by using the trial-and-error method. [4_17] proposed a variant of FLICM algorithm (RFLICM), which adopts the local coefficient of variation to replace the spatial distance as a local similarity measure. More references could be read at [4_9, 4_22, 4_47].

The goal of FCM is trying to find *C* clusters for *N* objects to minimize an objective function. Mostly used objective function is

$$y = \sum_{i=1}^{C} \sum_{n=1}^{N} \mu_{in}^{m} \left\| x_n - v_i \right\|^2, m \in [1, \infty) \tag{1}$$

Where *m* is the fuzzy exponent, mostly chosen value of 2, x_n $(n=1..N)$ is the n^{th} object and V_i $(i=1..C)$ is the center of the i^{th} cluster, u_{in} is the membership value for n^{th} sample in i^{th} cluster, which has value between *0* to *1*. All of u_{in} $(i=1..C,n=1..N)$ form a coefficient/partition matrix ***U***. A constraint is applied onto ***U***, which is $\sum_{i=1}^{C} \mu_{in} = 1, \forall n = 1,..,N$. This implies that the sum of the membership value for each sample on any one of the *C* clusters must be equal to 1. The sample is uniformly spread among those clusters. If the algorithm is working on the dataset containing *N* data points as $(\underline{x_1},y_1),(\underline{x_2},y_2),\ldots,(\underline{x_N},y_N)$. After we get the cluster, we can summarize the rules from those clusters. For each cluster, we can define the activation function for a data point $(\underline{x_n},y_n)$ on the clusters,

$$A_i(\underline{x}) = 1 / \sum_{j=1}^{C} \left(\frac{\left\| \underline{x_n} - \underline{v_i} \right\|}{\left\| \underline{x_n} - \underline{v_j} \right\|} \right)^{\frac{2}{m-1}}, i = 1,..,C, m > 1,$$

(2)

where $n = 1..N$ and $v_i, i = 1..C$ is the prototype for i^{th} cluster, x_n is the input of the data point.

To get the problem suitable for FCM, assume we need to get C clusters among N data points, so we will need weights for each data point in each cluster, thus we need totally $C \times N$ weights. For the C clusters, the weights for each data point should fulfill the constraint $\sum_{i=1}^{C} \mu_{in} = 1, \forall n = 1,.., N$.

The detail FCM algorithm flow is

Step 1: Get the N rules ready and related data

Step 2: Initializing the population and randomly select the weights for each data points on each cluster

Step 3: Standardize the input attribute weight among C clusters

$\mu_{ij} = \frac{\mu_{ij}}{\sum_{k=1}^{C} \mu_{ik}}, \forall i = 1,.., N, j = 1,.., C$, where μ_{ij} is the weight value for the i^th data point in *j^th* cluster. This step is trying to make sure the constraint $\sum_{i=1}^{C} \mu_{in} = 1, n = 1,.., N$ is satisfied.

Step 4: Calculate the new weights at stage *s+1* based on the old weights at stage *s*

$$\mu^{s+1}{}_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\| \mu^{s}{}_{ki} - v_i \|}{\| \mu^{s}{}_{kj} - v_k \|} \right)^{\frac{2}{m-1}}}, \forall i = 1,.., N, j = 1,.., C$$

Step 5: Calculate the new center of each cluster

$$v^{s+1}{}_{i} = \frac{\sum_{k=1}^{N} \left(\mu^{s+1}{}_{ki} \right)^{m} \times \underline{x}}{\sum_{k=1}^{N} \left(\mu^{s+1}{}_{ki} \right)^{m}}, i = 1,.., C$$

Step 6: if $\left\| \mu^{s+1} - \mu^{s} \right\| \le \varepsilon, .0 < \varepsilon < 1$, then exit, else return to step 3.

For FCM, one important parameter to be determined is the fuzzy factor, *m*. It can be any positive value bigger than *1*. Generally, *m* value is set to *2* in most application of FCM.

After FCM finishes, we get *C* prototypes on the *C* clusters. By optimizing the linear functions parameters with the performance of mean square error of the function output and the actual output of the data points.

To get the problem suitable for FCM, assume each input attribute *ATTR_{in}* (*i=1.A,n=1..N*) is one data sample. Thus, there are $A \times N$ samples all in all. We are trying to get *C* clusters for each rule, so we will need $C \times N$ clusters among the *N* rules. The *C* clusters inside a rule will have to fulfill the constraint $\sum_{i=1}^{C} \mu_{in} = 1, \forall n = 1,..,A$. To gain optimal parameters for the functions f_i

Let $a_i = \begin{bmatrix} a_{i0} \\ a_{i1} \\ a_{i2} \\ \cdots \\ a_{in} \end{bmatrix}$,output for linear consequents is

$$\hat{y}_k = \sum_{i=1}^{c} z_{ik}^T a_i, z_{ik} = [A_i(\underline{x_k}) \quad A_i(\underline{x_k}) x_k^T]^T = [z_{1k}^T \quad z_{1k}^T \quad \cdots \quad z_{ck}^T] \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_c \end{bmatrix},$$

$$y = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \cdots \\ \hat{y}_N \end{bmatrix} \text{ and } Z = \begin{bmatrix} z_{11}^T & z_{21}^T & \dots & z_{c1}^T \\ z_{12}^T & z_{21}^T & \dots & z_{c2}^T \\ \dots & \dots & \dots & \dots \\ z_{1N}^T & z_{2N}^T & \dots & z_{cN}^T \end{bmatrix},$$

$$y=\begin{bmatrix}\hat{y}_1\\ \hat{y}_2\\ \cdots\\ \hat{y}_N\end{bmatrix}=\begin{bmatrix}Z_{11} & Z_{12} & \cdots & Z_{1c}\\ Z_{21} & Z_{22} & \cdots & Z_{2c}\\ & \cdots & & \\ Z_{N1} & Z_{N2} & \cdots & Z_{Nc}\end{bmatrix}\begin{bmatrix}a_1\\ a_2\\ \cdots\\ a_c\end{bmatrix}, Z_{ij}=A_j(\underline{x_i})[1 \quad x_{i1} \quad \cdots \quad x_{in}]=\left[A_j(\underline{x_i})A_j(\underline{x_i})x_{i1} \quad \cdots \quad A_j(\underline{x_i})x_{in} \qquad \right]$$

$$y=\begin{bmatrix}A_1(\underline{x_1})A_1(\underline{x_1})x_{11} & \cdots & A_1(\underline{x_1})x_{1n} & A_2(\underline{x_1})A_2(\underline{x_1})x_{11} & \cdots & A_2(\underline{x_1})x_{1n} & \cdots & A_c(\underline{x_1})A_c(\underline{x_1})x_{11} & \cdots & A_c(\underline{x_1})x_{1n}\\ A_1(\underline{x_2})A_1(\underline{x_2})x_{21} & \cdots & A_1(\underline{x_2})x_{2n} & A_2(\underline{x_2})A_2(\underline{x_2})x_{21} & \cdots & A_2(\underline{x_2})x_{2n} & \cdots & A_c(\underline{x_2})A_c(\underline{x_2})x_{21} & \cdots & A_c(\underline{x_2})x_{2n}\\ & & & & \cdots & & & & & \\ A_1(\underline{x_N})A_1(\underline{x_N})x_{N1} & \cdots & A_1(\underline{x_N})x_{Nn} & A_2(\underline{x_N})A_2(\underline{x_N})x_{N1} & \cdots & A_2(\underline{x_N})x_{Nn} & \cdots & A_c(\underline{x_N})A_c(\underline{x_N})x_{N1} & \cdots & A_c(\underline{x_N})x_{Nn}\end{bmatrix}\begin{bmatrix}a_{10}\\ a_{11}\\ \cdots\\ a_{1n}\\ a_{20}\\ \cdots\\ a_{2n}\\ \cdots\\ a_{c0}\\ \cdots\\ a_{cn}\end{bmatrix}$$

then $y=Za$. Thus the solution is $a_{opt}=Z^{\theta}y, Z^{\theta}=(Z^TZ^{\theta})^{-1}Z^T$.

4.3 Complexity reduction in FCM

Suppose the rule-based system was got from a former algorithm, like *FNN*. There are total *N* rules and each rule has *A* input attributes. The i_{th} rules could be represented as

R_i: If X_{i1} *is* $Attr_1$ *AND* X_{i2} *is* $Attr_2$ *AND … AND* X_{iA} *is* $Attr_A$ then $y=fi(\underline{X},\underline{a_i})$

The goal is to reduce the number of input attributes in those *N* rules. When using clustering approach, we are trying to cluster the *A* attributes to *C* clusters. In this case, the i_{th} rule will be represented as

R_i: If X_{i1} *is* A_1 *AND* X_{i2} *is* A_2 *AND … AND* X_{iA} *is* A_C then $y=fi(\underline{X},\underline{a_i})$

Where A_i $(i=1..C)$ is a fuzzy cluster constructed in the A-dimensional input space and f_i is a local function.

$a_i, i=1..C$ is the parameter for the i^{th} function.

If we consider the linear form of the functions, we can represent the ith function as

$$y=f_i(\underline{x},\underline{a_i})=a_{i0}+a_{i1}x_1+a_{i2}x_2+\ldots+a_{iN}x_N$$

When we determine the parameters $a_{i0}, a_{i1}, ... a_{iN}$, the function can be used to apply on each data point to calculate the output and measure the performance by comparing with the actual output.

Our next step is trying to reduce the input space of the rules. For example, for ith rule, if we reduce the input space from *N* to n_i, where $n_i < N$, we map the original input space to the reduced input space, $\aleph \to \aleph_i$. The ith rule will be changed to

If $\underline{x}|_{\aleph_i}$ is Ai then $y = f_i(\underline{x}|_{\aleph_i}, \underline{a_i})$

The activation function will be computed as

$$A_i(\underline{x}|_{\aleph_i}) = \frac{1}{\sum_{j=1}^{C} \left(\frac{\left\| \underline{x} - \underline{v_i} \right\|_{\aleph_i}}{\left\| \underline{x} - \underline{v_j} \right\|_{\aleph_i}} \right)^{\frac{2}{m-1}}}, i = 1,..,C, m > 1$$

After reduction of the input space for each rule, we will get $n_1 + n_2 + ... + n_C = \kappa \bullet N, \kappa < 1, n_i \le N, i = 1..C$, κ is the space reduction coefficient. After we defined the space reduction coefficient, we can use GA to determine the existence of the input attributes in each rules to gain the result as good as possible comparing with the original rule sets.

4.4 Discussion

Clustering methods have been widely used in several of research areas, such as classification, pattern recognition, and data regression and proven to be powerful. In our research, we make use of traditional iteration based FCM training algorithm to find the optimal clusters for given training data objects. With the found clusters, we managed to figure out the linear estimation function to predict the output of the input variables. After that, we apply the rule reduction technique onto the FCM to get more concise fuzzy rule sets to improve the interpretability of the rule based system.

Reference

[4_1] M. Ahmed, S. Yamany, N. Mohamed, A. Farag, and T. Moriarty, “A modified fuzzy C-means algorithm for bias field estimation and segmentation of MRI data,” IEEE Trans. Med. Imag., vol. 21, no. 3, pp. 193–199, 2002.

[4_2] G. Berks, D. Graf, etc. "Fuzzy Clustering - A Versatile Mean To Explore Medical Database," ESIT 2000, 14-15, Aachen, German, 2000.

[4_3] J.C. Bezdek, "Pattern Recognition with Fuzzy Objective Function Algoritms", Plenum Press, New York, 1981.

[4_4] W. Bo and R. Nevatia, “Cluster boosted tree classifier for multi-view, multi-pose object detection,” in Proc. Int. Conf. Comput. Vision, pp. 1–8, Oct. 2007.

[4_5] W. Cai, S. Chen, and D. Zhang, “Fast and robust fuzzy C-means clustering algorithms incorporating local information for image segmentation,” Pattern Recognit., vol. 40, no. 3, pp. 825–838, 2007.

[4_6] L. Chen, C. Chen, and M. Lu, “A multiple-kernel fuzzy C-means algorithm for image segmentation,” IEEE Trans. Syst., Man, Cybern., B, Cybern., vol. 41, no. 5, pp. 1263–1274, Oct. 2011.

[4_7] S. Chen and D. Zhang, “Robust image segmentation using FCM with spatial constraints based on new kernel-induced distance measure,” IEEE Trans. Syst., Man, Cybern., B, Cybern., vol. 34, no. 4, pp. 1907–1916, 2004.

[4_8] Y. Chen, “Clustering parallel data streams,” in Data Mining and Knowledge Discovery in Real Life Applications, J. Ponce and A. Karahoca, Eds. Vienna, Austria: I-Tech, 2009.

[4_9] N. Cristianini and J. S. Taylor, "An Introduction to SVM’s and Other Kernel-Based Learning Methods." Cambridge, U.K.: Cambridge Univ. Press, 2000.

[4_10] S. Das, A. Abraham, and A. Konar, “Automatic clustering using an improved differential evolution algorithm,” IEEE Trans. Syst., Man, Cybern. A, Syst., Humans, vol. 38, no. 1, pp. 218–237, Jan. 2008.

[4_11] D. Dembele and P. Kastner, “Fuzzy c-means method for clustering microarray data,” bioinformatics, vol. 19, pp. 973–980, 2003.

[4_12] J.C. Dunn, "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters", Journal of Cybernetics 3: 32-57, 1973.

[4_13] B. S. Everitt, "Cluster Analysis," 3rd ed. New York: Halsted, 1993.

[4_14] J. Fan, A. Elmagarmid, X. Zhu, W. Aref, and L. Wu, “Clusterview: Hierarchical video shot classification, indexing and accessing,” IEEE Trans. Multimedia, vol. 16, no. 1, pp. 70–86, Feb. 2004.

[4_15] H. Frigui, “Simultaneous clustering and feature discrimination with applications,” in Advances in Fuzzy Clustering and Feature Discrimination With Applications. New York: Wiley, 2007, pp. 285–312.

[4_16] S. Guha, R. Rastogi, and K. Shim, “CURE: An efficient clustering algorithm for large databases,” Inf. Syst., vol. 26, no. 1, pp. 35–58, 2001.

[4_17] M. Gong, Z. Zhou, and J. Ma, “Change detection in synthetic aperture radar images based on image fusion and fuzzy clustering,” IEEE Trans. Image Process., vol. 21, no. 4, pp. 2141–2151, 2012.

[4_18] L. O. Hall and B. Ozyurt, "Scaling genetically guided fuzzy clustering." Proceedings of the 3rd International Symposium on Uncertainty Modeling and Analysis, 238-332, 1995.

[4_19] L. O. Hall, B. Ozyurt and J.C. Bezdek, "Clustering with genetically optimized approach." Proceedings of the IEEE Transactions on Evolutionary Computation, 3(2), 103-112, 1999.

[4_20] R. Hathaway, J. C. Bezdek, and Y. Hu, “Generalized fuzzy C-means clustering strategies using norm distances,” IEEE Trans. Fuzzy Syst., vol. 8, no. 5, pp. 576–581, Oct. 2000.

[4_21] R. Hathaway and J. Bezdek, “Extending fuzzy and probabilistic clustering to very large data sets,” Comput. Statist. Data Anal., vol. 51, pp. 215–234, 2006.

[4_22] R. Hathaway, J. C. Bezdek, and Y. Hu, “Generalized fuzzy C-means clustering strategies using norm distances,” IEEE Trans. Fuzzy Syst., vol. 8, no. 5, pp. 576–581, Oct. 2000.

[4_23] T. C. Havens, R. Chitta, A. K. Jain, and J. Rong, “Speedup of fuzzy and possibilistic kernel C-means for large-scale clustering,” in Proc. IEEE Int. Conf. Fuzzy Syst., pp. 463–470, Jun. 2011.

[4_24] K. Honda,A.Notsu, and H. Ichihashi, “Fuzzy PCA-guided robust k-means clustering,” IEEE Trans. Fuzzy Syst., vol. 18, no. 1, pp. 67–79, Feb. 2010.

[4_25] P. Hore, L. Hall, and D. Goldgof, “Single pass fuzzy c means,” in Proc. IEEE Int. Conf. Fuzzy Syst., London, U.K., pp. 1–7, 2007.

[4_26] A. K. Jain and R. C. Dubes, "Algorithms for Clustering Data." Englewood Cliffs, NJ: Prentice-Hall, 1988.

[4_27] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: A review,” ACM Comput. Surv., vol. 31, no. 3, pp. 264–323, Sep. 1999.

[4_28] S. Krinidis and V. Chatzis, “A robust fuzzy local information C-means clustering algorithm,” IEEE Trans. Image Process., vol. 19, no. 5, pp.1328–1337, May 2010.

[4_29] C.H. Li, B.C. Kuo, and C.T. Lin, “Lda-based clustering algorithm and its application to an unsupervised feature extraction,” IEEE Trans. Fuzzy Syst., vol. 19, no. 1, pp. 152–163, Feb. 2011.

[4_30] T. Liu, K. Ting, and Z.H. Zhou, “On detecting clustered anomalies using sciforest,” in Proc. Eur. Conf. Mach. Learn. Principles Pract. Knowl. Discov. Databases, pp. 274–290, 2010.

[4_31] U. Maulik and S. Bandyopadhyay, “Fuzzy partitioning using a real-coded variable-length genetic algorithm for pixel classification,” IEEE Trans. Geosci. Remote Sens., vol. 41, no. 5, pp. 1075–1081, May 2003.

[4_32] U. Maulik and I. Saha, "Automatic Fuzzy Clustering Using Modified Differential Evolution for Image Classification," IEEE transactions on geoscience and remote sensing, VOL. 48, NO. 9, pp:3503-3510, 2010

[4_33] B. Mirkin, "Mathematical Classification and Clustering," Springer, 1996.

[4_34] K. R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, “An introduction to kernel-based learning algorithms,” IEEE Trans. Neural Netw., vol. 12, no. 2, pp. 181–202, Mar. 2001.

[4_35] A.T. Papagiannakis, M. Bracher and N.C. Jackson. "Utilizing Clustering Techniques in Estimating Traffic Data Input for Pavement Design," J. Transp. Engrg. 132, 872, 2006.

[4_36] W. Pedrycz, V. Loia, and S. Senatore, “Fuzzy clustering with viewpoints,” IEEE Trans. Fuzzy Syst., vol. 18, no. 2, pp. 274–284, Apr. 2010.

[4_37] T. Przybyla, J. Jezewski, K. Horoba, and D. Roj, “Hybrid fuzzy clustering using LP norms,” in Proc. 3rd Int. Conf. Intell. Inf. Database Syst., pp. 187–196, 2011.

[4_38] V. Roth and V. Steinhage, “Nonlinear discriminant analysis using kernel functions,” in Advances in Neural Information Processing Systems 12, S. A Solla, T. K. Leen, and K.-R. Muller, Eds. Cambridge, MA: MIT Press, pp. 568–574, 2000.

[4_39] T. Sayed and W. Abdelwahab. "Comparison of Fuzzy and Neural Classifier for Road Accident Analysis," J. Comp. in Civ. Engrg, 12-42, 1998.

[4_40] B. Scholkopf, A. J. Smola, and K. R. Muller, “Nonlinear component analysis as a kernel eigenvalue problem,” Neural Comput., vol. 10, no. 5, pp. 1299–1319, 1998.

[4_41] V. Schwammle and O. Jensen, “A simple and fast method to determine the parameters for fuzzy c-means cluster analysis,” Bioinformatics, vol. 26, no. 22, pp. 2841–2848, 2010.

[4_42] P.H.A. Sneath and R.R. Sokal, "Numerical Taxonomy," San Francisco: W.H. Freeman, 1973.

[4_43] L. Szilagyi, Z. Benyo, S. Szilagyii, and H. Adam, “MR brain image segmentation using an enhanced fuzzy C-means algorithm,” in Proc. 25th Annu. Int. Conf. IEEE EMBS, pp. 17–21, 2003.

[4_44] P.-N. Tan, M. Steinbach, and V. Kumar, "Introduction to Data Mining." New York: Addison-Wesley, 2005.

[4_45] D. Tsai and C. Lin, “Fuzzy C-means based clustering for linearly and nonlinearly separable data,” Pattern Recognit., vol. 44, no. 8, pp. 1750–1760, 2011.

[4_46] T. Tu and Y. Chen, “Stream data clustering based on grid density and attraction,” ACM Trans. Knowl. Discov. Data, vol. 1, no. 1, 2008.

[4_47] K. Wu and M. Yang, “Alternative C-means clustering algorithms,” Pattern Recognit., vol. 35, no. 10, pp. 2267–2278, 2002.

[4_48] Z. Wu, W. Xie, and J. Yu, “Fuzzy c-means clustering algorithm based on kernel method,” in Proc. Int. Conf. Comput. Intell. Multimedia Appl., pp. 49–54, 2003.

[4_49] X. L. Xie and G. Beni, “A validity measure for fuzzy clustering,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 13, no. 8, pp. 841–847, Aug. 1991.

[4_50] X. Yang and G. Zhang, “A kernel fuzzy C-means clustering-based fuzzy support vector machine algorithm for classification problems with outliers or noises,” IEEE Trans. Fuzzy Syst., vol. 19, no. 1, pp. 105–115, 2011.

[4_51] X. Yin, S. Chen, E. Hu, and D. Zhang, “Semi-supervised clustering with metric learning: An adaptive kernel method,” Pattern Recognit., vol. 43, no. 4, pp. 1320–1333, 2010.

[4_52] L. Zhu, F. Chung, and S. Wang, “Generalized fuzzy C-means clustering algorithms with improved fuzzy partitions,” IEEE Trans. Syst., Man, Cybern., B, Cybern., vol. 39, no. 3, pp. 578–591, 2009.

[4_53] D. Zhang and S. Chen, “Kernel-based fuzzy clustering incorporating spatial constraints for image segmentation,” in Proc. IEEE 2nd Annu. Int. Conf. Mach. Learn. Cybern., pp. 2189–2192, 2003.

Chapter 5 Wireless Sensor Network and Node Deployment

Wireless sensor networks (WSNs) consist of a large number of inexpensive sensor nodes and provide a novel way to improve the border surveillance, aid disaster recovery, enable precision agriculture, real-time environmental surveying and so on. A good sensor node deployment algorithm is critical to enlarge the whole WSN's life time, so our focus of applying fuzzy rule-based system in controlling is put on the deployment of the sensor nodes.

5.1 Overview and Critical Issues

A wireless sensor network (WSN) consists of a number of sensor nodes working together to monitor a specific area and provides raw or partially processed data to a remote data processing center. The sensor nodes have sensing ability and limited computing ability, power supply and communicating ability. Each sensor node at least has computing unit, sensing unit, power unit and communication unit, so the sensor node could wirelessly transfer detected information, sometimes relays the information from its neighbouring nodes to a central data station.

With the merit of low cost and easy deployment of a large amount of nodes, the drawbacks are also obvious. The power supply is a main concern. The large number of nodes makes it not practicable to discover the power checking and replacement of the energy for failure nodes. Most of the research takes how to enlarge the power life into consideration. A good application of WSN should be able to work for a long period and the power life for the nodes in the network becomes a major measure [5_43]. Another important aspect is the constrained computing ability. On the cost of the node and the power constraint, the memory size and the processor’s ability are relatively poor. Large-scale calculation, complex algorithms and large data will not be suitable for WSN nodes. Most mature widely used algorithms in known application areas are not applicable in WSNs. This also widens the research topics in WSNs. Target tracking is a good example of finding new algorithms for the well settled applications [5_30]. Fault tolerant in data transmission is one hotter topic [5_28]. The nodes use wireless communication method to transform data. The radio signal sent by the nodes has the limitation of the transmission distance, which constrains the data will have to be sent through some relay nodes before they reach the destination [5_44, 5_45].

5.1.1 Energy consumption

Mostly, the WSN must be able to survive for a long period. The maintenance and the repair to the sensors in the WSN are impractical to be done, so the ability to resist the bad environment and work with limited energy become essential. The consumption of the provided energy largely controls the lifetime for an individual sensor. So the sensor will be in the status of sleeping when there are no needs for collecting data, computing or communicating. In order to balance the energy consumption among the nodes inside the network, we need to select dynamic routes in transferring data. This might need some more communication time, which will use extra energy. The trade off between the goals of maximize the single node's life and the whole network working period. The research on how to save the energy gets main focus in WSN. There are several ways to limit the energy consumption, such as efficiency in the communication protocol [5_4, 5_19, 5_47], computing unit OS optimization [5_9], and data pre-processing for saving communication time [5_48].

5.1.2 Topology recognition

In some use cases, the WSN is used to survey the area which is unreachable for human being. The topology of a WSN could be predefined, but mostly it is unknown at the beginning and the WSN has to figure out the topology by itself. On the other hand, the sensors are usually used in harsh environment, so they are prone to failures. The network topology of a WSN is supposed to change from time to time, so the WSN should have the ability to self-organize the network. The topic on recognizing the topology of WSN is gaining much attention [5_37, 5_46, 5_52, 5_55, 5_59].

5.1.3 Deployment

There are largely two types of node deployment, stationary and dynamic one. In a network with stationary deployment [5_8], the nodes in the WSN do not have the ability to move around. The stationary deployment takes advantage of the cheap sensing nodes, unwired node connection, known region of interest. For this type of WSN, the WSN nodes will be usually deployed automatically, like air drop. The random deployment will have trouble to preserve good observation over the ROI. The dynamic deployment supposed the WSN has all or some sensor nodes having the mobility. Nodes with mobility are considered to improve the overall performance of network. For dynamic deployment, it could be classified into two types considering the containing nodes, fully mobile WSN deployment and partially mobile WSN deployment [5_15, 5_60, 5_61]. In fully mobile WSN, all nodes are equipped with moving device [5_25]. For partially movable WSN, a portion of the nodes are stationary while the rest are mobile [5_15]. Based on the ROI type, dynamic deployment could be divided into two types, known area deployment and unknown area exploration [5_21, 5_22, 5_29].

5.1.4 Media access protocol

Half of the energy consumption of the WSN node is contributed by the communication as noted in [5_27]. In this paper, the medium access protocols for WSN are discussed and two new energy efficient medium access protocols, asynchronous MAC (A-MAC) protocol and asynchronous schedule-based MAC (ASMAC) protocol, are proposed. The fuzzy logic control was also applied to allocate the duration length. These two protocols need the WSN having the powerful data collection nodes (or normally cluster heads) to work properly. The simulation result was provided based on the local environment of a cluster. The comparison was made on A-MAC versus sensor medium access control (S-MAC) protocol [5_51], ASMAC versus S-MAC and traffic-adaptive medium access protocol (TRAMA [5_66]). The results show the two protocols outperform those protocols in extending the lifetime of the WSN.

5.1.5 Security

The authors in [5_65] took a deep look into the safety issues about the gateway or sink node. Security issues are gaining more and more interests. With the wide use of the WSN, the attacking risk raises [5_3, 5_7]. Ref. [5_58] did research in how to secure the data during the transmission by watching the nodes that are safe to use. The intrusion detection were adopted to mark those nodes might be under attack. The routing of the data then chooses those nodes were safe as the in-between hop nodes. In [5_43], the focus was laid on detecting the nodes that worked under anomaly mode, which would be the candidates of attacked nodes.

5.1.6 Target tracking

Target tracking is an important area in WSN application. The general target tracking with wireless sensor networks is focusing on the tracking accuracy and the error tolerance. For [5_1], the general concepts in target tracking in WSNs are thoroughly introduced. Sensors were classified and summarized their merits and shortcomings. In [5_41], the tradeoffs between the energy consumption and the calculation quality were discussed in depth. The sensors' ability was found related to the selection of different strategies best matched. [5_26] lowered the constraint on the sensors' ability to utmost. The balance needs to be found between the local processing and the communication needs. For [5_2, 5_6], the distributed tracking algorithms were investigated. For these distributed algorithms, the sensors are usually organized into tracking groups and the groups function as a unit in the tracking process. There are also some other researches, in which the groups were formed dynamically and a manager node was selected to coordinate the task, like in refs [5_6, 5_17, 5_30, 5_39, 5_50, 5_56, 5_57].

Tashtoush [5_54] applied fuzzy logic in predicating the next position of the moving target. Authors in [5_42] have developed a fuzzy target tracker to track a target. The fuzzy logic helped to eliminate the impact of the low signal-to-noise situation. [5_38] has introduced the application of fuzzy logic to solve the time-delay problem in tracking a moving target using passive acoustic sensors. A fuzzy tracker based on recursive estimation with multiple fuzzy models was developed in [5_35]. The paper employed the fuzzy rules to select a correct acceleration model. Subsequently, the target is estimated based on a selected model. Osman et al. [5_40] have proposed a fuzzy target tracker to track a single target in cluttered underwater environment. In [5_33, 5_34], a fuzzy interacting multiple model algorithm has been presented. The algorithm considers each Kalman filter to have only local validity, defined by the model conditioning input.

5.1.7 Other Issues

Ref. [5_16] laid focus on the cluster heads selection based on balancing the energy consumption on individual nodes, to improve the node lifetime and the network lifetime. Habitat study [5_5] is an important application of WSN. Some other research areas include the security issue existing in the communication [5_11, 5_12, 5_24] and localization [5_14, 5_23]. The application of the WSNs focuses the observation and the control of the physical world, such as outdoor monitoring, e.g. environmental and habitat monitoring [5_32], indoor monitoring and control applications [5_10] e.g. temperature control, security monitoring and reacting, and intelligent alarms. Habitat study is an important aspect among the applications of the wireless sensor networks. The wireless sensors are embedded in the monitoring entities' environment to gather and sensing the bio-physical/bio-chemical information. This kind of applications could be found in [5_53].

5.2 Performance Metrics for WSN Deployment

The performance of the deployment is measure with some metrics, such as coverage, uniformity, deployment time, travel distance, remaining power etc.

- ***Coverage***

If the sensing range for the node is R_s, the area that the node could monitor is usually a circle in 2-dimensional field or a ball in 3-dimensional field. To simplify the complexity of the problem, we will just talk about the 2-dimensional ROIs. Then the covered field for a single sensor $N(x, y)$ is the circle area πR_s^2 around point (x, y). In the deployment research, we are mainly interested in the coverage, the percentage of monitored area among the entire ROI. If the WSN is working in an ROI of rectangle with width X and height Y, then the coverage (C) for the WSN with n nodes will be calculated with the formula $C = \frac{\bigcup_{i=1}^{n} C_i}{X \times Y}$, where C_i is the covered area for i^{th} sensor node. There is also the coverage of the communication range (R_c) for WSN. We are not considering the communication coverage issue in this metric. It will be discussed as the orphan ratio instead. From the points of sensing coverage, the relation of two sensor nodes could be one of the 3 types.

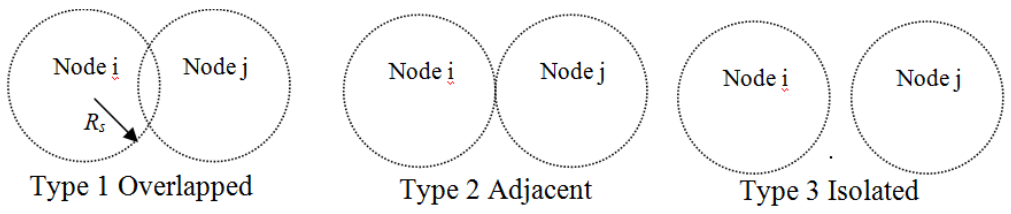


Figure 5.1 Relationship between two sensor nodes

The distance between them is less than $2 \times R_s$ for overlapped sensor nodes, and is equal to $2 \times R_s$ for adjacent sensor nodes, and is greater than $2 \times R_s$ for isolated sensor nodes. In this paper, we require that the overlapped or adjacent nodes should be neighbours, and then R_c should be at least $2 \times R_s$.

- ***Travel distance***

The average travel distance for each node is related to the energy consumption for the movement during the redeployment. So, the expected travel distance is important for estimating remaining energy. The variance in travel distance is also important to find out the fairness of the deployment algorithm and for energy use. If the variance of travel distance is large, the variance of energy remaining also is large. The nodes that have less energy than other nodes exhaust their energy early. Early dead nodes result in a loss of coverage and the remaining nodes may require an increased transmission range or a longer routing path. Then the lifetime for the WSN is significantly shortened.

- ***Orphan ratio***

Since the orphans are useless in WSN. We need to eliminate the orphans during the deployment process. This is also an important metric for the quality of the WSN. With a higher density of the nodes in the WSN, the chance of having orphans becomes rare.

- ***Time***

The time spent for redeployment to reach high coverage, good uniformity and low orphan rate is also importance. This could also be called as converging speed. This metric may be used as one of the design constraints such that the applications should act properly within the desired time requirement.

5.3 Related work on deployment strategies

The deployment of sensors varies with different applications. Several applications require placing sensors at desired locations like data collection and security, where critical area, buildings and facilities are monitored by a network of sensors placed adequately. For such placement-friendly applications, plenty knowledge of the environment is assumed to be available before deployment is carried out. In other applications where prior knowledge of the environment cannot be gained, sensors may have to be randomly air-dropped and human intervention after deployment to recharge or replace node batteries may not be feasible. Mobile sensors are desirable in this situation because they can move around to readjust their positions for high quality communication and better coverage and surveillance.

However changing position is energy consuming for mobile sensor deployment. An efficient sensor redeployment scheme is a necessity to save energy resources and improve the quality of communications. Deployment methods using mobile nodes have been proposed to increase network coverage and to extend the network lifetime via arrangement of uniformly delivered node topologies from random node distributions [5_67]. Since mobility itself requires energy from its own limited energy source, a deployment scheme should be designed carefully to minimize energy consumption during deployment. At the same time, certain goals such as satisfactory coverage or an energy efficient node topology should be reached. Moreover, it is desirable for a distributed sensor network node to have fairly simple hardware architecture, which requires slight computing power and memory. Each node should have a simple and efficient algorithm for deployment, organization, and management of the WSN. Not only lessening average moving distance, but also reducing the difference of the remaining energy among sensor nodes is essential for a longer lifetime. Because of the dynamic and distributed nature of deployment, it is a challenging task to gain full coverage in the ROI and to use energy of each sensor in a fair fashion.

One much referenced deployment algorithm is distributed self-spreading algorithm (DSSA) [5_20]. The main idea of DSSA is to define a partial force for the movement of sensors during the deployment process. The force a node receives from a closer neighbor node is greater than that from a farther neighbor.

The bottle neck in the deployment research should be how to improve the deployment efficiency when the node density is not low. When the node density is low, all proposed methods are able to deploy the nodes in a good way to cover more area without much overlapping. But when the node density is high, the overlapped covered area becomes a big issue. The algorithms currently available are all produced less difference in the coverage when the nodes are crowd. The distance is another important criteria always being used in comparison. Also the comparison is done to other criteria improvement. The probability of outrage criteria was applied by Shu etc. [5_49] when comparing the algorithm performance with DSSA. Termination time [5_62] is used to compare the result with DSSA.

Garetto etc. [5_13] discussed the sensor networks containing the nodes equipped with sensing and acting devices, which is both sensor and actuator network. Authors in [5_36] tried to combine the virtual force and particle swarm optimization. The results shown in the paper compared the three methods, virtual force (VF) only, particle swarm optimization (PSO) only and virtual force-directed particle swarm optimization (VFPSO). An application-specific clustering protocol in nodes deployment, called low-energy adaptive clustering hierarchy (LEACH), was proposed in [5_18]. Two researchers [5_63, 5_64] also introduced a clustering method to deal with the node deployment, namely Hybrid Energy-Efficient Distributed clustering (HEED). Simulated annealing algorithm was adopted to improve the deployment performance. The algorithm was called Stochastic Deployment Routine (SDR) [5_36]. Loo and others considered a system consisting of some cooperating mobile nodes that move toward a set of prioritized destinations under sensing and communication constrains [5_31]. They show how individual agents know when cooperation between agents improves the performance and when they should suspend cooperation.

5.4 Summary

Wireless sensor network (WSN) is attracting more and more attention with the development of the wireless communication, electronics, and mobile computation. It is critically important to maximize the lifetime of the whole network. This includes how to minimize the energy consumption of the single sensor node and how to balance the lifetime among the sensor nodes. Since the nodes in wireless sensor network are error pruned, the lost of the information and the broken down of some nodes should be expected. The fault tolerant is also very important. For mobile wireless sensor network, a good deployment algorithm could maintain balanced energy consumption among sensor nodes and avoid trouble brought by node failure. Also, deployment of the sensor node becomes critical when the network topology needs to be update from time to time, or reliable of the sensor coverage is critical. In this dissertation, we proposed to construct the rules for directing sensor node deployment using FNN and to apply rule complexity reduction to gain concise accurate rule set.

References

[5_1] A. Arora, P. Dutta, S. Bapat, and e. al., "A line in the sand: a wireless sensor network for target detection, classification, and tracking." Journal of Computer Networks, 605-634, 2004.

[5_2] S. Balasubramanian, I. Elangovan, S.K. Jayaweera, and K.R. Namuduri, "Distributed and collaborative tracking for energy-constrained ad-hoc wireless sensor networks," Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE, pp. 1732- 1737, 2004.

[5_3] L. Bononi, and C. Tacconi, "Intrusion detection for secure clustering and routing in Mobile Multi-hop Wireless Networks," International Journal of Information Security, vol. 6, pp. 379-392, 2007.

[5_4] D. Braginsky, and D. Estrin, "Rumor routing algorthim for sensor networks " Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, pp. 22-31, 2002.

[5_5] A. Cerpa, J. Elson, and D. Estrin, "Habitat monitoring: Application driver for wireless communication technology, " ACM SIGCOMM Workshop on Data Communications in Latin American and the Caribbean, pp. 20-41, 2001.

[5_6] W.P. Chen, J.C. Hou, and L. Sha, "Dynamic clustering for acoustic target tracking in wireless sensor networks." Mobile Computing, IEEE Transactions on, pp: 258- 271, 2004.

[5_7] S.H. Chi, and T.H. Cho, "Fuzzy logic anomaly detection scheme for directed diffusion based sensor networks," Fuzzy Systems and Knowledge Discovery, Proceedings, Lecture Notes in Computer Science 4223, Springer-Verlag Berlin, pp. 725-734, 2006.

[5_8] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, and K.K. Saluja, "Sensor deployment strategy for target detection " Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications pp. 42-48, 2002.

[5_9] S. Coleri, M. Ergen, and T.J. Koo, "Lifetime analysis of a sensor network with hybrid automata modelling " Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications pp. 98-104, 2002.

[5_10] W.S. Conner, et al. "Experimental evaluation of synchronization and topology control for in-building sensor network applications." in Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications. San Diego, CA, USA. 2003.

[5_11] B. Deb, S. Bhatnagar, and B. Nath, "Information assurance in sensor networks " Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications, pp. 160-168, 2003.

[5_12] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. Dean, F. Mueller, and M. Sichitiu, "Analyzing and modeling encryption overhead for sensor network nodes " Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications, pp. 151-159, 2003.

[5_13] M. Garetto, M. Gribaudo, C.-F. Chiasserini, and E. Leonardi, "Sensor Deployment and Relocation: A Unified Scheme," Journal of Computer Science and Technology, vol. 23, no. 3, pp. 400-412, 2008.

[5_14] J.v. Greunen, and J. Rabaey, "Lightweight time synchronization for sensor networks " Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications, pp. 11-19, 2003.

[5_15] W. Guiling, C. Guohong, and T. La Porta, "Proxy-based sensor deployment for mobile sensor networks," Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on, pp. 493-502, 2004.

[5_16] N. Gupta, D. Riordan, and S. Sampalli, "Cluster-head election using fuzzy logic for wireless sensor networks," Communication Networks and Services Research Conference, Proceedings of the 3rd Annual pp. 255-260, 2007.

[5_17] Q.B. Hao, D.J. Guenther, and B.D. Burchett, "Human tracking with wireless distributed pyroelectric sensors." Sensors Journal, IEEE. 1683-1696, 2006.

[5_18] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," Wireless Communications, IEEE Transactions on, vol. 1, no. 4, pp. 660-670, 2002.

[5_19] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishman, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," Proceedings of HICSS'00, pp. 3005--3014, 2000.

[5_20] N. Heo, and P.K. Varshney, "A distributed self spreading algorithm for mobile wireless sensor networks," Wireless Communications and Networking, 2003. WCNC 2003. IEEE, pp. 1597-1602 vol.1593, 2003.

[5_21] A. Howard, M. Mataric, and G. Sukhatme, "Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem," Distributed Autonomous Robotic Systems (DARS'02), pp. 299-308, 2002.

[5_22] A. Howard, M.J. Mataric, and G.S. Sukhatme, "An incremental self-deployment algorithm for mobile sensor network," Autonomous Robots, vol. 13, no. 2, pp. 113-126, 2002.

[5_23] A.s. Hu, and S.D. Servetto, "Asymptotically optimal time synchronization in dense sensor networks " Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications, pp. 1-10, 2003.

[5_24] Q. Huang, J. Cukier, H. Kobayashi, B. Liu, and J. Zhang, "Fast authenticated key establishment protocols for self-organizing sensor networks " Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications, pp. 141-150, 2002.

[5_25] D.B. Jourdan, and O.L. de Weck, "Layout optimization for a wireless sensor network using a multi-objective genetic algorithm," Vehicular Technology Conference, 2004. VTC 2004-Spring. IEEE 59th, pp. 2466-2470 Vol.2465, 2004.

[5_26] W. Kim, K. Mechitov, J.-Y. Choi, and S. Ham, "On target tracking with binary proximity sensors," Proceedings of the 4th international symposium on Information processing in sensor networks, Los Angeles, California, 2005.

[5_27] K. Langendoen, and N. Reijers, "Distributed localization in wireless sensor networks: a quantitative comparison," Computer Networks: The International Journal of Computer and Telecommunications Networking table of contents, vol. 43, no. 4, pp. 499-518, 2003.

[5_28] Q. Liang, "Fault-tolerant and energy efficient wireless sensor networks: a cross-layer approach," MILCOM. IEEE, pp. 1862-1868, 2005.

[5_29] B. Liu, P. Brass, O. Dousse, P. Nain, and D. Towsley, "Mobility improves coverage of sensor networks," Book Mobility improves coverage of sensor networks, Series Mobility improves coverage of sensor networks, ed., Editor ed.^eds., ACM, pp. 300-308, 2005.

[5_30] J. Liu, P. Cheung, F. Zhao, and L. Guibas, "A dual-space approach to tracking and sensor management in wireless sensor networks " Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications pp. 98-104, 2001.

[5_31] L.H. Loo, E. Lin, M. Kam and P.K. Varshney, "Cooperative multi-agent constellation formation under sensing and communication constrains," in Cooperative Control and Optimization, pp. 143-170, Kluwer Academic Press, 2002.

[5_32] A. Mainwaring, et al. "Wireless sensor networks for habitat monitoring," in Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications . Atlanta, Georgia, USA 2002.

[5_33] S. McGinnity and G.W. Irwin. "fuzzy assisted kalman filtering for target tracking." in Control '96, UKACC International Conference on 1996.

[5_34] S. McGinnity and G.W. Irwin, "Fuzzy logic approach to manoeuvring target tracking." Radar, Sonar and Navigation, IEE Proceedings - 145(6): p. 337-341, 1998.

[5_35] C.G. Moore, C.J. Harris, and E. Rogers. "Utilising fuzzy models in the design of estimators and predictors: an agile target tracking example." in Fuzzy Systems, Second IEEE International Conference on. 1993.

[5_36] H. Mousavi, A. Nayyeri, N. Yazdani, and C. Lucas, "Energy conserving movement-assisted deployment of ad hoc sensor networks," Communications Letters, IEEE, vol. 10, no. 4, pp. 269-271, 2006.

[5_37] A. Nasipuri, and K. Li, "A directionality based location discovery scheme for wireless sensor networks " Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications pp. 105-111, 2001.

[5_38] G.W. Ng, "Application of fuzzy logic in multipassive acoustic tracking." Proceedings of SPIE. 3692: p. 389-397, 1999.

[5_39] S. Oh, S. Satry, and L. Schenato, "A Hierarchical Multiple-Target Tracking Algorithm for Sensor Networks, Robotics and Automation," ICRA 2005. Proceedings of the 2005 IEEE International Conference on, 2005, pp. 2197- 2202, 2005.

[5_40] H.M. Osman, M. Farooq, and T. Quach, "Fuzzy logic approach to data association." Proceedings of SPIE. 2755: p. 313-322, 1996.

[5_41] S. Pattem, S. Poduri, and B. Krishnamachari, "Energy-Quality Tradeoffs for Target Tracking in Wireless Information Processing in Sensor Networks: Second International Workshop," Palo Alto, CA, USA, pp. 32-46, 2003.

[5_42] T. Quach and M. Farooq, "Fuzzy-logic-based target tracking algorithm." Proceedings of SPIE. 3390: p. 476-487, 1998.

[5_43] S. Raje, and Q. Liang, "Time Synchronization in Network-Centric Sensor Networks," Sensor Networks IEEE, pp. 333-336, 2007.

[5_44] V. Rajendran, K. Obraczka, and J.J. Garcia-Luna-Aceves, "Energy-efficient, collision-free medium access control for wireless sensor networks," in Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (Sen-Sys '03), pp. 181–192, 2003.

[5_45] Q.C. Ren, and Q.L. Liang, "Energy-efficient medium access control protocols for wireless sensor networks," Eurasip Journal on Wireless Communications and Networking, 2006, pp. 5-15.

[5_46] A. Savvides, H. Park, and M.B. Srivastava, "The bits and flops of the n-hop multilateration primitive for node localization problems " Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications pp. 112-121, 2002.

[5_47] S.D. Servetto, and G. Barrenechea, "Constrained random walks on random graphs: routing algorithms for large scale wireless sensor networks," Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications pp. 12-21, 2002.

[5_48] K. Sha, "Modeling and extending lifetime of wireless sensor networks by considering data consistency," M.S., Wayne State University, United States -- Michigan, 2007.

[5_49] H. Shu, Q. Liang, and J. Gao, "Distributed Sensor Networks Deployment Using Fuzzy Logic Systems," International Journal of Wireless Information Networks, vol. 14, no. 3, pp. 163-173, 2007.

[5_50] L. Song, and D. Hatzinakos, "A Cross-Layer Architecture of Wireless Sensor Networks for Target Tracking. Networking," IEEE/ACM Transactions on 15 pp: 145-158, 2007.

[5_51] T. Srinivasan, R. Chandrasekar, and V. Vijaykumar, "A fuzzy, energy-efficient scheme for data centric multipath routing in wireless sensor networks," Book A fuzzy, energy-efficient scheme for data centric multipath routing in wireless sensor networks, Series A fuzzy, energy-efficient scheme for data centric multipath routing in wireless sensor networks, pp.10-21, 2006.

[5_52] J. Staddon, D. Balfanz, and G. Durfee, "Efficient tracing of failed nodes in sensor networks " Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications pp. 122-130, 2001.

[5_53] R. Szewczyk, J. Polastre, A. M. Mainwaring and D. E. Culler. "Lesson from a sensor network expedition." In EWSN, Berlin, Germany, 2004.

[5_54] Y. Tashtoush, "Applying fuzzy logic and reinforcement learning to track a mobile target using a wireless sensor network," in Huntsville, The University of Alabama Huntsville Alabama. 2006.

[5_55] S. Tilak, N.B. Abu-Ghazaleh, and W. Heinzelman, "Infrastructure tradeoffs for sensor networks " Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications pp. 49-58, 2002.

[5_56] Y.C. Tseng, S.P. Kuo, H.-W. Lee, and C.-F. Huang, "Location Tracking in a Wireless Sensor Network by Mobile Agents and Its Data Fusion Strategies," The Computer Journal 47 pp:448-460, 2004.

[5_57] Q. Wang, W.P. Chen, R. Zheng, K. Lee, and L. Sha, "Acoustic Target Tracking Using Tiny Wireless Sensor Devices Information Processing in Sensor Networks: Second International Workshop," Palo Alto, CA, USA, pp. 642-657, 2003.

[5_58] L. Wang, S. Chen, and X. Li, "A Multiple-Objective Fuzzy Decision Making Based Information-Aware Routing Protocol for Wireless Sensor Networks," Wireless Communications, Networking and Mobile Computing. WiCOM 2006.International Conference on, pp. 1-4, 2006.

[5_59] K. Whitehouse, and D. Culler, "Calibration as parameter estimation in sensor networks " Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications pp. 98-104, 2003.

[5_60] A.F.T. Winfield, "Distributed sensing and data collection via broken ad hoc wireless connected networks of mobile robots," in Distributed Autonomous Robotic System 4, eds. LE Parker, G Bekey & J. Barhen, Springer-Verlag, pp. 273-282, 2000.

[5_61] X. Wu, Y. Niu, L. Shu, J. Cho, Y. Lee, and S. Lee, "Relay Shift Based Self-deployment for Mobility Limited Sensor Networks," Ubiquitous intelligence and computing, pp. 556-564, 2006.

[5_62] X. WU, L. SHU, J. YANG, H. XU, J. CHO, and S. LEE, "Swarm Based Sensor Deployment Optimization in Ad Hoc Sensor Networks," Embedded software and systems : (Second international conference, ICESS 2005) (proceedings), pp. 533-541, 2005.

[5_63] O. Younis, and S. Fahmy, "HEED: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," Ieee Transactions on Mobile Computing, vol. 3, no. 4, pp. 366-379, 2004.

[5_64] O. Younis, and S. Fahmy, "Distributed clustering in ad-hoc sensor networks: a hybrid, energy-efficient approach," INFOCOM. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, pp. 629-640, 2004.

[5_65] W. Youssef, and M. Younis, "A cognitive scheme for gateway protection in wireless sensor network," Applied Intelligence, pp. 216-227, 2007.

[5_66] M. Yusuf, and T. Haider, "Energy-aware fuzzy routing for wireless sensor networks," Emerging Technologies, Proceedings of the IEEE Symposium on, pp. 63-69, 2005.

[5_67] Y. Zou, and K. Chakrabarty, "Sensor deployment and target localization based on virtual forces," Proc. Of the IEEE INFOCOM Conference, pp. 1293-1303, 2003.

Chapter 6 Construction and complexity reduction of fuzzy rule-based systems with Fuzzy Neural Networks

Fuzzy logic has been widely used in data modeling in recent years. In this chapter, using FNN to construct and simplify the rule-based system is presented. The FNN structure used in constructing the rule base system is described and the pruning method for FNN is applied to simplify the rules. In order to gain best performance on accuracy and simplicity of the rule based system, multi-objective approach is adopted. The fuzzy neural networks are constructed with the help of evolutionary intelligence technique. The complexity reduction is also done using evolutionary computation algorithms.

Our point of departure is a “standard” Takagi-Sugeno fuzzy model comprising of “c” rules in the form

-if **x** is A_i then y= $f_i(\mathbf{x}, \mathbf{a}_i)$

i=1,2, .., c where $\mathbf{x} \in R^n$. The design of the modes is well-reported in the literature and as usual consists of the two main steps, namely a construction of condition parts (through clustering of input data done in the input space) and applying linear regression to the formation of the linear local models f_i. The number of rules(c) is determined by monitoring the behaviour of the model on the training and testing data.

The performance of the model is expressed by the RMSE computed for the training set

$$RMSE_{training} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(\mathrm{FM}(\mathbf{x}_k) - \mathrm{target}_k)^2} \tag{1}$$

Where N denotes the number of data in the training set. In the same way, determined is the performance of the constructed model on the testing data (M data points)

$$RMSE_{testing} = \sqrt{\frac{1}{M}\sum_{k=1}^{M}(FM(\mathbf{x}_k) - target_k)^2} \tag{2}$$

6.1 Approaches presented in literature

The concept of interpretability of fuzzy rule-based models has been around for several decades and attracted a significant deal of attention. The transparency of fuzzy models is one of the outstanding and important features of fuzzy models. In contrast to the criterion of accuracy, whose quantification is relatively straightforward and easy to come up with performance indexes, transparency of fuzzy rules is more difficult to describe. What makes the fuzzy rule-based easier to interpret is still an open issue. It is quite subjective to assess and in one way or another invokes a factor of subjective judgment given that a human user is ultimately involved in the evaluation process. What become very much apparent are a multifaceted nature of the problem and a multitude of various approaches supported by various optimization technologies including evolutionary optimization. When it comes to the main factors worth considering when discussing a concept of interpretability, as denoted in Chapter 5, there have been quite a number of factors got involved in the reduction process.

Ref. [6_1] discussed about 4 types of feature evaluation algorithms used for feature selection or reduction and proposed a min-redundancy and max-relevance search strategy to improve the classifier performance. Experiments on 14 datasets from machine learning repository were shown to demonstrate the advantages of new way of feature selection. The results were not compared with any existing reference to verify if the algorithm here was better or worse than other algorithms.

The paper is focusing on considering the attributes in the dataset having monotonic relation with the decision, and trying to design the classification algorithm for this type of datasets. Although it is not realistic that the real datasets could satisfy this condition, but the paper treats the inputs having monotonic relation with decision as criteria, kind of key attributes and the rest as normal attributes. Ref. [6_2] introduced a way to do feature selection based on fuzzy-rough sets by maximizing the relevance and minimizing the redundancy of the selected features. The experiment results were evaluation in terms of classification error, class reparability and two dependency indexes, calculated based on some known datasets, like iris, wine, letter, isolet and satimg. There were quite a number of parameters to be chosen for the algorithm to perform well. This is a fuzzy rough set related feature selection technique. Ref. [6_3] reported the work in 1995 to construct and reduce the fuzzy rule based system. It showed the result for FAM bank data. In total, there are (5*7)*7 [inputs*output], in total 245 rules after error back-propagation training. After pruning the rules, 35 (all input combination) rules remained and the performance improved after pruning compared with using all 245 rules in classification. The paper put forward an algorithm to construct the fuzzy rule base using FNN with all possible combination of input/output fuzzy sets. Then select one OR neuron for each AND neuron to connect with, in order to cut off conflicting rules. Ref. [6_4] discussed about the attribute reduction in fuzzy decision systems using the generalized fuzzy evidence theory. Some new concepts were introduced into this area and gave the related theory and proof. This paper is mainly discussing about the theory, no actual experiments were done. In ref. [6_5], the focus is putting on balancing invert pendulum to evaluate the result for the research. The algorithm in the paper found 12 rules were good enough, while some referred papers need 81. The experiment showed that the 12 rules could work well under changed parameters for the balancing invert pendulum system. The paper uses GA to construct the rules by selecting all possible combination of input variable linguistic terms. This is somewhat like using FNN. The rule number reduction is done by setting maximum rule number and removing conflicting rules.

We also applied the FRBS into the real world control problem, the deployment of wireless sensors in the wireless sensor networks, which will be stated in Chapter 7.

6.2 Dataset description

In the study, we experiment with a number of publicly available datasets coming from eight datasets from UCI Machine learning repository and DELVE repository; their main characteristics are listed in Table 6.1.

Dataset	Number of input variables	Number of data	Origin of the data
Abalone	8	4,177	UCI Machine learning repository (http://archive.ics.uci.edu/ml/)
Auto MPG	7	392	UCI Machine learning repository
Boston Housing	13	506	UCI Machine learning repository
Computer Activity	21	8,192	DELVE repository (http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html)
Concrete strength	8	1,030	UCI Machine learning repository
Forest fires	12	517	UCI Machine learning repository
Red wine quality	11	1,599	UCI Machine learning repository
White wine quality	11	4,898	UCI Machine learning repository

Table 6.1 Main characteristics of datasets used in the experiments (number of data and dimensionality)

6.3 Construction of Fuzzy Neural Networks

We can generate the rule-based system based on the given problem from an algorithm, like *FNN*. There are total *N* rules and each rule has *A* input attributes. The i_{th} rules could be represented as

R_i: If X_{i1} *is* $Attr_1$ *AND* X_{i2} *is* $Attr_2$ *AND … AND* X_{iA} *is* $Attr_A$ then $y=fi(\underline{X},\underline{a_i})$

The FNN applied here is fuzzy AND-OR neural network. The structure of FNN is shown in figure 6.1.

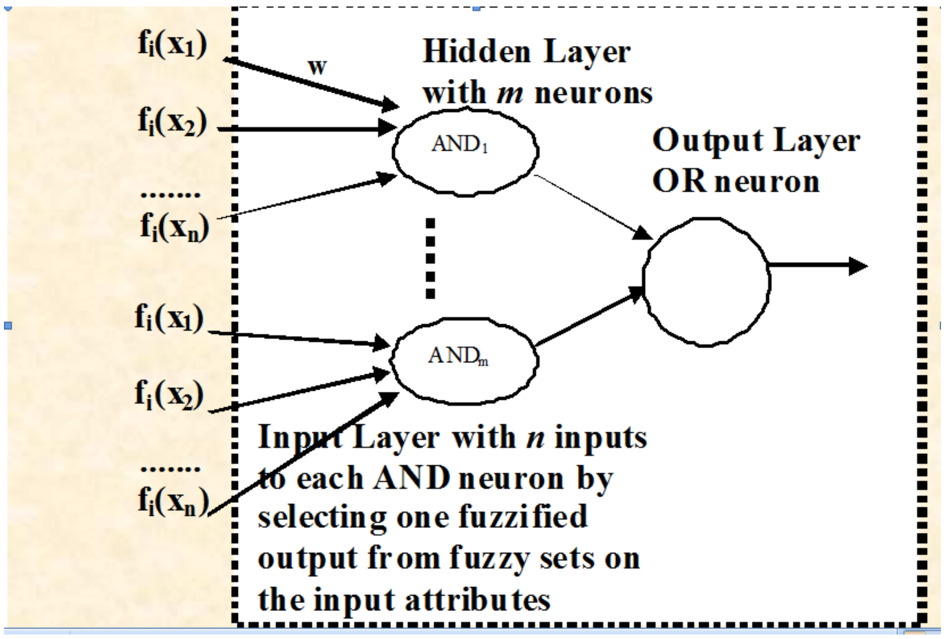
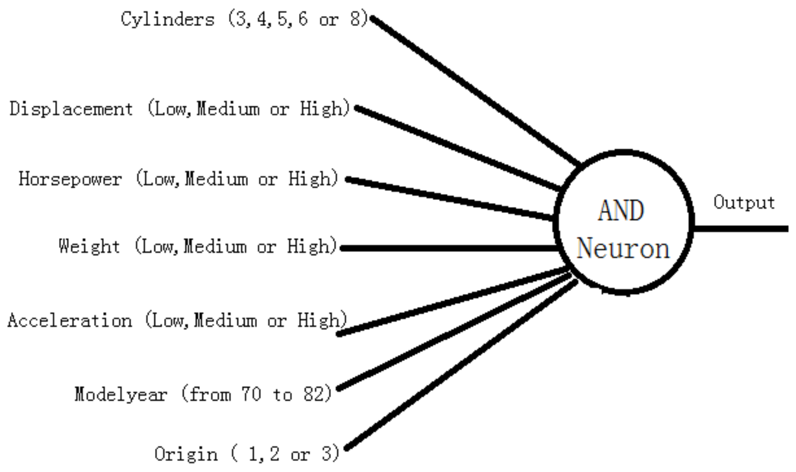

$f_i(x_1)$
$f_i(x_2)$
....
$f_i(x_n)$
w
Hidden Layer with *m* neurons
AND_1
Output Layer OR neuron
$f_i(x_1)$
$f_i(x_2)$
....
$f_i(x_n)$
AND_m
Input Layer with *n* inputs to each AND neuron by selecting one fuzzified output from fuzzy sets on the input attributes

Figure 6.1 FNN structure

This is a novel way to construct the FNN. Like normal FNN, it has input layer to accept all input to the model and hidden layer with predefined number, *m*, of AND neurons, and the output layer is the OR neuron to aggregate the output from AND neurons. The number *m* is not related to the input attributes in the dataset under process or to the number of fuzzy sets defined on the input attributes. The value of *m* can be chosen universally for all the experiments. In our research, we select 9 for the value of *m*. In this way, the difficulty and the calculation complexity in designing the model can be dropped significantly for large complex datasets. On the other hand, we need to maintain the diversity of the candidate models in order to find out the best fit model for a given dataset. Each AND neuron has *n* inputs. The number *n* equals to the number of input attributes in the raw inputs of the dataset. The input from the attribute is actually the fuzzy set membership value for continuous attributes or the 1-out-of-n value for discrete attributes. For each AND neuron, the dataset attributes will choose one of the fuzzy set to process the raw value and feed the fuzzy set calculation as the actual input value to this neuron. So to construct the FNN, we do not need only to tune the weights between inputs to AND neurons, AND neurons to OR neuron, but we also need to figure out the best attribute input fuzzy set combination to the AND neurons. This will enable the model to search for the best model with a comparatively simple FNN structure. To clearly describe the inputs to AND neuron, a sample input to the AND neuron for auto mpg dataset is displayed in figure 6.2.


Cylinders (3,4,5,6 or 8)
Displacement (Low,Medium or High)
Horsepower (Low,Medium or High)
Weight (Low,Medium or High)
Acceleration (Low,Medium or High)
Modelyear (from 70 to 82)
Origin (1,2 or 3)
AND Neuron
Output
Figure 6.2 Sample of the inputs to the AND neuron

The FNN will be trained with GA. The chromosome contains two sections, as shown in figure 6.3.

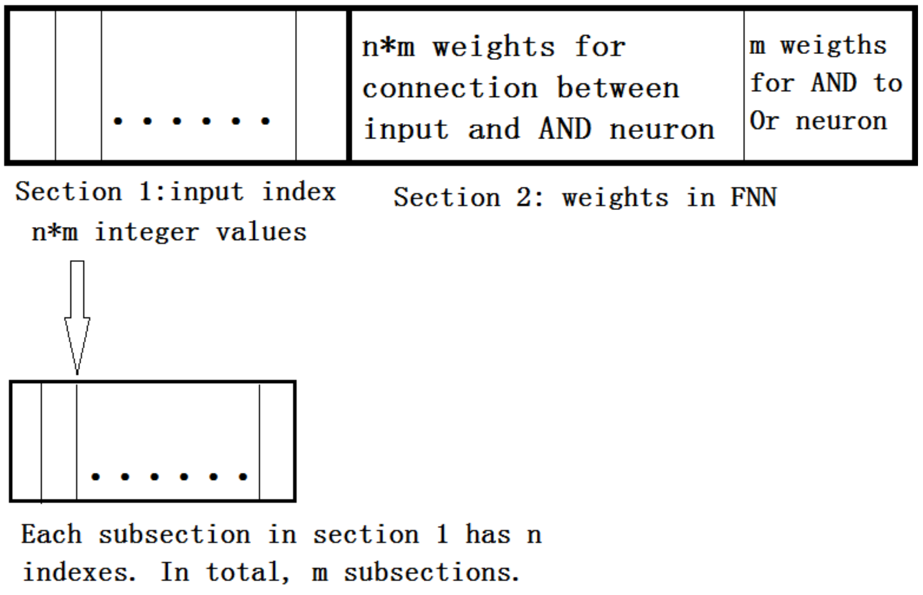

n*m weights for connection between input and AND neuron
m weigths for AND to Or neuron
Section 1:input index n*m integer values
Section 2: weights in FNN
Each subsection in section 1 has n indexes. In total, m subsections.

Figure 6.3 Chromosome structure

One section is allocated to store the inputs to AND neurons, to represent the input attribute membership function (or discrete value index) indicator. The length is $n \times m$, where n is the number of input attributes in raw dataset and m is the number of AND neurons. Each value in this section is an integer index value to indicate the fuzzy set to be applied. For example, for autompg dataset, if the gene is supposed to represent the input attribute of cylinders and the value is 2, this means the third fuzzy set, the index is starting from 0, will be applied. Thus the discrete value 5 will be returned. If the gene is used to stand for the input attribute horsepower and the value is 2, the fuzzy set representing "*horsepower is high*" will be applied to process the raw input value. The value for each gene in this section is set to be the integer values ranging from 0 to number of fuzzy sets defined on the related input attributes minus 1.

The other section in the chromosome is used for weights connecting inputs to hidden layer AND neurons and hidden layer AND neurons to output layer OR neuron. The length for second part is $n \times m + m$. The gene is this section is the real number in the range *[0,1]*.

we defined the accuracy of the model, Accu(FNN), based on the RMSE for training data in equation (1).

$$Accu(FNN) = \frac{1}{1 + RMSE_{training}}$$

(3)

The accuracy of the model is the larger the better, so it meets the requirement for GA fitness function. We use equation (3) to calculate the fitness value for each of the chromosome in the population.

The pruning procedure introduced here is based on two thresholds *(λ, μ)* for OR and AND neurons respectively. The underlying concept is straightforward and relates to the nature of the *AND* and *OR* neurons and their monotonic with regard to the values of the corresponding connections. Owing to the expression for the *OR* neuron, we note that the higher the value of the connection, the more essential the input. Lower values of the connections could then be dropped (pruned). To formalize this effect, let us introduce a certain threshold λ with the values confined to the *[0, 1]*:

$$w_{\lambda} = \begin{cases} w & \text{if } w \geq \lambda \\ 0 & \text{otherwise} \end{cases}$$

(4)

which returns an original value of the connection *(w)* if it exceeds or is equal to λ. Otherwise, it is too weak to keep. We prune the weak connections and replace them with zero values. The operation reflects our previous observation that weaker connections are related to the variables that could be eliminated. The higher the threshold, the more connections are eliminated from the original network, the more simple the network is, the less accurate the reduced FNN.

The threshold operation for the *AND* neurons are defined as follows:

$$w_{\mu} = \begin{cases} w & \text{if } w \le \mu \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

where we use a certain different threshold, say μ. Now, the connections whose values are above the threshold μ are pruned. By setting their values to 1 (corresponding to a complete masking effect), we eliminate the corresponding input. The lower the value of the threshold, the more radical the resulting pruning of the *AND* neurons. The combination of these two thresholds *(λ, μ)* serves as a general pruning mechanism. The choice of their values arises as a compromise between the reduced accuracy of the network and its increasing interpretability. More radical pruning (with the values of λ close to one and values of μ close to zero) leads to higher approximation error, but at the same time, results are in more compact and therefore easily interpretable structure (its underlying logic expression). These two characteristics are in conflict. Possible optimization is very much user-oriented: we may wish move with further reductions of the network given its interpretation benefits from the reduced form of the model and accept somewhat higher values of the approximation error.

After the FNN is trained with GA, the thresholds for AND/OR neurons will be selected to reduce the complexity of the rules. The range for the thresholds is from 0 to 1, since the weights in FNN are in the range of *[0,1]*. When carrying out Spruning, our focus will be put on how to get simplest structure of resulting FNN trained by GA, while keeping the FNN working with acceptable accuracy. We defined the simplicity of FNN, Simp(FNN).

$$Simp(FNN) = \frac{N_{\text{Cutweights}}}{N_{weights}}$$

(6)

Thus, when we do pruning, we need to satisfy two competing objectives, accuracy and simplicity of the FNN to find the best thresholds for the AND and OR neurons. The two objectives are all monotonically increasing, so in order to gain the best trade-off between the model accuracy on dataset and the rule-based system simplicity, the multi-objective optimization is used to find the optimal thresholds. The multi-objective optimization method adopted here is Pareto approach. It is trained with multi-objective GA (MOGA), modified GA to realize Pareto approach.

The MOGA chromosome for Pareto training just contains two real number values in *[0,1]*, matching the thresholds for AND and OR neurons, as shown in figure 6.4.

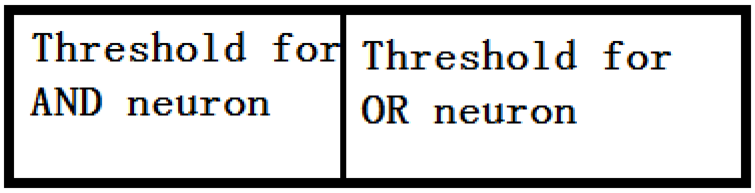


Figure 6.4 Chromosome for MOGA of Pareto

6.4 Experiments on data modeling

Data modeling is to define and analyze the data requirement based on given data samples in order to be able to predict the future data input.

For above 8 datasets described in table II-1, the FNN containing 9 AND neurons ($m=9$) and 1 OR neuron is selected in the following experiments. The experiments were carried out with 60-40 random training/testing split. The performance is reported as RMSE. The GA mutation rate is 0.1 and the crossover rate is 0.8. Population size is 100 and maximum generation is 100.

The GA performance on some of the datasets is shown in figure 6.5.

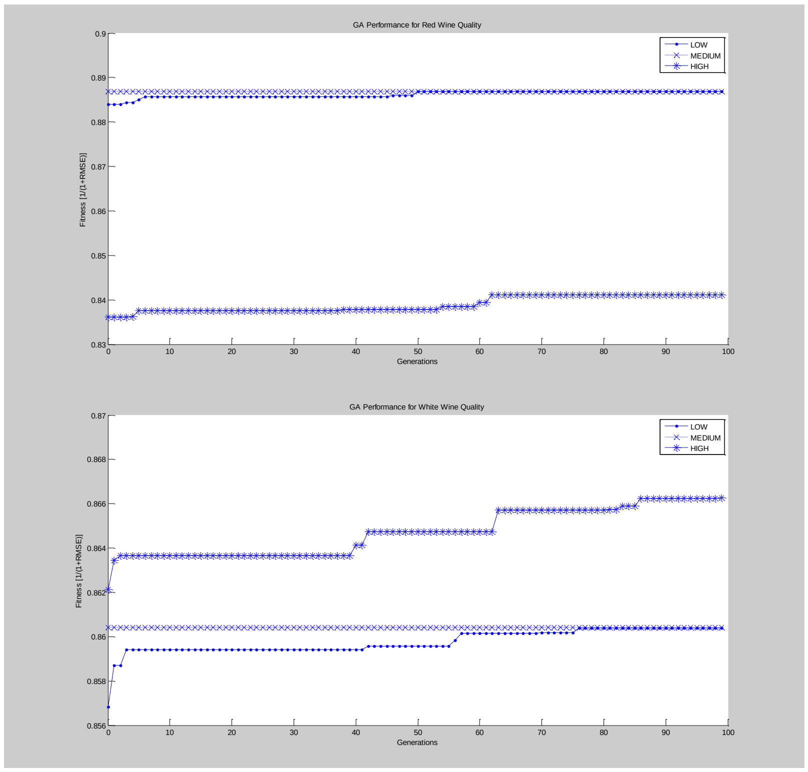


Figure 6.5 GA performances over generations on Wine Quality datasets (red&white)

The rules got from the GA trained FNN are containing all the input attributes. Taking auto mpg dataset results as the example, the rule sets for low, medium and high mpg are as following:

IF

{[Cylinders IS 5]0.000 AND [Displacement IS LOW]1.000 AND [Horsepower IS MEDIUM]0.000 AND [Weight IS MEDIUM]0.372 AND [Acceleration IS MEDIUM]0.791 AND [ModelYear IS 79]0.845 AND [Origin IS 1]1.000}1.000

OR {[Cylinders IS 6]0.000 AND [Displacement IS HIGH]0.940 AND [Horsepower IS MEDIUM]0.951 AND [Weight IS MEDIUM]0.981 AND [Acceleration IS MEDIUM]0.466 AND [ModelYear IS 79]0.879 AND [Origin IS 1]0.980}1.000

OR {[Cylinders IS 6]0.083 AND [Displacement IS MEDIUM]0.861 AND [Horsepower IS MEDIUM]0.000 AND [Weight IS LOW]0.871 AND [Acceleration IS LOW]0.635 AND [ModelYear IS 82]0.754 AND [Origin IS 1]0.642}1.000

OR {[Cylinders IS 6]0.211 AND [Displacement IS MEDIUM]1.000 AND [Horsepower IS MEDIUM]0.168 AND [Weight IS LOW]0.801 AND [Acceleration IS HIGH]0.771 AND [ModelYear IS 77]0.556 AND [Origin IS 1]0.165}0.454

OR {[Cylinders IS 3]0.219 AND [Displacement IS LOW]0.466 AND [Horsepower IS MEDIUM]0.191 AND [Weight IS MEDIUM]0.467 AND [Acceleration IS LOW]0.755 AND [ModelYear IS 70]0.917 AND [Origin IS 1]0.790}0.303

OR {[Cylinders IS 5]0.851 AND [Displacement IS MEDIUM]0.823 AND [Horsepower IS MEDIUM]0.000 AND [Weight IS LOW]1.000 AND [Acceleration IS MEDIUM]0.851 AND [ModelYear IS 74]0.947 AND [Origin IS 2]0.838}0.792

OR {[Cylinders IS 4]0.927 AND [Displacement IS LOW]0.963 AND [Horsepower IS MEDIUM]0.891 AND [Weight IS MEDIUM]0.110 AND [Acceleration IS HIGH]0.526 AND [ModelYear IS 73]0.755 AND [Origin IS 3]0.537}0.407

OR {[Cylinders IS 3]0.117 AND [Displacement IS MEDIUM]0.665 AND [Horsepower IS LOW]0.578 AND [Weight IS MEDIUM]1.000 AND [Acceleration IS LOW]0.374 AND [ModelYear IS 81]0.612 AND [Origin IS 2]0.000}0.108

OR {[Cylinders IS 6]0.883 AND [Displacement IS LOW]0.249 AND [Horsepower IS LOW]0.174 AND [Weight IS MEDIUM]0.354 AND [Acceleration IS LOW]0.402 AND [ModelYear IS 70]0.986 AND [Origin IS 2]0.875}0.000

THEN MPG IS LOW

IF

{[Cylinders IS 3]0.843 AND [Displacement IS MEDIUM]0.044 AND [Horsepower IS LOW]0.411 AND [Weight IS MEDIUM]0.665 AND [Acceleration IS LOW]0.540 AND [ModelYear IS 76]0.390 AND [Origin IS 1]0.753}0.749

OR {[Cylinders IS 6]0.403 AND [Displacement IS MEDIUM]0.310 AND [Horsepower IS LOW]0.307 AND [Weight IS MEDIUM]0.806 AND [Acceleration IS MEDIUM]0.398 AND [ModelYear IS 81]0.166 AND [Origin IS 1]0.287}0.967

OR {[Cylinders IS 5]0.979 AND [Displacement IS MEDIUM]0.263 AND [Horsepower IS LOW]0.262 AND [Weight IS MEDIUM]0.416 AND [Acceleration IS LOW]0.877 AND [ModelYear IS 80]0.089 AND [Origin IS 1]0.092}0.331

OR {[Cylinders IS 4]0.394 AND [Displacement IS LOW]0.366 AND [Horsepower IS MEDIUM]0.881 AND [Weight IS MEDIUM]0.331 AND [Acceleration IS LOW]0.626 AND [ModelYear IS 77]0.426 AND [Origin IS 1]0.630}0.473

OR {[Cylinders IS 5]0.735 AND [Displacement IS MEDIUM]0.166 AND [Horsepower IS MEDIUM]0.432 AND [Weight IS MEDIUM]0.073 AND [Acceleration IS LOW]0.984 AND [ModelYear IS 78]0.802 AND [Origin IS 1]0.048}0.987

OR {[Cylinders IS 5]0.664 AND [Displacement IS LOW]0.893 AND [Horsepower IS LOW]0.045 AND [Weight IS LOW]0.354 AND [Acceleration IS LOW]0.793 AND [ModelYear IS 74]0.890 AND [Origin IS 2]0.330}0.208

OR {[Cylinders IS 6]0.398 AND [Displacement IS MEDIUM]0.773 AND [Horsepower IS MEDIUM]0.166 AND [Weight IS MEDIUM]0.184 AND [Acceleration IS LOW]0.802 AND [ModelYear IS 78]0.641 AND [Origin IS 1]0.407}0.017

OR {[Cylinders IS 5]0.655 AND [Displacement IS LOW]0.591 AND [Horsepower IS MEDIUM]0.294 AND [Weight IS LOW]0.512 AND [Acceleration IS MEDIUM]0.792 AND [ModelYear IS 79]0.899 AND [Origin IS 2]0.993}0.391

OR {[Cylinders IS 5]0.199 AND [Displacement IS MEDIUM]0.435 AND [Horsepower IS LOW]0.713 AND [Weight IS MEDIUM]0.765 AND [Acceleration IS LOW]0.919 AND [ModelYear IS 71]0.113 AND [Origin IS 2]0.297}0.721

THEN MPG IS MEDIUM

IF

{[Cylinders IS 3]0.000 AND [Displacement IS MEDIUM]1.000 AND [Horsepower IS LOW]0.397 AND [Weight IS MEDIUM]0.000 AND [Acceleration IS MEDIUM]0.521 AND [ModelYear IS 76]0.122 AND [Origin IS 1]0.000}0.087

OR {[Cylinders IS 3]1.000 AND [Displacement IS LOW]0.739 AND [Horsepower IS LOW]0.169 AND [Weight IS LOW]0.000 AND [Acceleration IS LOW]0.000 AND [ModelYear IS 80]0.934 AND [Origin IS 1]0.912}0.546

OR {[Cylinders IS 5]0.235 AND [Displacement IS LOW]0.284 AND [Horsepower IS LOW]0.094 AND [Weight IS HIGH]0.385 AND [Acceleration IS LOW]0.810 AND [ModelYear IS 80]0.006 AND [Origin IS 2]0.099}0.000

OR {[Cylinders IS 3]0.000 AND [Displacement IS LOW]0.523 AND [Horsepower IS HIGH]0.522 AND [Weight IS LOW]0.000 AND [Acceleration IS MEDIUM]1.000 AND [ModelYear IS 73]1.000 AND [Origin IS 1]0.664}0.000

OR {[Cylinders IS 5]0.066 AND [Displacement IS LOW]1.000 AND [Horsepower IS LOW]0.362 AND [Weight IS HIGH]0.815 AND [Acceleration IS LOW]0.000 AND [ModelYear IS 81]1.000 AND [Origin IS 1]0.131}0.494

OR {[Cylinders IS 3]0.000 AND [Displacement IS LOW]0.682 AND [Horsepower IS LOW]0.242 AND [Weight IS HIGH]1.000 AND [Acceleration IS HIGH]0.000 AND [ModelYear IS 82]0.234 AND [Origin IS 2]0.000}0.000

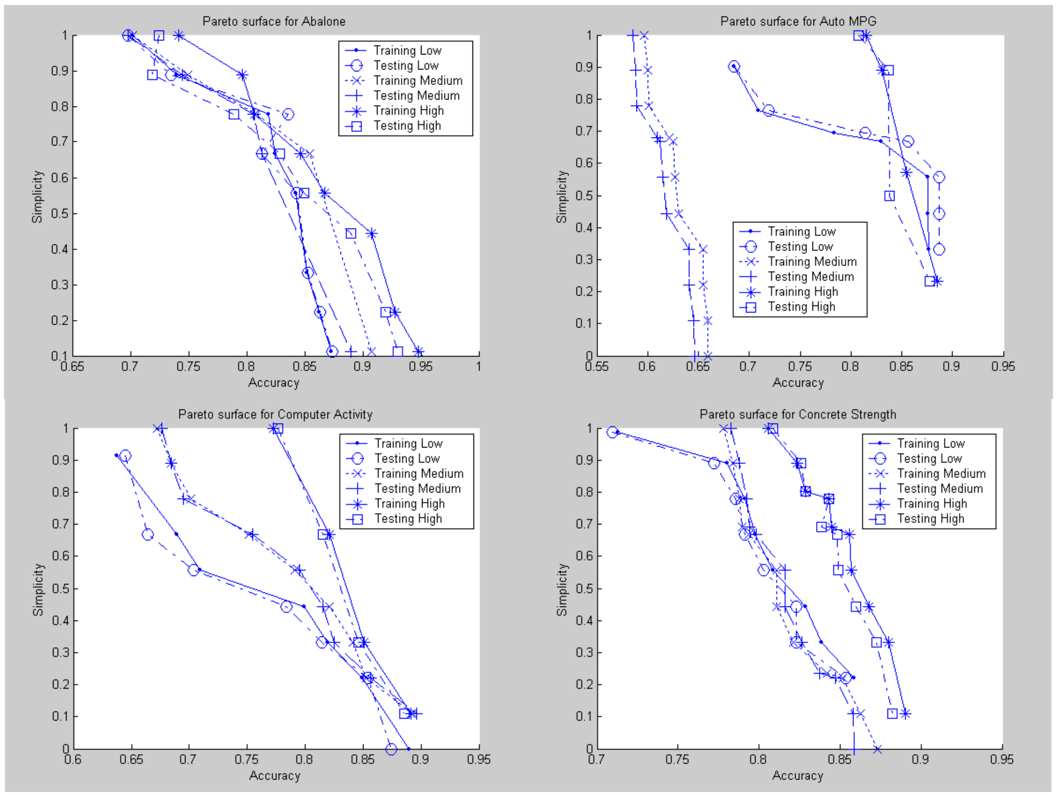
OR {[Cylinders IS 8]0.000 AND [Displacement IS HIGH]0.194 AND [Horsepower IS MEDIUM]0.817 AND [Weight IS HIGH]0.442 AND [Acceleration IS HIGH]1.000 AND [ModelYear IS 70]0.354 AND [Origin IS 3]0.853}0.000

OR {[Cylinders IS 4]0.953 AND [Displacement IS LOW]0.507 AND [Horsepower IS HIGH]0.000 AND [Weight IS MEDIUM]0.652 AND [Acceleration IS LOW]1.000 AND [ModelYear IS 70]0.000 AND [Origin IS 1]0.986}0.000

OR {[Cylinders IS 4]0.821 AND [Displacement IS LOW]1.000 AND [Horsepower IS LOW]0.939 AND [Weight IS HIGH]0.604 AND [Acceleration IS MEDIUM]0.000 AND [ModelYear IS 80]0.000 AND [Origin IS 2]0.458}0.000

THEN MPG IS HIGH

Pareto approach is used to reduce the complexity of the rules. Two goals are chosen as competitive objects in Pareto, accuracy and simplicity. The two goals are all trying to get the optimal maxima and conflicting with each other. The Pareto surface is shown as below.



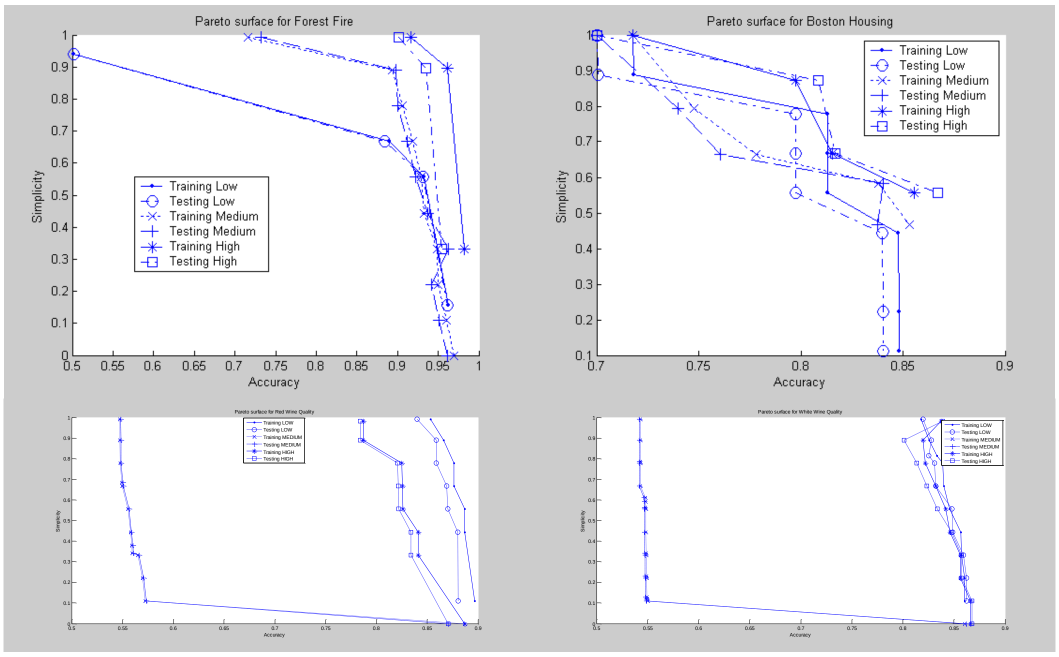


Figure 6.6 Pareto surfaces for each dataset

If selecting an acceptable accuracy and a good simplicity, for above auto mpg dataset rules, picking (accuracy, simplicity) pairs of (0.829770, 0.666667) for low, (0.621555, 0.680556) for medium, and (0.870625, 0.888889) for high. The resulting rule set becomes much simpler.

IF

{[Cylinders IS 5]0.000 AND [Displacement IS LOW]1.000 AND [Horsepower IS MEDIUM]0.000 AND [Weight IS MEDIUM]0.372 AND [Acceleration IS MEDIUM]0.791 AND [ModelYear IS 79]0.845 AND [Origin IS 1]1.000}1.000

OR {[Cylinders IS 6]0.000 AND [Displacement IS HIGH]0.940 AND [Horsepower IS MEDIUM]0.951 AND [Weight IS MEDIUM]0.981 AND [Acceleration IS MEDIUM]0.466 AND [ModelYear IS 79]0.879 AND [Origin IS 1]0.980}1.000

OR {[Cylinders IS 6]0.083 AND [Displacement IS MEDIUM]0.861 AND [Horsepower ISMEDIUM]0.000 AND [Weight IS LOW]0.871 AND [Acceleration IS LOW]0.635 AND [ModelYear IS 82]0.754 AND [Origin IS 1]0.642}1.000

THEN MPG IS LOW

IF

{[Cylinders IS 3]0.843 AND [Displacement IS MEDIUM]0.044 AND [Horsepower IS LOW]0.411 AND [Weight IS MEDIUM]0.665 AND [Acceleration IS LOW]0.540 AND [ModelYear IS 76]0.390 AND [Origin IS 1]0.753}0.749

OR {[Cylinders IS 6]0.403 AND [Displacement IS MEDIUM]0.310 AND [Horsepower IS LOW]0.307 AND [Weight IS MEDIUM]0.806 AND [Acceleration IS MEDIUM]0.398 AND [ModelYear IS 81]0.166 AND [Origin IS 1]0.287}0.967

OR {[Cylinders IS 5]0.735 AND [Displacement IS MEDIUM]0.166 AND [Horsepower IS MEDIUM]0.432 AND [Weight IS MEDIUM]0.073 AND [ModelYear IS 78]0.802 AND [Origin IS 1]0.048}0.987

THEN MPG IS MEDIUM

IF

{[Cylinders IS 3]0.000 AND [Horsepower IS LOW]0.397 AND [Weight IS MEDIUM]0.000

AND [Acceleration IS MEDIUM]0.521 AND [ModelYear IS 76]0.122 AND [Origin IS 1]0.000}0.087

OR {[Horsepower IS LOW]0.169 AND [Weight IS LOW]0.000 AND [Acceleration IS LOW]0.000}0.546

OR {[Cylinders IS 5]0.066 AND [Horsepower IS LOW]0.362 AND [Acceleration IS LOW]0.000 AND [Origin IS 1]0.131}0.494

THEN MPG IS HIGH

From above results for autompg dataset, we can see that the rule set after applying the pruning method become much simpler and more interpretable. The number of rules in the rule set dropped from 9 to 3 for each of the three outputs. The dimension is also reduced for some rules. For example, for output "*MPG is HIGH*", the three rules have input numbers of 6, 4 and 3 respectively, compared with the original 7 attributes for each rules before pruning. Table 6.2 summarized the rule complexity reduction results for each dataset on every fuzzy set for the output. The pruning thresholds for AND and OR neurons are selected based on the Pareto front by choosing the acceptable balance between the simplicity (interpretability) and the accuracy of the FNN.

The rule complexity before pruning has the entry of format $n \times m$, where n is the number of input attributes in each rule and m is the number of rules, e.g., for autompg dataset, "MPG is HIGH" output, 7×9 means there are in total 9 rules and 7 input attributes for each rule. The entries in column rule complexity after pruning are displayed as the aggregation of a serial of integers, in the format of $n_1+n_2+..+n_r$, where the number of coefficient, r, is the number of rule and each coefficient indicates the number of input attributes in that specific rule. For example, for output "*MPG is HIGH*", the entry is $6 + 4 +3$. This means that there are in total 3 rules left after pruning and the three rules have number of input attributes as 6, 4 and 3 respectively.

Dataset		Rule complexity before pruning	Rule complexity after pruning
Abalone	Low	8 ×9	5 + 3
	Med	8 ×9	4 + 3 + 3
	High	8 ×9	2 + 2 + 3 + 3
Autompg	Low	7 ×9	7 + 7 + 7
	Med	7 ×9	7 + 7 + 6
	High	7 ×9	6 + 4 + 3
Boston housing	Low	13 ×9	6 + 5 + 3 + 2
	Med	13 ×9	5 + 3 + 3
	High	13 ×9	7 + 4 + 3 + 3 + 2
Computer activity	Low	21 ×9	8 + 6 + 2
	Med	21 ×9	7 + 6
	High	21 ×9	3 + 3 + 3
Concrete strength	Low	8 ×9	4 + 3 + 2
	Med	8 ×9	5 + 2 + 2
	High	8 ×9	3 + 3
Forest fires	Low	12 ×9	4 + 2
	Med	12 ×9	3 + 3
	High	12 ×9	6 + 6 + 5 + 4 + 3 + 2
Red wine quality	Low	11 ×9	5 + 4 + 2
	Med	11 ×9	4 + 3
	High	11 ×9	3 + 3 + 3
White wine quality	Low	11 ×9	4 + 2 + 2
	Med	11 ×9	5 + 3 + 2
	High	11 ×9	4 + 3 + 3 + 2

Table 6.2 Rule complexity reduction results

From table 6.1, we can summarize that the models after pruning with acceptable balancing between simplicity and accuracy really produce the good interpretability from the generated FNN model for all the 8 datasets in our experiments.

6.5 Discussion

In this study, we have proposed a method to make use of fuzzy neural network composed of AND/OR fuzzy neurons. By adjusting the connections of the neurons one could easily model intermediate logic characteristics of the logic mapping. We have proposed a genetic scheme of optimization of the network with intent of addressing the structural facet of learning.

In the design of the network we have demonstrated the role of the interface between the physical variables and those of logic character required by the model. The encoding scheme dwells on fuzzy sets treated as a collection of semantically sound information granules. By using Pareto approach in getting better accuracy and simplicity, we can choose the best model suitable for the actual needs.

From the experiments, we can see that the construction of the rule-based system and the pruning of the system are efficient and acceptable.

Reference

[6_1] Q. Hu, W. Pan, Lei Zhang, D. Zhang, etc., Feature Selection for Monotonic Classification, IEEE TRANSACTIONS ON FUZZY SYSTEMS, VOL. 20, NO. 1, FEBRUARY 2012, pp: 69-81

[6_2] P. Maji and S. K. Pal, Feature Selection Using f-Information Measures in Fuzzy Approximation Spaces, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 22, NO. 6, JUNE 2010, pp: 854-867

[6_3] J.J. Shann, H.C. Fu, A fuzzy neural network for rule acquiring on fuzzy control systems, Fuzzy Sets and Systems, Volume 71, Issue 3, 12 May 1995, Pages 345-357

[6_4] Y. Yao, J. Mi, Z. Li, Attribute reduction based on generalized fuzzy evidence theory in fuzzy decision systems, Fuzzy Sets and Systems, Volume 170, Issue 1, 1 May 2011, Pages 64-75

[6_5] N. Xiong, L. Litz, Reduction of fuzzy control rules by means of premise learning – method and case study, Fuzzy Sets and Systems, Volume 132, Issue 2, 1 December 2002, Pages 217-231

Chapter 7 Neuro-fuzzy Controller for Deployment of Sensors

Deployment methods using mobile nodes have been proposed to enhance network coverage and to extend the network lifetime via configuration of uniformly distributed node topologies from random node distributions. Since mobility itself requires energy from its own limited energy source, a deployment scheme should be designed carefully to minimize energy consumption during deployment as well as achieve certain goals such as satisfactory coverage and/or an energy efficient node topology. Moreover, it is desirable for a distributed sensor network node to have relatively simple hardware architecture, which requires minimal computing power and memory. Each node should have a simple and efficient algorithm for deployment, organization, and management of the WSN. Deployment process itself is very energy consuming due to the locomotive action as well as computation and communication associated with it. Not only minimizing average moving distance, but also reducing the difference of the remaining energy among sensor nodes is essential for a longer system lifetime. Due to the dynamic and distributed nature of deployment, it is a challenging task to obtain full coverage in the region of interest (ROI) and to utilize energy of each sensor in a relatively fair fashion.

In this chapter, we applied FNN based FRBS, proposed in Chapter 6, into the deployment of the wireless sensor networks. Fuzzy C-means based FRBS will not be tackled for this controlling issue.

7.1 Overview and Architecture

Neuro-fuzzy Controller is applied in the strategy of sensor node deployment. The goal of the deployment is to achieve the high coverage and good uniformity. The fuzzy controller will direct the nodes to move away from crowd environment. In consequence, more uncovered area will be in the sensing range of sensor nodes and the uniformity will be improved. The effect resulting from the fuzzy logic system is the sensor's traveling distance during the redeployment.

- ***Inputs***

There are two antecedents to indicate the metrics for rule-based deployment. The first antecedent is for local density (LD). The neighbour is defined as the other sensors inside the communication range, $R_{c,}$ of the sensor. The value for LD is the number of neighbours plus 1 and ranging from 1 (isolated) to total node number *n*, that is *[1,n]*. The second antecedent is for average Euclidean distance (AED) from current sensor to all its neighbours. The calculation of Euclidean distance between two n-dimension nodes, ***p*** and ***q***, is shown in equation (1), where ***p*** is represented as $(p_1, p_2, \ldots , p_n)$ and ***q*** is represented as $(q_1, q_2, \ldots , q_n)$.

$$ED_{p,q} = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{1}$$

To simplify the symbol here, we denote ED_i as the Euclidean distance from current sensor to its i^{th} neighbour.

The two antecedents will be useful for improving the coverage and to increase the uniformity. These two characters described the crowdedness in the specified area. If it is too crowd, the sensor should move a little far from its neighbours. On the other hand, it might move a shorter distance or even stand still. The distance to the neighbours indicates the crowdedness in the surrounding area. The range for AED is from 0 to communication range (R_c). The network lifetime is close related to each node's lifetime. If one node dies, the whole wireless sensor network might become non-functioning. So the remaining power for each node should be as similar as possible to extend the network lifetime. The remaining power of the nodes is closely related to the moving distance during the deployment stage. In order to extend the lifetime of the WSN, the deployment algorithm needs to minimize the movement of the nodes in the network. Table 7.1 summarizes the antecedents interested in carrying out node deployment in WSN.

antecedent	**Calculation**	**Range**
local density (LD)	$LD = N_{neigbour} + 1$	*[1,n]*
average Euclidean distance (AED)	$AED = \frac{\sum_{i=1}^{m} ED_i}{m}$	*[0,R_c]*

Table 7.1 Antecedent description

- ***Fuzzification***

The three antecedents provide numeric inputs to the FLS, which are actually inputs for the fuzzifier in the FLS. Three fuzzy sets are defined for the antecedents. The fuzzy sets use trapezoid and triangular membership functions.

- ***Fuzzification for Local Density***

Local density (LD) is the counting of the nearby nodes within current sensor's communication range. The three fuzzy sets are labelled as low, moderate and high.

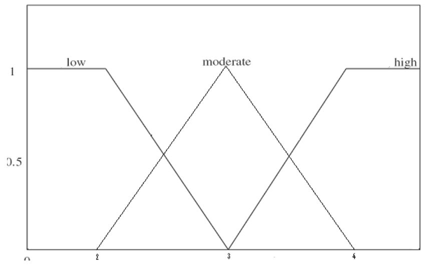
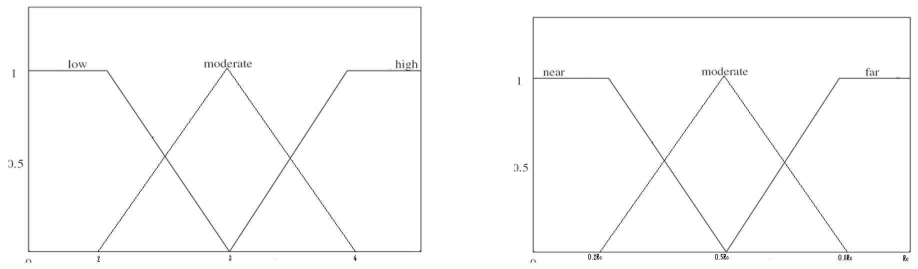


Figure 7.1 Fuzzy sets for local density

- ***Fuzzification for Average Euclidean Distance***

Average Euclidean distance (AED) is the result of the mean value for all the distance between the node and its neighbours. The minimum value for AED is 0, which means all nodes are located at the same spot. The maximum value is the communication range, $R_{c,}$ which indicates that all the neighbours are in the marginal range of the definition of a neighbour. The three fuzzy sets are labelled as near, moderate and far.



a) For local density b) For AED and consequence

Figure 7.2 Fuzzy sets defined for the input/output variables

- ***Fuzzy Sets for Consequence***

The consequence of the FLS, which is the travel distance for the sensor node in current deployment iteration, works in the range ***[0,B]***, where ***B*** is the base moving distance. B is also used in the scaling stage. The consequence also has three fuzzy sets. They are denoted as near, moderate and far, which were shown as Figure 7.2.

- ***Fuzzy neural network structure***

The fuzzy neural network will have three OR neurons matching the three fuzzy set output of the consequence. The number of AND neurons in hidden layer is equal to the max number of rules. For 2 numeric inputs with each having 3 fuzzy sets, the max number of rules will be 9, then the fuzzy neural network will have 9 AND neurons. Each AND neuron is connected to the 3 OR neurons at the output layer. For 3 numeric inputs with each having 3 fuzzy sets, the max number of rules will be 27, then the fuzzy neural network will have 27 AND neurons. Each AND neuron is connected to the 3 OR neurons at the output layer. Here shows the fuzzy neural network structure with 2 numeric inputs.

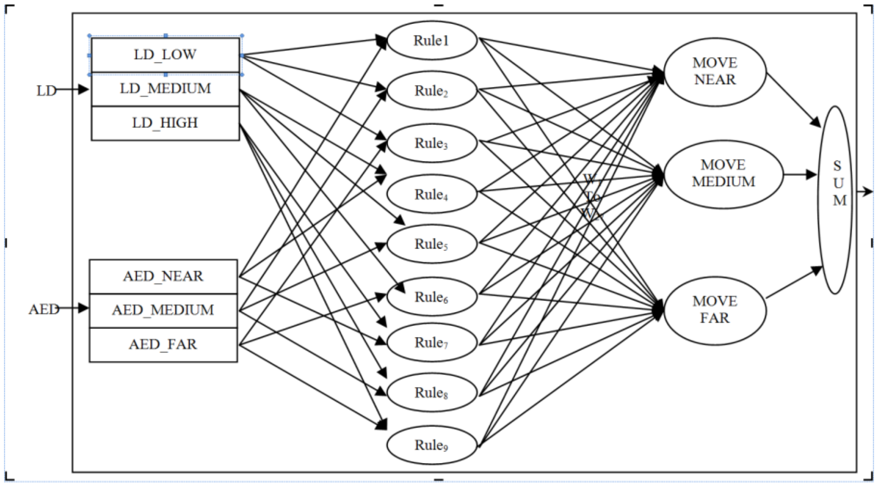


Figure 7.3 fuzzy neural network structures for two numeric inputs

- ***Inference and defuzzification of the output***

Taking above fuzzy network as example in Figure 7.3, for each consequence fuzzified output, a summarizing node is defined, which collects the rules that give the corresponding consequence fuzzy value. The 3 outputs of the summarizing nodes will go into the final processing node, where the following calculation is done.

$$F = \frac{O_1 \times P_1 + O_2 \times P_2 + O_3 \times P_3}{O_1 + O_2 + O_3} \tag{2}$$

Where O_i is the output from i^{th} summarizing node, and P_i is the dividing point for i^{th} fuzzy scope of the consequence. With this calculation, the final value will be in the real world range, no scaling operation needed.

- ***Moving direction***

The sensor's moving direction is calculated using the Coulomb's law in physics, which computes the sum of the vector's subtraction. If the sensor locates at position $\vec{S}(x, y)$ and it has n neighbours, whose position is denoted as $\vec{S}_i(x_i, y_i)$ for i^{th} neighbour, the final vector

$$\vec{S}_{final}(x', y') = \sum \frac{\vec{S}_{(x,y)} - \vec{S}_{(x_i,y_i)}}{\left|\vec{S}_{(x,y)} - \vec{S}_{(x_i,y_i)}\right|^2} \tag{3}$$

The direction will be computed with $\vec{S}_{final}(x', y')$. The angle α is

$$tag(\alpha) = \frac{y'}{x'}, \alpha = arctag(\frac{y'}{x'}) \tag{4}$$

As a result, the new position $\vec{S}_{new}(x_{new}, y_{new})$ is

$$\begin{aligned} x_{new} &= x + D \times \cos(\alpha) \\ y_{new} &= y + D \times \sin(\alpha) \end{aligned} \tag{5}$$

Figure 7.4 describes the calculation of the moving angle when the node $S(x, y)$ has two neighbours $S(x_1, y_1)$ and $S(x_2, y_2)$.

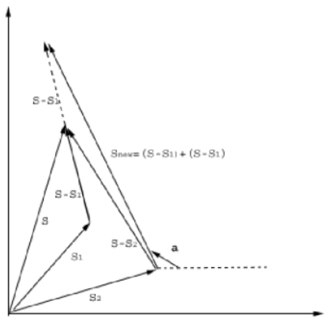


Figure 7.4 Calculation of the moving angle

- ***Special Treatment for the Isolated Nodes***

When the node has no neighbour, it is an isolated node. Since the isolated nodes are totally no use for the wireless sensor network. The information got by the isolated nodes could not be shared with other nodes and the base station, so we should try best to eliminate the isolated nodes. For the isolated node, the FLS will not be used to direct the movement of the node. Instead, the node will move a fixed distance in the possible direction of meeting other nodes. Here the fixed distance is used as the maximum possible movement distance of a node, which is defined as the base movement distance, ***B***. In this research, *B* will be using $0.1\times R_c$, $0.2\times R_c$ or $0.3\times R_c$ as sample experiments. For the moving direction of the isolated node, we define 4 areas for the location of the node and shown in Figure 7.5. If the node is in area 1, it will be moving toward P2. If it is in area 2, the direction will be facing P4 and area 3 for P1 and area 4 for P3.

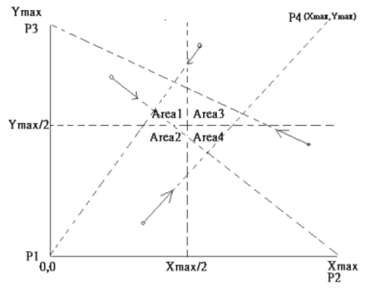


Figure 7.5 Moving directions for isolated nodes

7.2 Pruning to obtain rules

In the heart of fuzzy logic networks lies their interpretability. In essence this means that we can take the optimized network and effortlessly produce its logic expression. As the neurons come with a well- defined semantics, the transparency of the network is evident. There is however, a possibility of producing even a more condensed logic description by confining ourselves to the most "essential" part of the network. This is accomplished by reducing the weakest and least meaningful connections. This concept of pruning is guided by the properties of AND and OR neurons and the underlying t- and s- norms, in particular.

The pruning procedure introduced in this section is inherently associated with the (λ, μ) notation hence the name of the pruning procedure. The underlying concept is straightforward and relates to the nature of the AND and OR neurons and their monotonicity with regard to the values of the corresponding connections. Owing to the expression for the OR neuron, we note that the higher the value of the connection, the more essential the input. Lower values of the connections could be then dropped (pruned). To formalize this effect, let us introduce a certain threshold λ with the values confined to the [0,1]. If $\boldsymbol{w}$ is quite weak ($\boldsymbol{w<\lambda}$), we prune the connection and return a zero value. Or else $\boldsymbol{w_\lambda}$ is set to the original value of the connection ($\boldsymbol{w}$) if it exceeds or is equal to $\boldsymbol{\lambda}$. The higher the threshold, the more connections become eliminated from the original network. The operation reflects our previous observation that weaker connections relate to the variables that could be eliminated.

Similarly, the threshold could be defined for the AND neurons is defined, say μ in range *[0,1]*. Now the connections whose values are above the threshold μ are pruned by setting their values to 1 (that corresponds to a complete masking effect). Thus we eliminate the corresponding input. The lower the value of the threshold, the more radical the resulting pruning of the AND neurons. The combination of these two thresholds (λ, μ) serves as a general pruning mechanism. The choice of their values arises as a compromise between the reduced accuracy of the network and its increasing interpretability. More radical pruning (with the values of λ close to one and values of μ close to zero) leads to higher approximation error but at the same time results in more compact and therefore easily interpretable structure (its underlying logic expression). These two characteristics are in conflict. Possible optimization is very much user-oriented: we may wish move with further reductions of the network given its interpretation benefits from the reduced form of the model and accept somewhat higher values of the approximation error.

7.3 Simulator for WSN deployment

Simulator could model and simulate Wireless Sensor Networks. There are a number of simulators available, such as NS-2, J-Sim, MatLab’s Prowler (Probabilistic Wireless Network Simulator) toolbox, etc. Those available simulators are having their merits in simulating the WSN behaviors, but either it is not easy (or unable) to modify the functionality, or not suitable for deployment process. After putting much effort in making an existing simulator working for our research, we started to create a simulator for WSN deployment. Currently, the simulator is working properly for node deployment. The simulator is designed as combination of different modules, which makes it easy to do experiments for different deployment algorithms. Currently, the following algorithms have been added in: DSSA, fuzzy rule-based deployment algorithm in [57] and our neuro-fuzzy networks.

7.4 Experiments

- *General setting*

Some preliminary experiments are carried out to train the fuzzy neural networks. Since the sensor nodes equipped with mobile device are more expensive than the static sensor nodes. When dealing with the deployment for mobile wireless sensor network, our focus should be lay on the scenario about using less sensor nodes to cover the whole region of interest. Here, the sample experiments are done for the WSN not able to cover the whole area. The sensor node parameters are communication range R_c=100, sensing range R_s=50, the region of interest with the size as 500 * 500, or 250 * 250. Two numeric inputs, local density and average distance, are applied.

- *GA for constructing the FNN*

The GA was applied in training the model. The population size was set to 100 and the maximum generation is 100. The chromosome represents the connecting weights between neurons in the neuro-fuzzy network. To be generalized, the model will be run over 30 randomly initialized network and try to optimize on those WSNs. The fitness function is the average coverage over those separate runs.

The scope for this is in *[0,1]*. A large number of GA experiments were done based on different generation, population, crossover rate, and mutation rate.

In our setting, we defined 3 fuzzy sets on each of the 2 inputs, so the maximum number of rules will be 9. Our model is using all 9 rules as the AND neurons and connecting those AND neurons to the 3 output OR neurons matching 3 fuzzy sets on conclusion. With normal pruning, we will just cut off the connections based on the threshold. The final structure will have same AND neuron connecting to more than one output OR neurons. Logically, this is confusing when trying to interpret those rules. The new way of analyzing the structure is trying to avoid confusion.

- Simple pruning: 1 out of n connection pruning method

For each AND neuron, there will be n (here equals to 3) connections to each of the OR neuron. The simple pruning will cut off all the connections coming out of the AND neuron but the one with highest weight value. In our model, it will be kept one connection from each AND neuron and the other two will be cut off. There will not have duplicate rules among outputs.

- Pareto based pruning

Similar pruning approach was applied as done for FNN in chapter 6. Here the Pareto will try to find the optimal solution for three competitive objectives: maximum coverage, network complexity and the contrary of minimum moving distance. The contrary of the minimum moving distance is calculated with

$$\frac{B-D}{B} \tag{6}$$

where *B* is the base movement value and the D is the average moving distance for each node in each iteration of deployment.

Here we are taking advantage of our model. We are comparing the 3 weight coming out of each AND neuron to the 3 OR neurons. Thus the final model will take all 9 rules (currently, without applying pruning) scattered on 3 outputs. Based on above information, following experiments were carried out.

7.4.1 Coverage optimization with simple pruning

GA was used to train the FNN to construct the system. The fitness function is just the coverage of the WSN. The chromosome length is 27, which is the count of the number of connections between each of the 9 AND neurons and each of the 3 OR neurons. The chromosome uses real number in *[0,1]* for the gene value.

First, we want to see what the optimal value for maximum deployment iteration will be.

	10	15	20	25	30	35	40	45
0	25.4+2.0	35.1+2.6	44.2+2.9	51.1+3.2	57.9+3.6	63.6+3.4	68.1+3.3	72.9+3.4
1	26.2+1.9	35.7+2.5	44.3+2.9	51.1+3.2	57.6+3.5	63.6+3.4	68.1+3.4	73.0+3.3
2	27.6+1.7	38.1+2.4	47.4+2.9	54.8+3.2	61.6+3.5	67.9+3.3	72.6+3.3	77.4+3.2
3	28.4+1.5	39.8+2.2	49.8+2.8	57.7+3.1	65.0+3.4	71.5+3.2	76.4+3.2	81.1+3.0
4	29.0+1.3	41.0+2.0	51.5+2.6	60.0+3.0	67.7+3.3	74.5+3.0	79.5+3.1	84.1+2.8
5	29.4+1.3	41.8+1.9	52.9+2.5	61.9+2.9	69.8+3.1	76.8+2.9	81.9+2.8	86.4+2.6
6	29.6+1.0	42.4+1.7	53.9+2.4	63.2+2.8	71.4+3.0	78.5+2.6	83.6+2.6	88.0+2.3
7	29.8+0.9	42.9+1.6	54.7+2.2	64.2+2.7	72.6+2.8	79.8+2.5	84.9+2.4	89.1+2.1
8	29.9+0.9	43.3+1.5	55.3+2.1	64.9+2.6	73.5+2.7	80.7+2.3	85.8+2.1	89.9+2.0
9	30.1+0.8	43.7+1.4	55.7+2.0	65.5+2.5	74.1+2.6	81.4+2.2	86.4+1.9	90.4+1.9
10	**30.2+0.6**	**43.8+1.4**	**56.1+2.0**	**66.0+2.4**	**74.6+2.4**	**81.8+2.0**	**86.8+1.8**	**90.9+1.7**
11	30.2+0.6	44.0+1.3	56.5+1.9	66.4+2.4	75.0+2.4	82.2+2.0	87.1+1.7	91.1+1.5
12	30.3+0.5	44.2+1.3	56.8+1.8	66.8+2.3	75.2+2.3	82.4+1.9	87.4+1.7	91.3+1.4
13	30.3+0.5	44.3+1.2	57.0+1.8	67.1+2.2	75.5+2.3	82.6+1.7	87.7+1.6	91.5+1.4
14	30.4+0.5	44.4+1.1	57.2+1.7	67.3+2.2	75.7+2.2	82.8+1.7	87.8+1.5	91.6+1.3
15	30.4+0.5	44.5+1.0	57.3+1.7	67.6+2.1	75.9+2.2	82.9+1.7	87.9+1.5	91.7+1.2
16	30.4+0.4	44.6+1.0	57.5+1.6	67.7+2.1	76.1+2.1	83.0+1.7	88.0+1.4	91.8+1.2
17	30.4+0.4	44.7+1.0	57.5+1.7	67.9+2.1	76.3+2.2	83.0+1.6	88.1+1.4	91.9+1.1
49	30.5+0.4	45.2+0.7	58.4+1.4	69.2+1.8	77.4+1.9	83.7+1.4	88.6+1.1	92.5+1.0
Move	10.1+1.6	9.8+1.1	9.6+1.0	9.4+0.8	8.9+0.6	8.6+0.4	8.3+0.3	8.1+0.1

distance								
Orphan ratio	36.8+2.2	26.9+0.9	20.2+1.8	17.7+1.9	12.1+1.5	7.5+1.7	3.6+0.2	0.9+0.6

Table 7.2 R_c=100, R_s=50, moving base value 20 and 200 runs

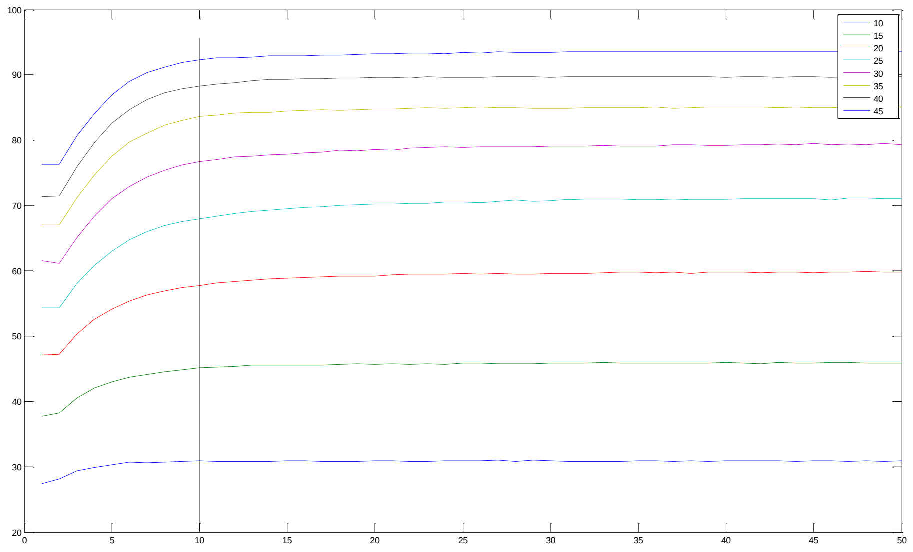


Figure 7.6 Plot for the coverage when base is 20

Similar experiments were redone on the changes to the base moving value of 10 and 30.

	10	15	20	25	30	35	40	45
0	25.3+1.9	35.4+2.4	44.1+3.0	51.5+3.2	58.1+3.1	63.0+3.3	68.2+3.2	72.1+3.6
1	25.8+1.9	35.7+2.4	44.2+2.9	51.5+3.2	58.0+3.1	63.0+3.3	68.2+3.2	72.2+3.6
2	26.7+1.8	37.0+2.4	45.9+2.9	53.4+3.2	60.1+3.1	65.3+3.3	70.5+3.2	74.5+3.6
3	27.4+1.7	38.2+2.4	47.4+2.9	55.1+3.2	62.1+3.0	67.4+3.2	72.7+3.2	76.7+3.5
4	28.0+1.6	39.1+2.3	48.6+2.8	56.7+3.2	63.9+2.9	69.3+3.1	74.7+3.1	78.7+3.5
5	28.5+1.5	40.0+2.2	49.8+2.8	58.1+3.2	65.5+2.9	71.0+3.0	76.5+3.1	80.5+3.4
6	28.9+1.4	40.7+2.2	50.7+2.8	59.3+3.1	66.9+2.8	72.6+3.0	78.1+3.0	82.1+3.3
7	29.1+1.3	41.3+2.1	51.6+2.7	60.4+3.0	68.2+2.7	74.0+2.8	79.6+3.0	83.6+3.2
8	29.4+1.3	41.8+2.1	52.4+2.6	61.4+3.0	69.4+2.6	75.3+2.7	81.0+2.9	84.9+3.1
9	29.6+1.2	42.2+2.0	53.1+2.6	62.3+2.9	70.4+2.5	76.5+2.6	82.1+2.8	86.1+3.0
10	29.8+1.1	42.6+2.0	53.7+2.5	63.1+2.9	71.3+2.4	77.5+2.5	83.2+2.7	87.0+2.8

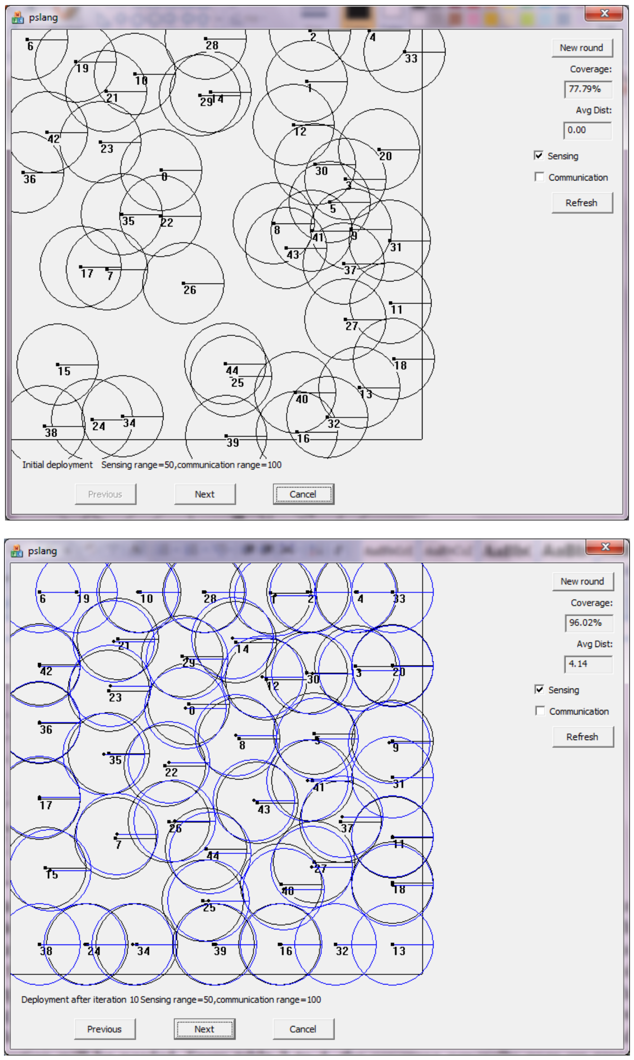
11	29.9+1.1	42.9+2.0	54.2+2.4	63.7+2.8	72.2+2.4	78.3+2.4	84.0+2.6	87.8+2.7
12	30.1+1.0	43.2+1.9	54.6+2.4	64.3+2.8	72.8+2.3	79.1+2.3	84.8+2.5	88.5+2.5
13	30.1+1.0	43.4+1.9	55.0+2.4	64.8+2.7	73.4+2.3	79.7+2.3	85.4+2.4	89.1+2.3
14	30.2+0.9	43.6+1.8	55.3+2.3	65.2+2.7	73.8+2.2	80.3+2.2	85.9+2.3	89.6+2.2
15	30.3+0.8	43.8+1.8	55.6+2.3	65.6+2.7	74.2+2.1	80.7+2.2	86.3+2.3	90.0+2.0
16	**30.4+0.7**	**44.0+1.7**	**55.8+2.3**	**66.0+2.7**	**74.6+2.1**	**81.1+2.1**	**86.7+2.2**	**90.3+1.9**
17	30.5+0.7	44.1+1.7	56.1+2.2	66.3+2.7	74.9+2.1	81.5+2.1	87.0+2.1	90.6+1.7
49	30.9+0.3	45.4+0.4	58.2+1.9	69.1+2.4	77.4+2.0	83.9+1.6	88.9+1.4	92.6+1.2
Move distance	5.2+0.7	5.0+0.6	4.9+0.5	4.6+0.4	4.4+0.3	4.3+0.2	4.1+0.1	4.1+0.1
Orphan ratio	34.1+1.9	25.6+1.6	19.9+2.1	16.2+0.9	10.6+2.5	6.6+0.8	2.8+0.2	0.6+0.6

Table 7.3 R_c =100, R_s =50, moving base value 10 and 200 runs

	10	15	20	25	30	35	40	45
0	25.2+1.9	35.5+2.6	44.0+3.0	51.4+3.2	57.9+3.4	63.6+3.5	68.2+3.2	71.8+3.5
1	26.2+1.9	36.0+2.6	44.1+2.9	51.1+3.4	57.6+3.4	63.4+3.4	68.1+3.2	71.6+3.5
2	28.0+1.6	39.1+2.4	48.3+2.8	56.2+3.3	63.5+3.4	69.5+3.4	74.6+3.1	78.0+3.4
3	28.9+1.2	40.9+2.2	51.1+2.6	59.8+3.1	67.7+3.2	74.0+3.2	79.4+3.0	82.8+3.1
4	29.4+1.0	41.8+1.9	52.8+2.4	62.2+2.9	70.3+2.9	77.1+2.8	82.6+2.7	86.1+2.7
5	29.7+0.9	42.4+1.8	53.9+2.2	63.5+2.7	71.9+2.6	79.0+2.4	84.6+2.3	88.1+2.3
6	29.7+0.9	42.9+1.7	54.6+2.1	64.4+2.7	72.8+2.4	80.1+2.2	85.7+2.0	89.2+2.0
7	**29.7+0.8**	**43.1+1.5**	**55.1+2.1**	**65.0+2.5**	**73.5+2.4**	**80.8+2.0**	**86.5+1.8**	**89.9+1.6**
8	29.8+0.8	43.5+1.4	55.5+1.9	65.6+2.4	73.9+2.3	81.4+1.9	86.9+1.6	90.3+1.4
9	29.8+0.8	43.7+1.4	55.8+1.8	66.1+2.3	74.4+2.3	81.7+1.8	87.2+1.5	90.7+1.3
10	29.8+0.7	43.8+1.3	56.1+1.8	66.3+2.3	74.6+2.3	81.9+1.7	87.3+1.4	90.9+1.2
11	29.8+0.7	43.9+1.3	56.3+1.8	66.8+2.3	74.7+2.2	82.1+1.7	87.5+1.4	91.0+1.1
12	29.9+0.7	43.9+1.2	56.5+1.8	66.9+2.2	74.9+2.1	82.3+1.6	87.5+1.3	91.1+1.0
49	29.9+0.7	44.6+0.9	57.5+1.3	68.1+1.5	76.2+1.8	82.7+1.5	87.7+1.2	91.9+0.8
Move distance	15.4+2.4	15.0+2.1	14.4+1.4	13.9+1.2	13.5+1.0	12.8+0.6	12.4+0.4	12.1+0.2
Orphan ratio	38.8+0.8	28.8+2.1	22.3+1.7	18.8+1.5	13.3+0.8	8.2+1.4	3.0+0.9	0.8+0.2

Table 7.4 R_c =100, R_s =50, moving base value 30 and 200 runs

We can get a better view of the improvement after the nodes are directed by neuro-fuzzy network controller through the simulator results. Figure 7.7 shows the pictures captured from the simulator. The deployment improved the coverage and uniformity of the WSN during iterations for 45 nodes with base movement value as 10. The left picture presents the initial coverage after dropped. The right picture gives the node position and coverage after 10 iterations. We can see that the coverage improved from 77.79% up to 96.02%. The initial drop had some high density spots with a number of nodes sitting together. After 10 iterations, the nodes were almost evenly distributed.



Coverage: 77.79%
Avg Dist: 0.00
☑ Sensing
☐ Communication
Initial deployment Sensing range=50,communication range=100

Coverage: 96.02%
Avg Dist: 4.14
☑ Sensing
☐ Communication
Deployment after iteration 10 Sensing range=50,communication range=100

Figure 7.7 Deployment progress after 10 iterations

From table 7.2, 7.3 and 7.4, it is obvious that the sensor coverage improved with more iteration. If the movement base value becomes bigger, the converging speed is accelerated, while the travel distance becomes bigger. With small moving distance base, the final coverage rate is usually higher than larger settings with more iteration and this shows that small moving base is slower in convergence but has better performance in improving the performance steadily. The larger moving base might cause the performance swinging from some iteration, especially when the node density is high. Although more iteration improves the coverage, more time and energy consumption will be needed. From table 7.2 to 7.4, the coverage generally goes up rapidly from iteration 1 to 8 or 9, from then on, the improvement speed becomes slower. With 8 or 9 iteration, the coverage rate is close to that of 49th iteration. The difference is less than 5%. In this setting, 10 iterations are enough for getting the optimal point.

Comparison of new rule set and rule set from fuzzy system based deployment paper [8_2]

RULE	FS Decision (Moving)	New Decision (Moving)
LD_low and AED_near	Medium	Far
LD_low and AED_medium	Near	Near
LD_low and AED_far	Near	Near
LD_med and AED_near	Far	Medium
LD_med and AED_medium	Medium	Medium
LD_med and AED_far	Near	Medium
LD_high and AED_near	Far	Far
LD_high and AED_medium	Medium	Far
LD_high and AED_far	Medium	Near

Table 7.5 Rule comparison

Comparison of results from new rule set and rule set from fuzzy system based deployment paper, FS [8_2].

Node number	250*250		500*500	
	New	FS	New	FS
15	0.959+0.035	0.935+0.047	0.447+0.011	0.433+0.012
20	0.984+0.016	0.975+0.016	0.582+0.016	0.563+0.024
25	0.991+0.011	0.983+0.016	0.692+0.029	0.666+0.028
30	0.997+0.003	0.994+0.007	0.771+0.042	0.739+0.046
35	1.000+0.001	0.999+0.002	0.824+0.019	0.789+0.019
40	1.000+0.000	1.000+0.001	0.894+0.028	0.860+0.029

Table 7.6 Performance comparison for different ROI

The new rule sets can combine the 3 rules for moving in medium range to one rule

IF LD is Medium THEN moving medium

The new rule set results are also compared with DSSA. The performance of the two approaches is pretty close to each other.

When mainly focusing on the coverage of the network, the experiment has found the optimal rule set, which is simpler and more efficient compared with those in referred paper. On the other hand, for the WSN deployment, the goal is not only keeping optimal coverage with given number of sensor node, but we also need to make sure to extend the lifetime of the WSN. To maximize the lifetime of the network, we need to reduce the node movement during the deployment. Thus we need to minimize the node moving distance during the process. The objectives about coverage and moving distance become competitive goal. Multi-objective method is proven to be efficient in finding the balance among competitive goals.

7.4.2 Pareto in finding the optimal coverage and moving distance with simple pruning

In reality, the problem of deployment of sensors is a multi-criteria optimization process. A number of important aspects have been discussed in former chapters that have to be considered for constructing sensor deployment rules.

In the proposed research we will use Pareto-based approach to address the needs of satisfying two objectives. A simple example of such situation is deployment of sensors that not only maximizes the coverage, but also maximizes the energy savings. One important criterion for the energy usage in mobile WSN is the moving distance during the deployment process. These two goals are actually contradict each other and the good way to balance these two goals is using multi-objective optimization approach.

Generally speaking, solving multi-objective problems is very difficult. In an attempt to stochastically solve problems of this generic class in an acceptable timeframe, specific multi-objective evolutionary algorithms were initially developed in the mid-1980s. The implementation of multi-objective genetic algorithm (MOGA) will be applied in this part of research. We use GA to train the neuro-fuzzy network models based on two competitive objectives, coverage and moving distance.

The fitness functions are still based on the average results on a number of separate WSN optimization. The fitness for coverage is same as the one used in GA optimization. The moving distance fitness is calculated as equation (5). In this way, the two fitness functions are all maximizing optimization problem. The chromosome length is 27, which is the count of the number of connections between each of the 9 AND neurons and each of the 3 OR neurons. The chromosome uses real number in *[0,1]* for the gene value.

The parameters for sensor nodes are as following, communication range R_c=100, sensing range R_s=50. The maximum iteration of the deployment is 10. The WSN with 15 nodes on 250 by 250 ROI was tested wit Pareto approach. The results are recorded in Figure 7.8 and Table 7.7.

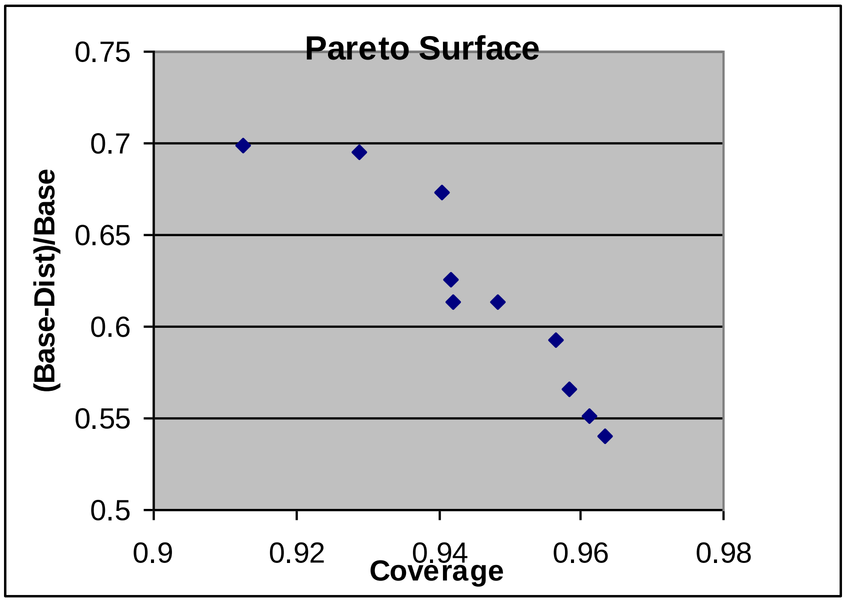


Figure 7.8 Coverage vs. distance for topology 250x250 for 15 sensors

	coverage		distance	
	average	std. dev.	average	std. dev.
Best coverage	0.974	0.008	44.71	3.47
Mid-way	0.961	0.012	38.18	3.99
Best distance	0.923	0.020	34.65	5.04

Table 7.7 Average coverage and distance for Parto-based algorithm

The tests made on 500 by 500 ROI for 40, 60 and 80 nodes are shown in Figure 7.9.

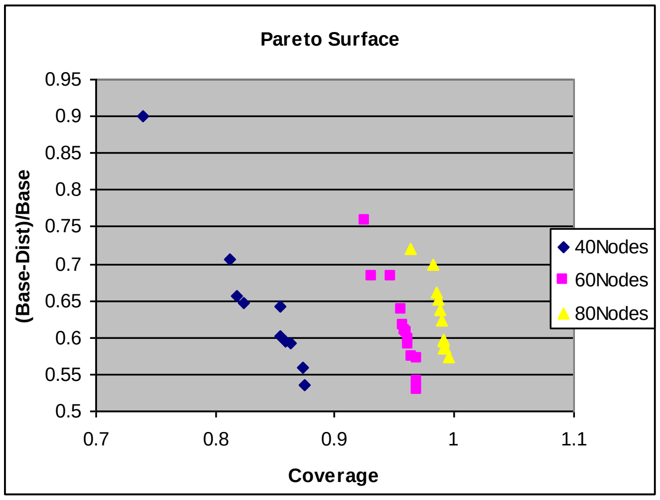


Figure 7.9 Coverage vs. distance for topology 500x500

From above Pareto experiments, this is a useful way to find a good fuzzy neural network structure to meet different goals of WSN performance.

7.4.3 Constructing FNN based FRBS using Pareto pruning to reduce complexity

The chromosome for MOGA based Pareto approach contains three parts. First part is the input attribute membership function indicator. The length for this part is

$$m \times n \tag{7}$$

where *m* is the number of input attributes, here equals to 2, *n* is the number of the AND neurons. The value range is from 1 to the number of fuzzy sets or discrete values. The second part is the weights to the connections for the inputs to AND neurons and those AND neurons to the OR neurons. The length for second part is

$$m \times n + m \times c \tag{8}$$

where *m* is the number of input attributes, here equals to 2, *n* is the number of the AND neurons, *c* is the number of OR neurons. The third part is two real number for storing the thresholds for the connections to AND neurons and OR neurons respectively. It is describe in figure 7.10.

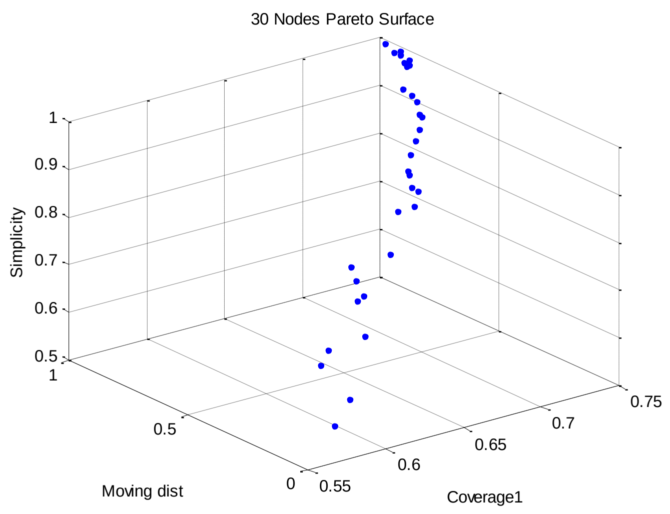
Integer index for ***m*×*n*** inputs from ***m*** attributes to ***n*** AND neurons	**Real numbers for *m*×*n*+*c*×*n* connecting weights for *m*×*n* input to AND neuron connections and c×*n* connections from *n* AND neurons to *c* OR neurons**	**Two real number for thresholds**

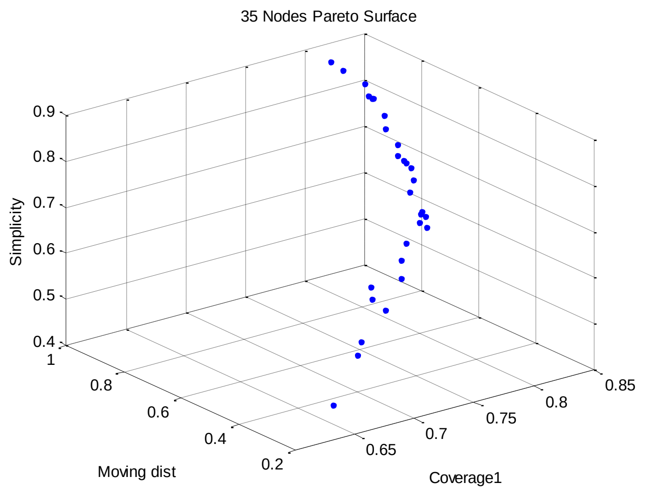
Figure 7.10 Chromosome structure for Pareto pruning FRBS on deployment control

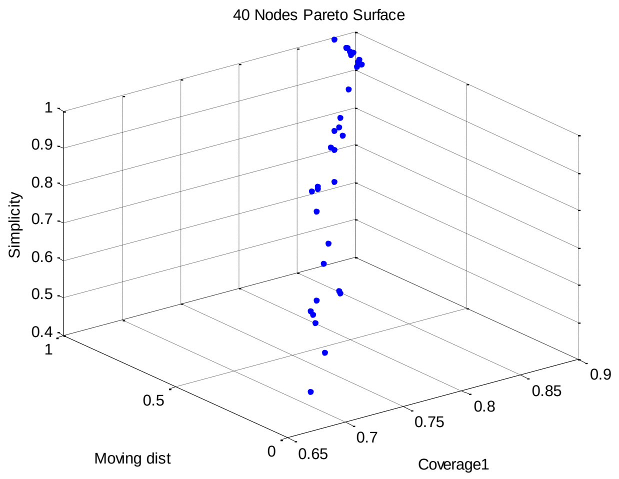
The values for the weight are from 0 to 1. The GA chromosome for Pareto training just contains two real number values in *[0,1]*, matching the thresholds for AND and OR neurons. The FNN complexity will be reduced at the same time of the construction of it. The Pareto is working on three competitive objectives, the coverage, moving distance and simplicity of the rules. The simplicity is the cut off number of input attributes vs. total number of input attributes among the rules. The moving distance objective is to maximize equation (6).

Experiments have been done for ROI of 500 by 500 units. The WSNs with 30, 35, 40 and 45 nodes were tested. The gained three dimensional Pareto surfaces are shown in figure 7.11.

-


30 Nodes Pareto Surface
Simplicity
1
0.9
0.8
0.7
0.6
0.5
Moving dist
1
0.5
0
Coverage1
0.55
0.6
0.65
0.7
0.75


35 Nodes Pareto Surface
Simplicity
0.9
0.8
0.7
0.6
0.5
0.4
Moving dist
1
0.8
0.6
0.4
0.2
Coverage1
0.65
0.7
0.75
0.8
0.85



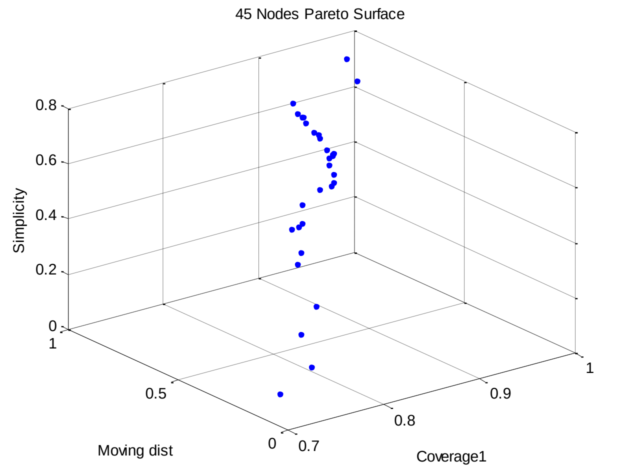


Figure 7.11 Pareto surface for 500 by 500 ROI

The Pareto surface evolution trend is shown in figure 7.12 for 40 nodes.

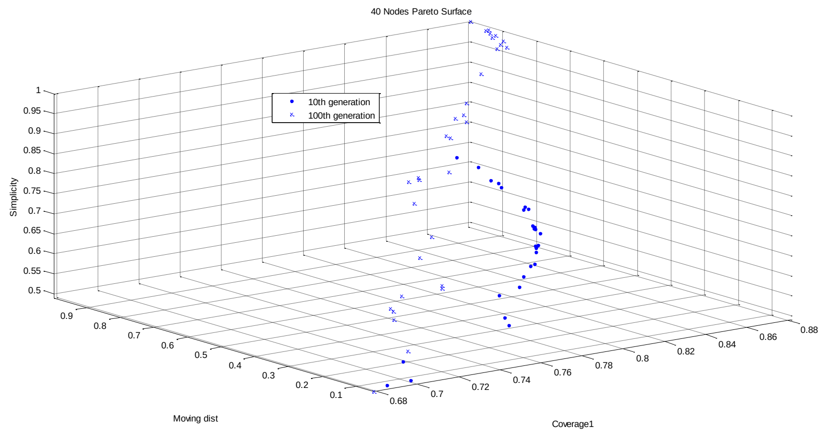


Figure 7.12 Pareto surface changes for 40 node WSN deployment

From figure 7.11 and 7.12, we can see that the resulting candidate solutions are smoothly distributed on the 3D Pareto surface. This gives us the chance to pick the optimal solution based on our needs. In order to maintain better coverage and maximize the WSN lifetime, we need to select the solution which brings reasonable trade off among the three objectives. After applied the thresholds onto the FNN and we got the new rule sets and the comparison of the rules sets are done in table 7.8 and 7.9.

RULE	FS	FNN + simple pruning	FNN + Pareto Pruning
LD_low and AED_near	Medium	Far	Near
LD_low and AED_medium	Near	Near	Near
LD_low and AED_far	Near	Near	Near
LD_med and AED_near	Far	Medium	Medium
LD_med and AED_medium	Medium	Medium	Medium
LD_med and AED_far	Near	Medium	Medium
LD_high and AED_near	Far	Far	Far
LD_high and AED_medium	Medium	Far	Medium
LD_high and AED_far	Medium	Near	Medium

Table 7.8 Rule comparison for the three methods

Comparison of results from new rule set and rule set from fuzzy system based deployment paper, FS [8_2].

Node number	250*250			500*500		
	Simple	FS	Pareto	New	FS	Pareto
15	0.959+0.035	0.935+0.047	0.951+0.066	0.447+0.011	0.433+0.012	0.432+0.017
20	0.984+0.016	0.975+0.016	0.985+0.009	0.582+0.016	0.563+0.024	0.575+0.012
25	0.991+0.011	0.983+0.016	0.989+0.021	0.692+0.029	0.666+0.028	0.670+0.006
30	0.997+0.003	0.994+0.007	0.996+0.003	0.771+0.042	0.739+0.046	0.752+0.027
35	1.000+0.001	0.999+0.002	1.000+0.000	0.824+0.019	0.789+0.019	0.799+0.022
40	1.000+0.000	1.000+0.001	1.000+0.000	0.894+0.028	0.860+0.029	0.901+0.021

Table 7.9 Performance comparison for different ROI for three methods

From table 7.8 and 7.9, the new rule set obtained through Pareto pruning is even simpler. The performance on coverage of the three methods is pretty similar. With almost same performance matrix, the simpler rule set will give more convenient to understand the logic of the deployment control and shorter processing time on the sensor nodes when actually moving the nodes in deployment process. The performance of the three methods is also comparable with ref [8_1].

7.5 Conclusions

In most of the rule based systems, the rules are provided by the experts in the problem area. It is same for the rule set in the referred. The rules sometimes are not the optimal ones. Using the rule construction approach, we can examine a large amount of possibilities and obtain the optimal rule set with simpler rules. With the calculated Pareto surface, we can choose the proper model to meet the focusing objectives.

Reference

[8_1] N. Heo, and P.K. Varshney, “A distributed self spreading algorithm for mobile wireless sensor networks,” Wireless Communications and Networking, 2003. WCNC 2003. IEEE, pp. 1597-1602 vol.1593, 2003.

[8_2] H. Shu, Q. Liang, and J. Gao, “Distributed Sensor Networks Deployment Using Fuzzy Logic Systems,” International Journal of Wireless Information Networks, vol. 14, no. 3, pp. 163-173, 2007.

Chapter 8 Data modeling using fuzzy c-means

Previous chapters focused on the construction and complexity reduction of FRBSs using FNN (Chapter 6) and on the real world application of FRBS for the problem of the deployment of sensors in the wireless sensor network (Chapter 7). In this chapter, we will address the issue of construction and complexity reduction of FRBS using fuzzy c-means (FCM) clustering approach.

In the design of fuzzy rule-based models we strive to develop models that are accurate and interpretable (transparent). The approach proposed here is aimed at the enhancement of transparency of the fuzzy model already constructed with the accuracy criterion in mind by proposing two modifications to the rules. First, we introduce a mechanism of reduction of the input space by eliminating some less essential input variables. This results in rules with the reduced subspaces of input variables making the rules more transparent. The second approach is concerned with an isolation of input variables: fuzzy sets defined in the n-dimensional input space and forming the condition part of the rules are subject to a decomposition process in which some variables are isolated and interpreted separately. The reduced dimensionality of the input subspaces in the first approach and the number of isolated input variables in the second one are the essential parameters controlling impact of enhanced transparency on the accuracy of the obtained fuzzy model. The two problems outlined above are of combinatorial character and the optimization tasks emerging there handled with the use of Genetic Algorithms. A series of numeric experiments is reported where we demonstrate the effectiveness of the two approaches and quantify the relationships between the criterion of accuracy and interpretability.

The concept of interpretability of fuzzy rule-based models has been around for several decades and attracted a significant deal of attention. The transparency of fuzzy models is one of the outstanding and important features of fuzzy models. In contrast to the criterion of accuracy, whose quantification is relatively straightforward and easy to come up with performance indexes, transparency of fuzzy rules is more difficult to describe. What makes the fuzzy rule-based easier to interpret is still an open issue. It is quite subjective to assess and in one way or another invokes a factor of subjective judgment given that a human user is ultimately involved in the evaluation process. What become very much apparent are a multifaceted nature of the problem and a multitude of various approaches supported by various optimization technologies including evolutionary optimization. When it comes to the main factors worth considering when discussing a concept of interpretability, we can enumerate a list of factors that may be involved in the reduction process:

- number of rules forming a rule base of the model,
- number of sub conditions (input variables) forming a condition part of a given rule,
- number or rules and the number of input variables,
- complexity of local regression models (in case of Takagi-Sugeno model)
- interpretability of a family of fuzzy sets formed in the input space for individual variables

As a result, given this diversity of possible ways of reduction of rules, it is difficult to quantify the effect of reduction. For instance, it is not always clear if it would be better to have a larger number of simple rules (whose condition parts are linear functions) or a smaller number of rules of a more complex conclusion parts (say, those of polynomial form).

8.1. Approaches presented in literature

In the construction of fuzzy rule based system, we have been witnessing a wealth of design strategies and detailed algorithms involving the technology of Evolutionary Computing and neurocomputing. Just recent developments reported in this realm can be found in a series of studies [7_1, 7_2, 7_3, 7_5, 7_6, 7_7]. Predominantly, the development of fuzzy models is guided by the criterion of accuracy. Another fundamental criterion being at the heart of fuzzy modeling is interpretability (transparency) of resulting fuzzy models. This criterion is central to fuzzy models however its multifaceted nature requires a thorough formulation, quantification of essential aspects of interpretability and subsequently calls for advanced optimization techniques supporting the realization of the ensuing design.

The reader may refer to a large body of studies devoted to the issue of interpretability, cf. [7_3, 7_9]. Quite often given a combinatorial nature of the reduction problems, Genetic Algorithms are used in the process, see [7_8]. There are also more specialized algorithmic vehicles to support the reduction process such as e.g., singular value decomposition [7_10].

As to the overall strategy of rule reduction and interpretability enhancement, there are two general design strategies: either the reduction is completed once the model has been constructed or the reduction aspect of rule-based modeling is incorporated into the design process from its very beginning.

The objective of the study is to pursue a fundamental issue of building more transparent and user-centric fuzzy rule-based models by starting from an already designed fuzzy model. The intent is to make the rules more readable in two different ways: (a) by reducing the input space (the number of antecedents) of the individual rules, and (b) by isolating input variables completed for the input variables treated en block in the condition parts of the rules.

In both these fundamental scenarios we can establish and quantify a tradeoff between a gradual reduction of accuracy (which is inevitable when realizing any of the reduction mechanisms of the rules and enhancing its intensity) and the increased interpretability of the rules.

8.2 Fuzzy rule-based models based on FCM

Our point of departure of the reduction processes of the rules is a “standard” Takagi-Sugeno fuzzy model comprising “c” rules coming in the following form

$$\text{-if } \mathbf{x} \text{ is } A_i \text{ then } y = f_i(\mathbf{x}, \mathbf{a}_i) \tag{1}$$

i=1,2, .., c where $\mathbf{x} \in \mathbf{R}^n$. A_i is a fuzzy set defined in the n-dimensional space while f_i is the corresponding local model (linear or nonlinear) endowed with its parameters $\mathbf{a}_i$ forming the conclusion part of the i^{th} rule. The design of such models is well reported in the literature and as usual consists of the two main steps, namely (a) a construction of condition parts (through clustering of input data done in the input space) and (b) estimating parameters of the linear models (which leads to the problem of linear regression). The number of rules (c) is determined by monitoring the behaviour of the model on the training and testing data. The rules in the form (1) come with several essential properties. The condition part is a fuzzy set expressed in $\mathbf{R}^n$ and this format is concise and helps avoid a curse of dimensionality we are commonly faced with in rule based systems with a higher number of input variables. The number of rules is rather small and the rule base is compact. Unfortunately, the interpretability could be negatively impacted as no individual variables in the condition part are treated and visualized separately.

When it comes to the quantification of the accuracy of the fuzzy model (1) its performance is expressed by the RMSE index computed for the training set

$$\sqrt{\frac{1}{N}\sum_{k=1}^{N}(\mathrm{FM}(\mathbf{x}_k)-\mathrm{target}_k)^2}$$

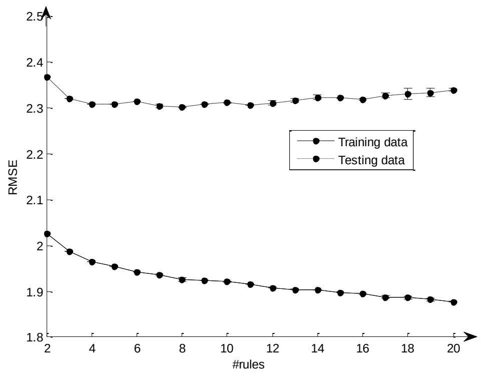
(2)

where N denotes the number of data in the training set. In the same way, quantified is the performance of the constructed model on the testing data (consisting of M data points)

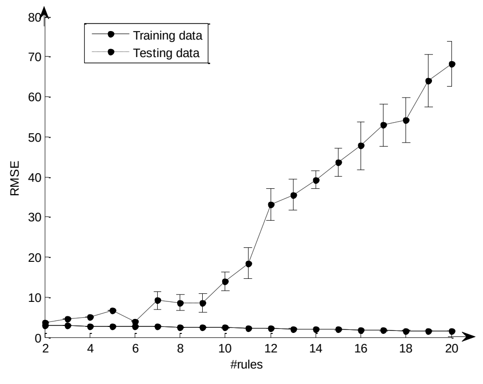
$$\sqrt{\frac{1}{M}\sum_{k=1}^{M}(\mathrm{FM}(\mathbf{x}_k)-\mathrm{target}_k)^2}$$

(3)

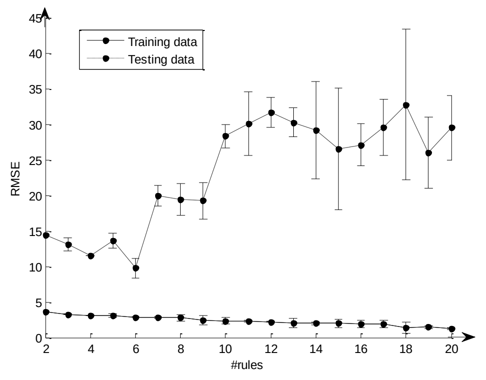
Proceeding with the data summarized in Section 1, the performance of the corresponding models visualized versus the number of rules (c) is illustrated in a series of figures shown below, Figure 7_1. The results are reported both for the training and testing data. In the design, we use a standard version of the Fuzzy C-Means (FCM) [4] with the fuzzification coefficient (m) set to 2.0. The algorithm was run for 10 iterations (more specifically, we completed 10 time runs with different training/testing data splits)



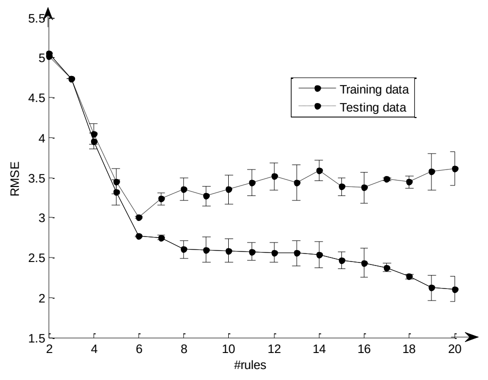
(a) Abalone



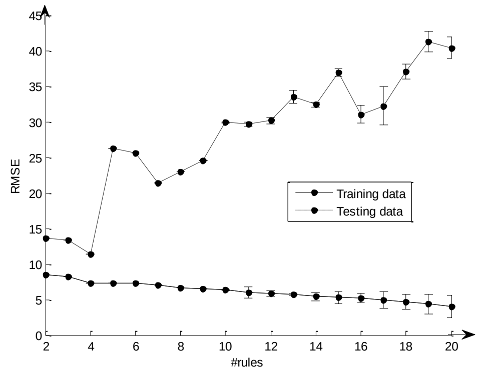
(b) Auto MPG



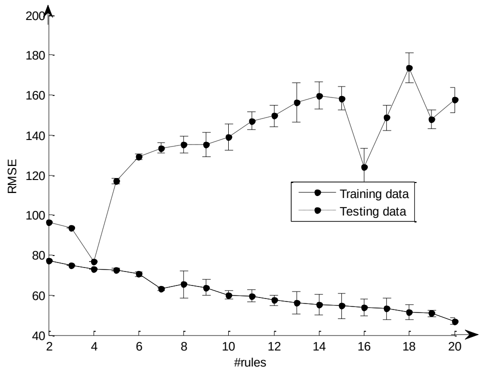
(c) Boston Housing



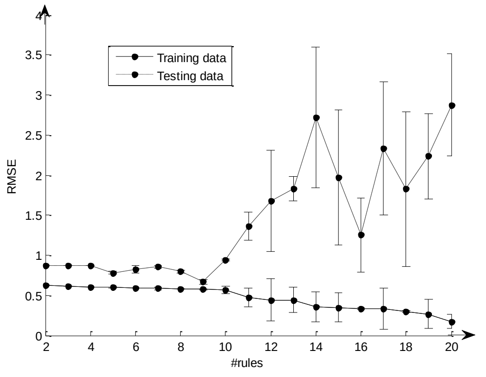
(d) Computer Activity



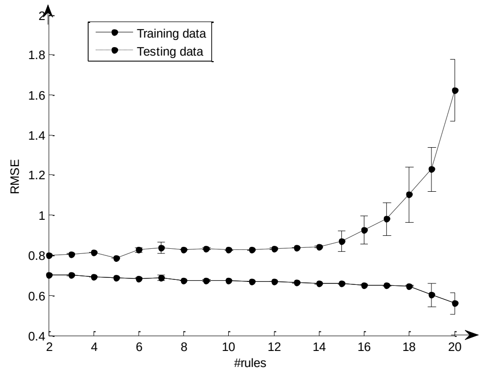
(e) Concrete Strength



(f) Forest Fires



(g) Red Wine Quality



(h) White Wine Quality

Figure 8.1 Performance of fuzzy models versus the number of rules reported for the training and testing data

For the training sets, there is a general tendency of lower values of the RMSE with the increase of the number of rules. The performance reported on the testing sets points at the memorization effect where the some models tend to lose their generalization capabilities. By eyeballing these plots, we choose a suitable number of rules (clusters) where sound approximation abilities come hand in hand with the generalization of the models. The values selected in this way are collected in Table 8.1.

Data name	Selected number of rules
Abalone	7
Auto MPG	6
Boston Housing	6
Computer Activity	6
Concrete strength	4
Forest fires	4
Red wine quality	9
White wine quality	5

Table 8.1 Number of rules of fuzzy models constructed for the corresponding data

After the FRBS is constructed with FCM, the rule complexity reduction methods will be applied to simplify system. Here two methods in complexity reduction will be presented, reduction of the input spaces and isolation of the input variables.

8.3 Reduction of input subspaces

Reduction of input spaces method is going to cut off the variables in the input space to meet a certain threshold, defined as the percentage of the keeping variables in the total number of variable among the whole rule base.

8.3.1 Algorithm description

The essence of the enhancement of interpretability of the rules is accomplished by reducing the number of input variables standing in the condition parts of the rules. The reduced rules are concisely described in the form

-if **x** is $[A_i]_{X_i}$ then $y= f_i(\mathbf{x}, \mathbf{a}_i)$

(4)

where the symbol $[\ ]_{X_i}$ stresses the fact that the fuzzy set A_i is now effectively confined to the reduced input space $\mathbf{X}_i \subset \mathbf{R}^n$ where some original input variables have been removed. In other words, dim $(\mathbf{X}_i) = n_i < n$.

The computing of the activation level of A_i positioned in this new reduced space is realized as follows

$$[A_i]_{\aleph_i} = 1/\sum_{j=1}^{c}\left(\frac{\|\underline{\mathbf{x}} - \underline{\mathbf{v}}_i\|_{\aleph_i}}{\|\underline{\mathbf{x}} - \underline{\mathbf{v}}_j\|_{\aleph_i}}\right)^{\frac{2}{m-1}}$$

(5)

i=1,2,.., c; m>1 where the computations of the distance are realized in the reduced input space $\mathbf{X}_i$.

The reduction of the rules can be quantified in terms of a reduction factor ν, which relates with the number n*c (expressing an overall number of variables across all the rules) in the following way

$p = \nu(n*c)$

(6)

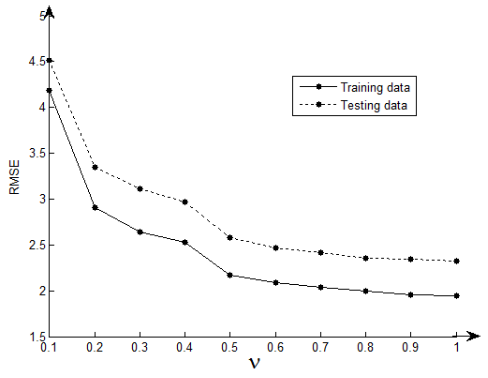
The reduction of the input variables present in the new, more interpretable rules is done by engaging the optimization capabilities genetic algorithms (GAs). More specifically, we optimize a matrix of allocation of input variables W = $[w_{ij}]$ with “c” rows and “n” input variables. The “p” largest entries are selected giving rise to a 0-1 matrix where these entries with these largest values are set to 1 while the remaining ones are suppressed to zero. Each row of the matrix formed in this way identifies the variables to be used in the corresponding reduced rule. If all entries of the i^{th} row of W are equal to zero, this entails that the corresponding rule does not exist in the reduced set of rules.

The fitness function of the GA used in the optimization is the RMSE computed for the training set $\sqrt{\frac{1}{N}\sum_{k=1}^{N}(\mathrm{FM}(\mathbf{x}_k)-\mathrm{target}_k)^2}$ where FM(.) is described by (4). In the same way quantified is the performance of the constructed model on the testing data, that is we determine the value of $\sqrt{\frac{1}{M}\sum_{k=1}^{M}(\mathrm{FM}(\mathbf{x}_k)-\mathrm{target}_k)^2}$. As an optimization vehicle, we use a standard real coded GA. The chromosome incorporates the 0-1 matrix of allocation of variables while the fitness function is expressed with the use of the RMSE value, namely 1/(1+ RMSE) (evidently the maximization of the fitness function of this form results in the minimization of the RMSE).

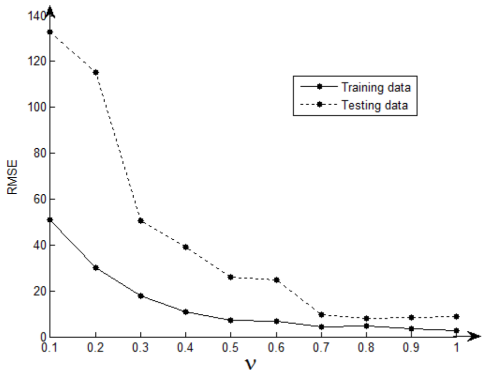
8.3.2 Experimental studies

In the following experiments we present how the reduction of the rules proceeds and how the reduced, more interpretable rules perform. We use different values of ν and report the corresponding values of the RMSE for the training and the testing data. The GA used a population of 100 individuals and was run up to 100 generations. The crossover rate was set to 0.8 while the mutation rate was equal to 0.1. The choice of these numeric values was a result of some experimentation.

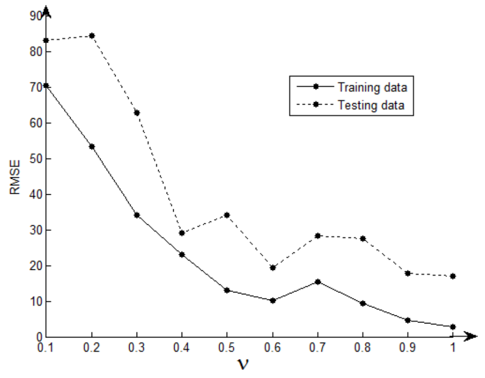
The results are quantified by reporting the RMSE values obtained for different values of the reduction index; refer to Figure 8.2.



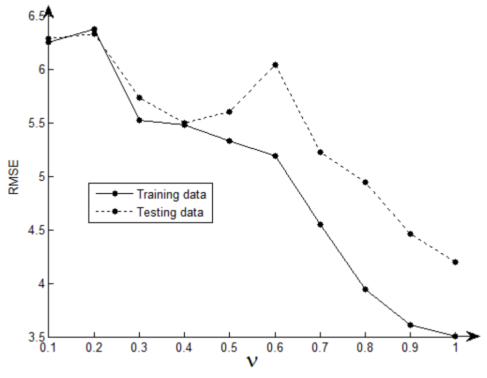
(a) Abalone



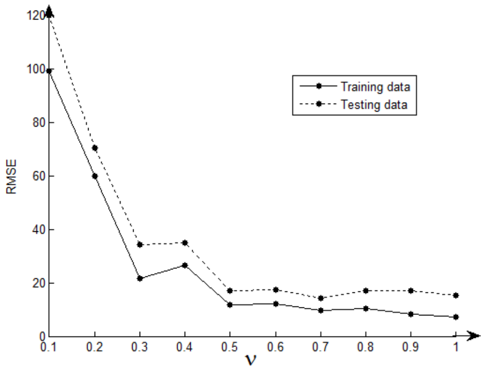
(b) Auto MPG



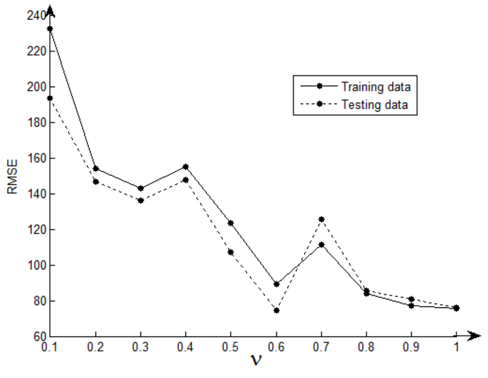
(c) Boston Housing



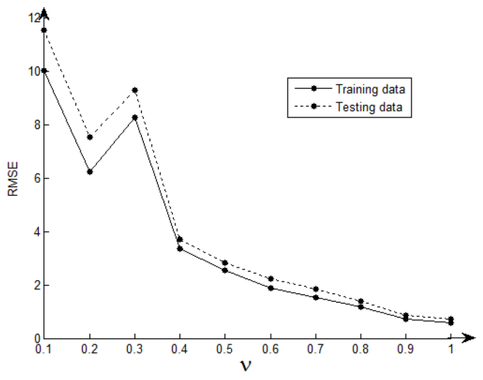
(d) Computer Activity



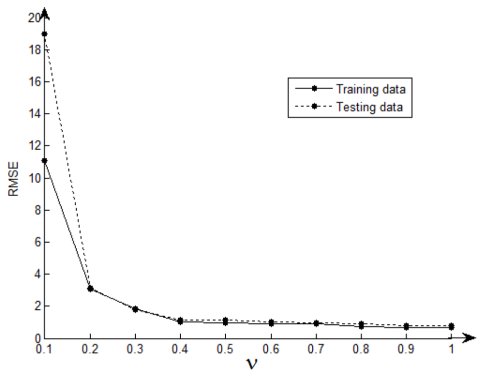
(e) Concrete Strength



(f) Forest Fires



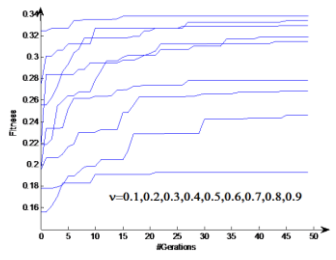
(g) Red Wine Quality



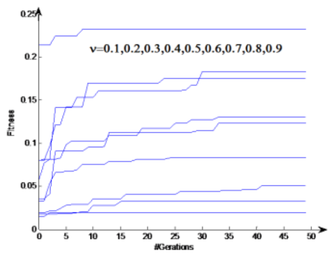
(h) White Wine Quality

Figure 8.2 RMSE values of the reduced rule-based models vs reduction level ν

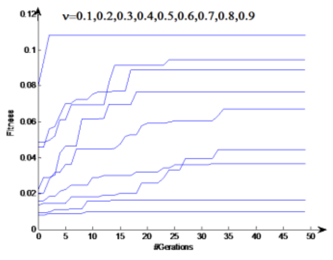
It becomes apparent (and intuitively anticipated) that lower values of ν result in higher RMSE values. The detailed behaviour varies across data with regard to how far the rules can be reduced and how the differences shape up for the training and testing data. For example, the reduction could be made quite substantial not compromising the performance of the model as this becomes present in case of abalone, auto, concrete, and white wine. In some case, we witness a phenomenon of increased generalization abilities of the model (lower differences of the RMSE for the training and testing data for lower values of ν). Figure 8.3 illustrates the performance of the GA for some selected data; most of the improvement is visible at the beginning of the optimization (first 20-30 generations).



(a) Abalone



(b) Auto MPG

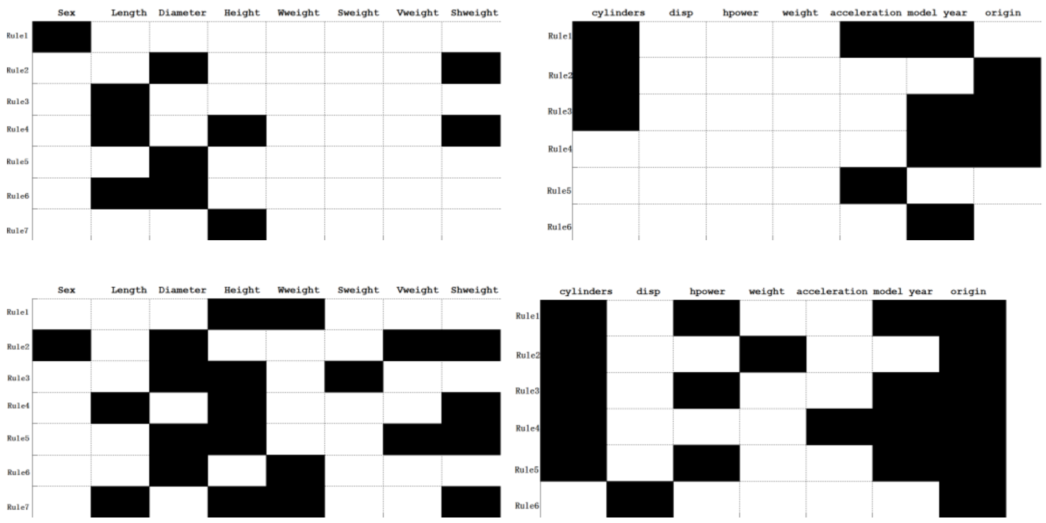


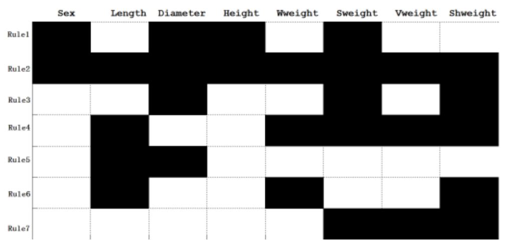
(c) Concrete

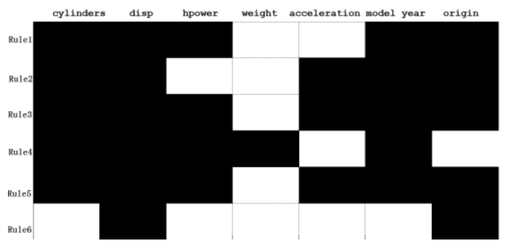
Strength

Figure 8.3 Values of fitness function reported in successive GA generations for ν ranging from 0.1 to 0.9 with step 0.1

The detailed results of reduction of the number of variables in the rules are contained in Figure 8.4. The shaded regions identify the input variables being retained in the corresponding rules. This offers a better view as to which input variables can be dropped and points at a sequence of the variables, which have been successively eliminated.

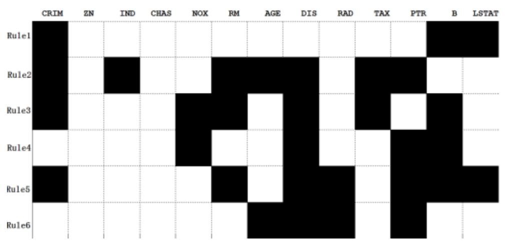


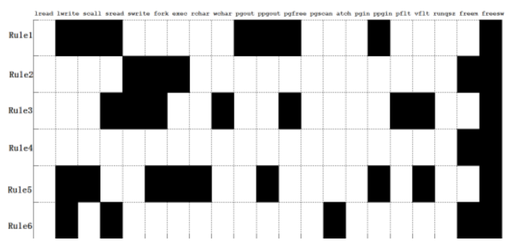


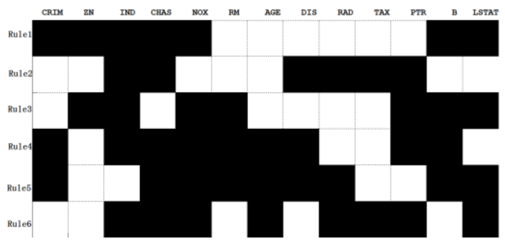


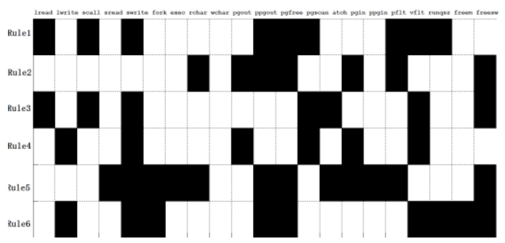
(a) Abalone; ν=0.2, 0.4, 0.5

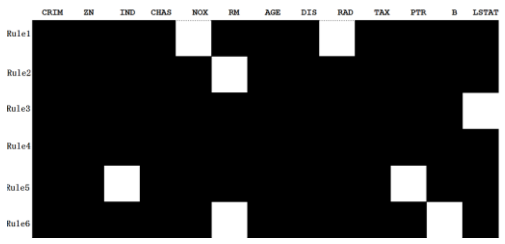
(b)Auto MPG; ν=0.3, 0.5, 0.7

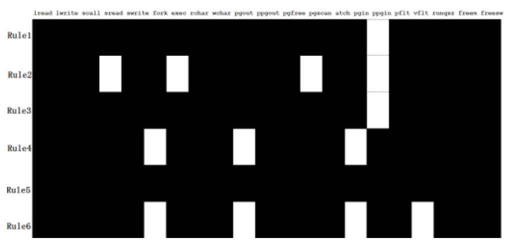




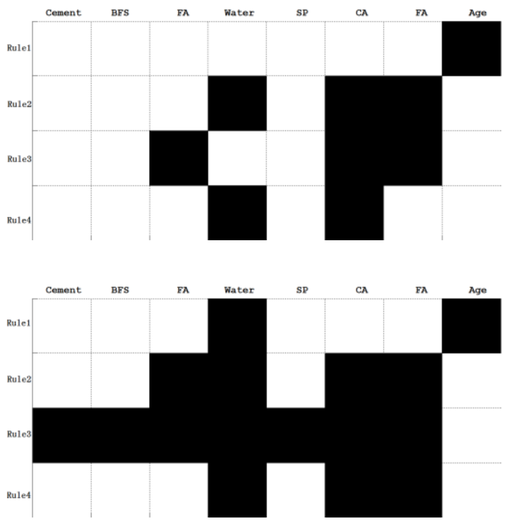


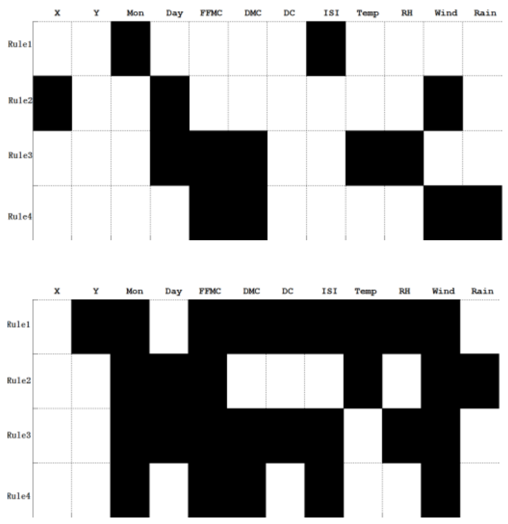


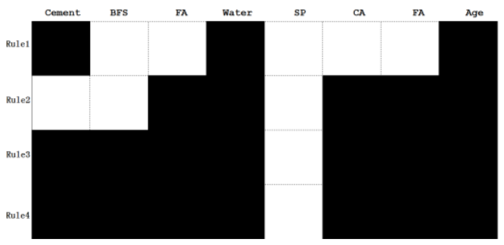




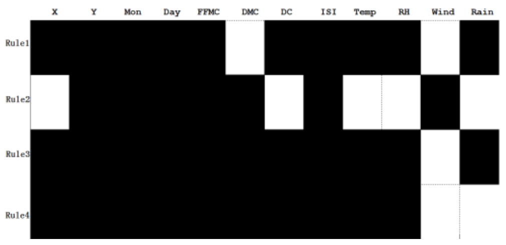
(c) Boston Housing; ν=0.4, 0.6, 0.9 (d) Computer Activity; ν=0.3, 0.4, 0.9



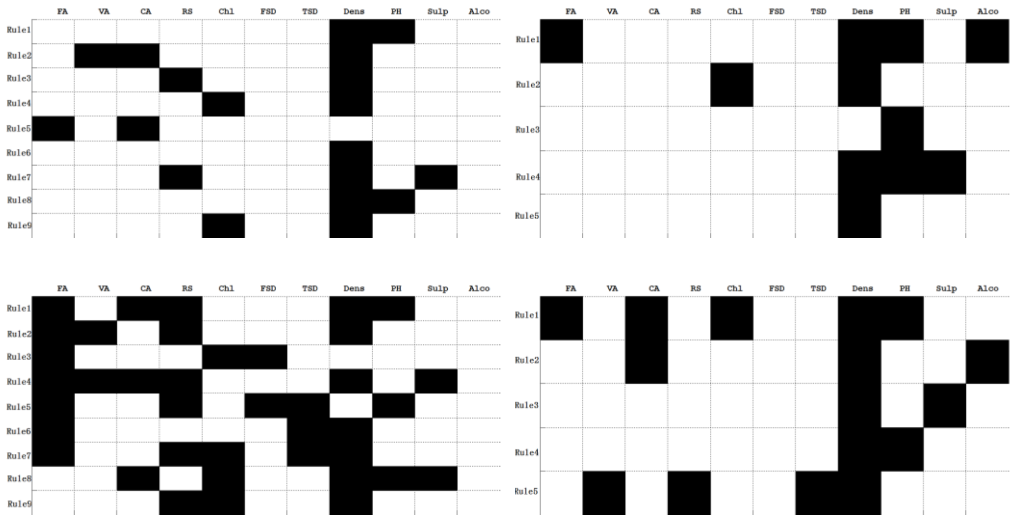


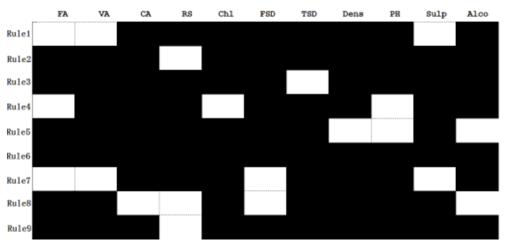


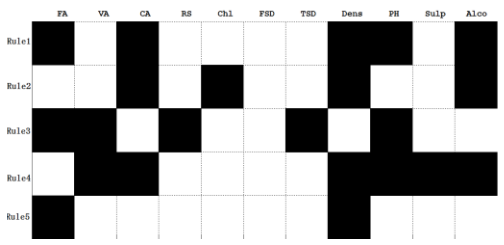
(e) Concrete Strength ν=0.3, 0.5, 0.7



(f) Forest Fires ν=0.3, 0.6, 0.8







(g) Red Wine Quality; ν=0.2, 0.4, 0.8 (h) White Wine Quality; ν=0.2, 0.3, 0.4

Figure 8.4 Visualization of reduced rules: shaded regions identify the input variables being retained

8.4 Isolation of input variables

Isolation of input variables method finds the variables which can be isolated in the given rule. With a defined value for the number to isolation of input variables, the rules will become simpler.

8.4.1Algorithm

To enhance the transparency of the rules, we express the fuzzy set A_i as a Cartesian product of a single *isolated* fuzzy set defined in **R** and a *relational* remainder expressed in $\mathbf{R}^{n-1}$. In other words, we form the expression describing the condition part as follows

$$A_i^{\wedge}(x_j) \times A_i^{\sim}(\mathbf{x}^{\sim}) \tag{7}$$

where $A_i^{\wedge}$ is a fuzzy sets defined in **R** and $A_i^{\sim}$ is expressed in $\mathbf{R}^{n-1}$. Then the rules read as follows

-If x_j is $A_i^{\wedge}(x_j)$ and $\mathbf{x}^{\sim}$ is $A_i^{\sim}(\mathbf{x}^{\sim})$ then y is $f_i(\mathbf{x}, \mathbf{a}_i)$ (8)

Here a certain input variable (j^th one) has been selected to be isolated. The term isolation pertains to the fact that a certain variable has been chosen and subsequently a fuzzy set isolated from the fuzzy set A_i is treated separately and thus becomes more visible and interpretable.

Obviously, the above Cartesian product is not identical to the original A_i that is the following holds

$$A_i \neq (A_i^{\wedge} \times A_i^{\sim}) \tag{9}$$

In terms of membership functions this means that

$$A_i(\mathbf{x}) \neq \min(A_i^{\wedge}(x_j), A_i^{\sim}(\mathbf{x}^{\sim})) \tag{10}$$

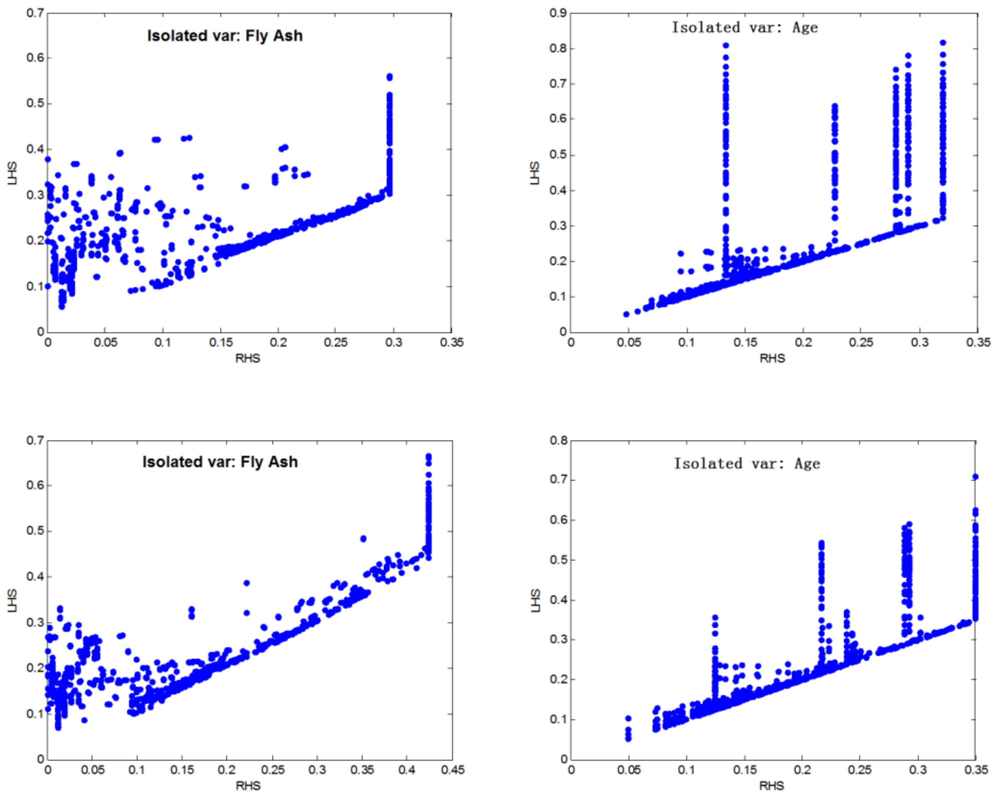
Note that the corresponding membership functions of $A_i^{\wedge}(x_j)$ and $A_i^{\sim}(\mathbf{x}^{\sim})$ are computed as follows

$$A_i^{\wedge}(x_j) = \frac{1}{\sum_{l=1}^{c}\left(\frac{x_j - v_{ij}}{x_j - v_{il}}\right)^{2/(m-1)}} \tag{11}$$

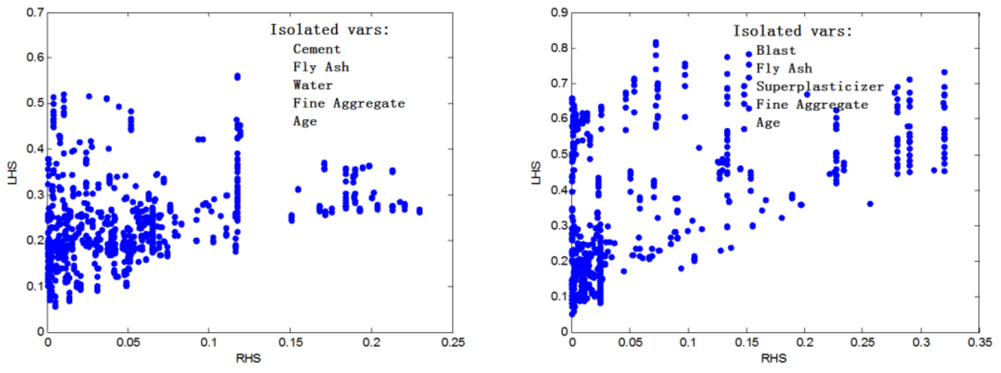
$$A_i^{\sim}(x^{\sim}) = \frac{1}{\sum_{l=1}^{c}\left(\frac{\| \mathbf{x}^{\sim} - \mathbf{v}_i^{\sim} \|}{\| \mathbf{x}^{\sim} - \mathbf{v}_l^{\sim} \|}\right)^{2/(m-1)}} \tag{12}$$

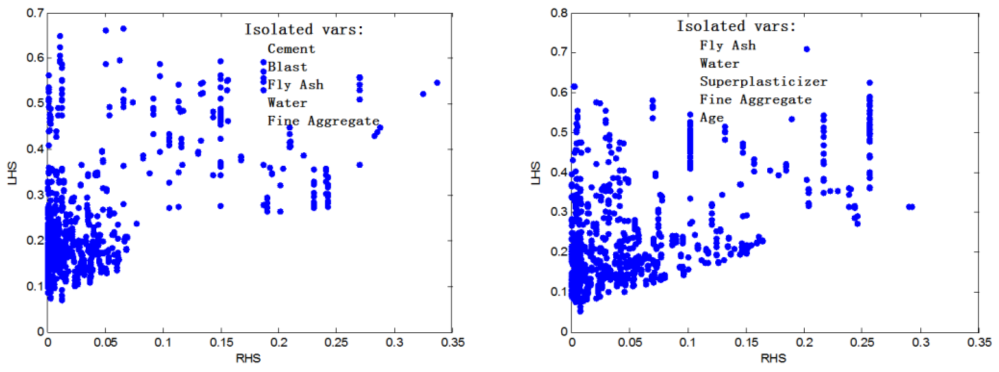
The consequence is that if $A_i^{\wedge}(x_j) \times A_i^{\sim}(\mathbf{x}^{\sim})$ is used as the condition part of the i-th rule, the output of the model is going to be different than the original rule. It is likely that the accuracy of the model could be reduced as a result of the increased interpretability of the rules because of the isolation of the input variables. In the above formulation, one is interested in choosing an individual variable (j) for which the results provided by the rule-based model are as close as possible to those formed by the original fuzzy model. The selection of the input variable is quite straightforward through a direct enumeration.

The plots showing the LHS and the RHS relationship is shown for concrete strength with 1 or 5 isolated input variables.



(a) single input variable isolated





(b) 5 isolated input variables

Figure 8.5 Values of the original activation of the rules value vs. the one with isolated input variables - concrete strength data set, 4 rules in the rule base

From the plots above, several observations of a general character can be drawn. First, the values of the LHS are higher than the corresponding ones for the LHS. This is reflective of the fact that the separation of the variable(s) leads to the higher activation levels of the rules with eventual reduction of the specificity of the results of reasoning. It is also apparent that with the increase of the variables being isolated – compare Fig.5 (a) and (b), the differences between the values produced by the LHS and RHS of the expression (9) are more profound. Again, this is not surprising as by isolating more variables we depart from the RHS more vigorously.

In a general setting, one can realize an isolation of “L” input variables, which as a result leads to the rules in the form

-If x_{j1} is $A_i^{\wedge}$ (x_{j1}) *and* x_{j2} is $A_i^{\wedge}$ (x_{j2}) *and…and* x_{jL} is $A_i^{\wedge}$ (x_{jL}) *and* $\mathbf{x}^{\sim}$ is $A_i^{\sim}(\mathbf{x}^{\sim})$ then y is $f_i(\mathbf{x}, \mathbf{a}_i)$

(13)

note that in this case here $\mathbf{x}^{\sim}$ is defined in $\mathbf{R}^{n-L}$.

Here the choice of L variables gives rise to a combinatorial optimization problem. This could be solved by GA optimization. Considering that the value of L is specified in advance, GA forms an optimal isolation matrix **I** consisting of “c” rows (number of rules) and “n” rows (number of input variables) where in each row there are L 1s indicating the variables which are isolated in the rule. For instance, for L =3 the matrix with the entries

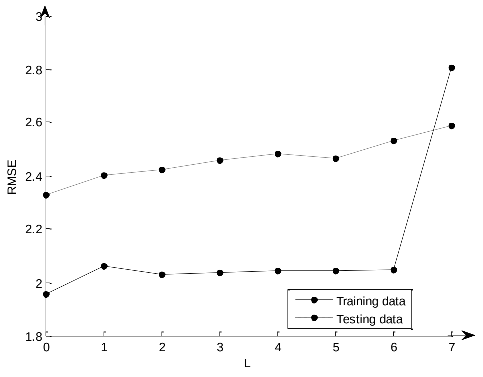
$$\mathbf{I}=\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ \ldots \end{bmatrix} \tag{14}$$

states that in the first rule isolated are variables 1, 4, and 5; in the second rule we isolate variables 2,3, and 4, etc…

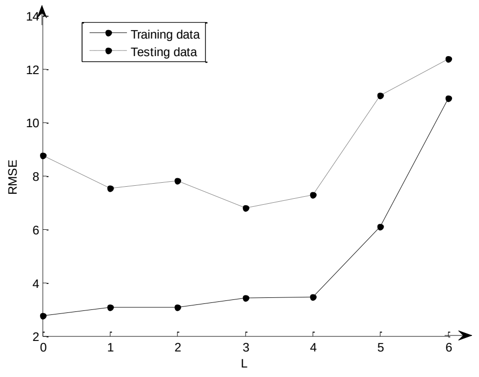
8.4.2 Experiments

The GA was carried out with 100 populations and maximum 50 generations. Some GA stop improvement before 50th generation, so the maximum generation is set to 50. The population is not large, so the moderate crossover and mutation rate are used. The crossover rate is set as 0.8 and mutation rate is 0.1.

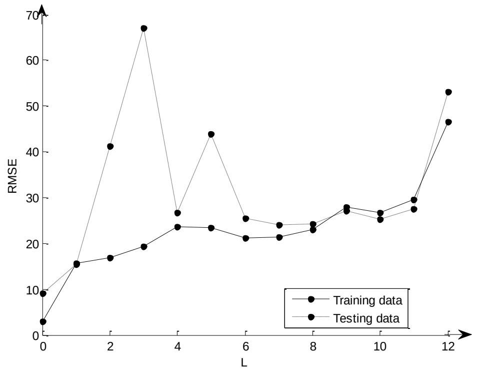
We present the results in a similar way as before by focusing on the presentation of the rules with isolated variables and showing how the families of isolated variables impact the performance of the model



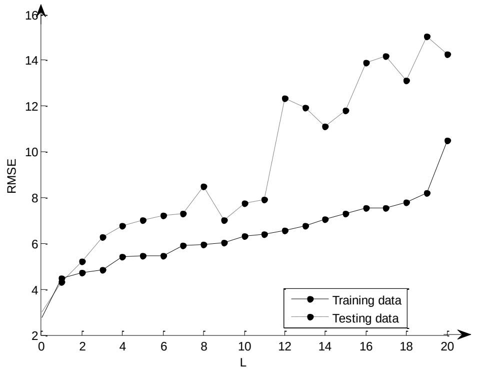
a) Abalone



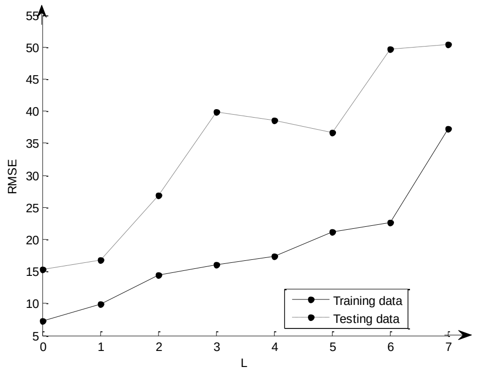
b)Auto MPG



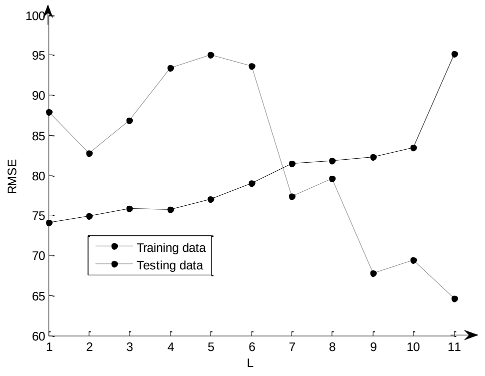
c) Boston Housing



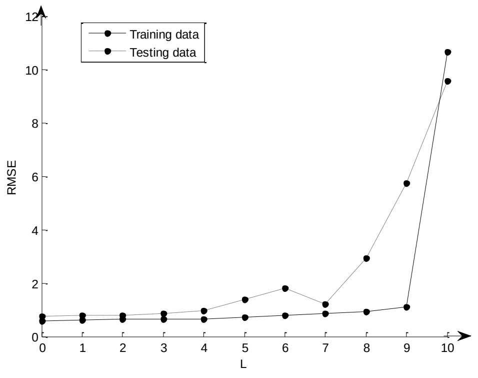
d) Computer Activity



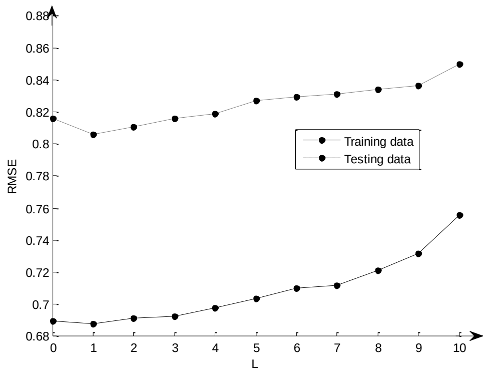
e) Concrete Strength



f) Forest Fires



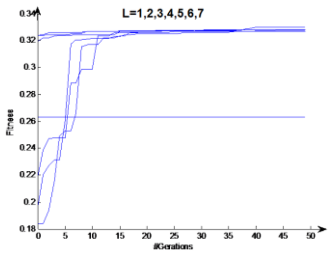
g) Red Wine Quality

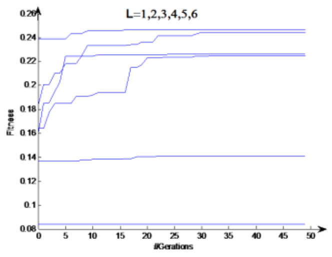


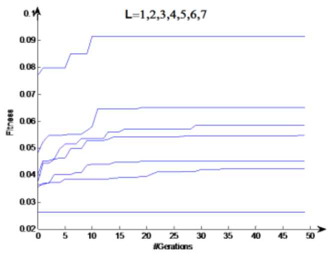
h) White Wine Quality

Figure 8.6 Performance of the fuzzy model versus the number of isolated input variables

GA performance plots

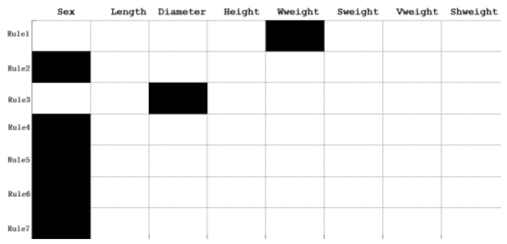


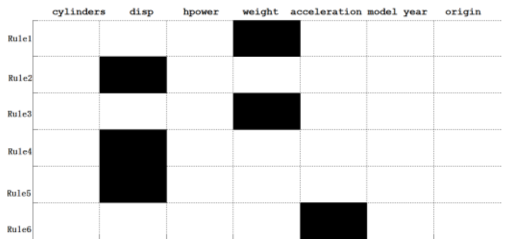


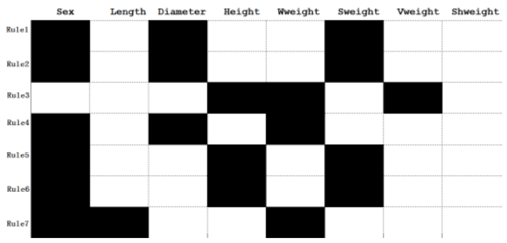


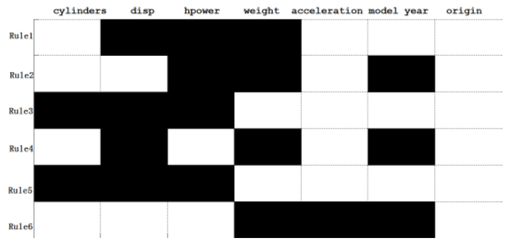
(a) Abalone (b) Auto MPG (c) Concrete Strength

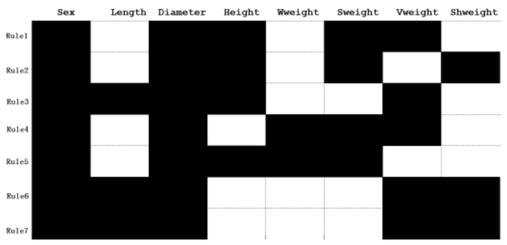
Figure 8.7 Fitness function reported in successive GA generations

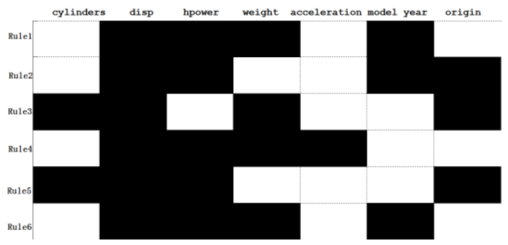




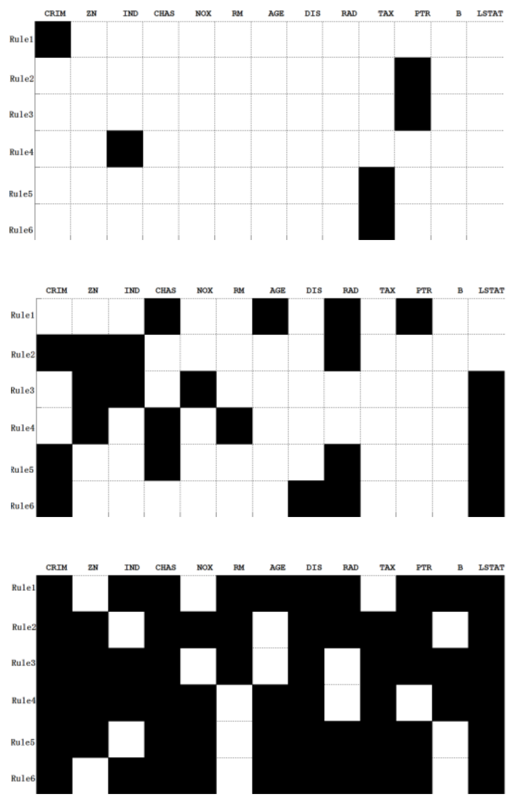


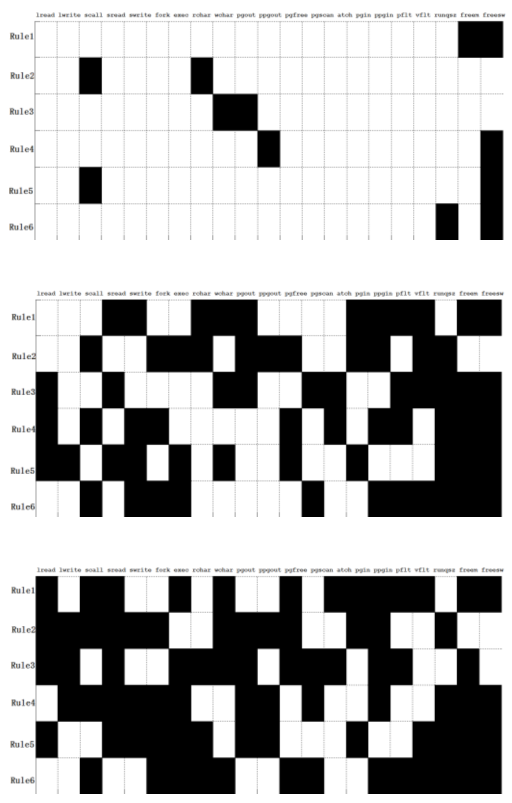




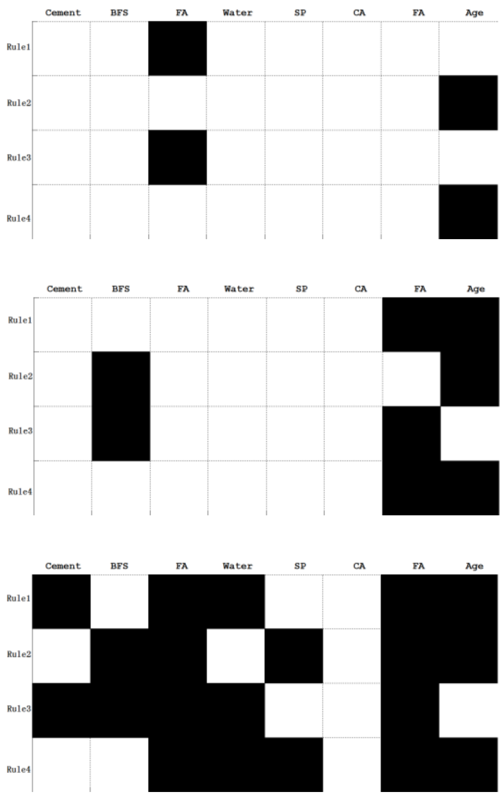


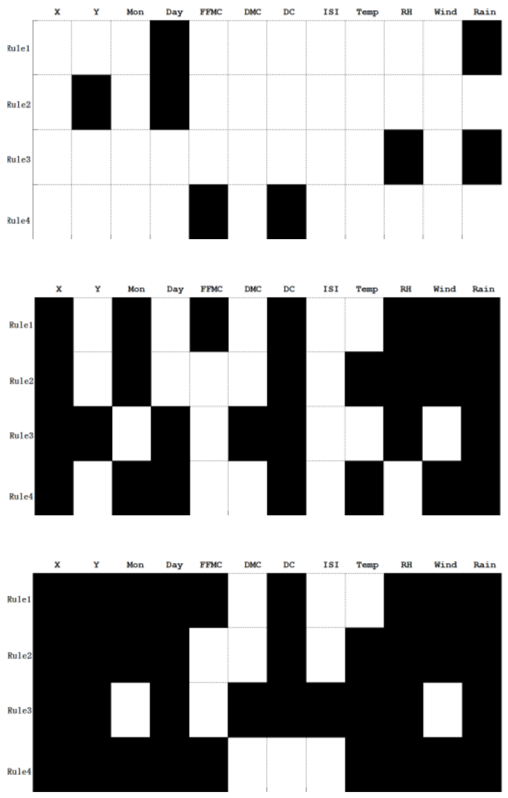
a) Abalone with L= 1, 3, 5 b)Auto MPG with L= 1, 3, 4



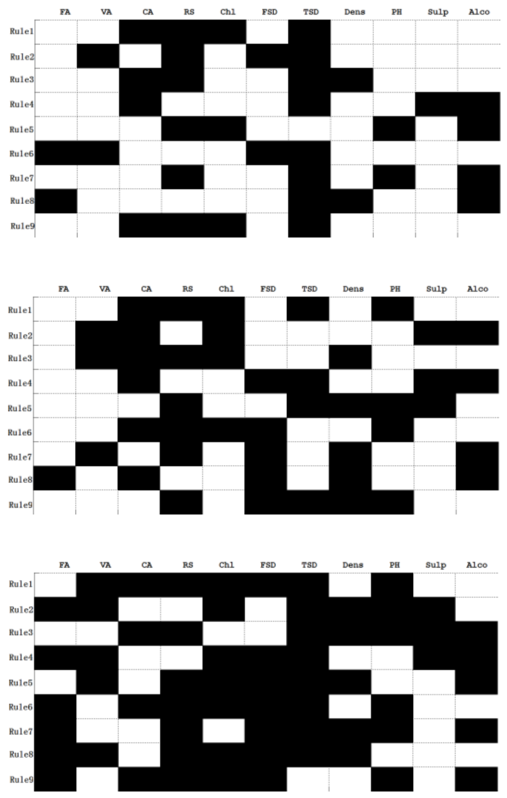


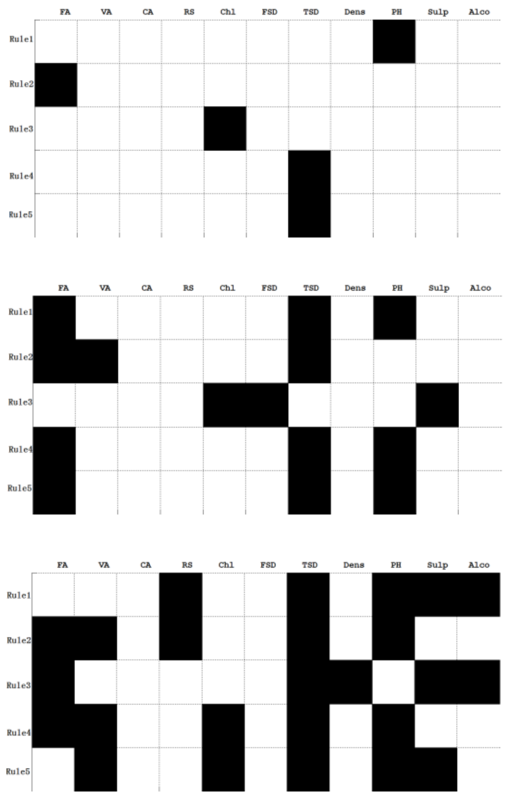
c) Boston Housing with L= 1, 4, 10 d) Computer Activity with L= 2, 11, 13





e) Concrete Strength with L= 1, 2, 5 f) Forest Fires with L= 2, 7, 9





(g) Red Wine Quality with L= 4, 5, 7 (h) White Wine Quality with L= 1, 3, 5

Figure 8.8 Isolated variables (shaded) obtained for selected values of L

8.5 Comparison of the two complexity reduction methods

Both of the two rule complexity reduction methods can efficiently simplify the rules in the rule set got from FCM. The two methods keep at least one attribute in each rule, so that the number of rules will not be reduced. After the optimization of GA, the more significant attributes will be chosen. Beside the similarity, the two methods also have obvious difference. Reduction of input spaces method will cut the rules in the rule base unevenly. After reduction, some rules will be much simpler than others, like rule #5 and #6 for Auto MPG dataset when ν=0.3 in figure 8.4 (b). Isolation of variables will cut off all rules in same manor. Every rule will have same number of input variables after the process. Reduction of input spaces will physically remove the inputs to the rules. Those cut-off inputs will not have effect on the functionality of the resulted FRBS. Isolation of variables will out-stand the more significant variables, while still keep the rest less significant variables as a group to provide their contribution to the FRBS output.

8.6 Conclusions

The two approaches enhancing the interpretability of rule-based models are directly applied to the already constructed Takagi-Sugeno fuzzy models realized with the use of fuzzy clustering. Both the formation of the input subspace of conditions as well as the isolation of the input variables are the methods refining multivariable fuzzy sets produced through fuzzy clustering. The proposed approaches are quantifiable in terms of the level of the interpretability abilities offered by them (expressed either in terms of the number of variables eliminated or the variables isolated). This aspect is helpful in determining how much the interpretability could be enhanced without any significant sacrifice of accuracy of the model. Furthermore in this way one could reveal input variables (or their combinations) that are essential in rule-based modeling.

The approach offers a certain new view at the enhancement of fuzzy rule-based models. There could be several avenues worth pursuing in the future including (a) development of a hybrid arrangement of the formation of subspaces of conditions of the rules associated with some further isolation of variables from such subspaces, (b) use of other techniques of Evolutionary Optimization in the entire process, (c) construction of interpretability measures quantifying various facets of the interpretation mechanisms.

References

[7_1] R. Alcala, M. J. Gacto, and F. Herrera, "A fast and scalable multiobjective genetic fuzzy system for linguistic fuzzy modeling in high-dimensional regression problems," ***IEEE Trans. Fuzzy Syst.***, vol. 19, no. 4, pp. 666–681, 2011.
[7_2] J. M. Alonso, L. Magdalena, and S. Guillaume, "Linguistic knowledge base simplification regarding accuracy and interpretability," ***Mathware Soft Comput***., vol. 13, no. 3, pp. 203–216, 2006.
[7_3] S. Ayouni, S. B. Yahia, Anne Laurent, Extracting compact and information lossless sets of fuzzy association rules, ***Fuzzy Sets and Systems***, Volume 183, Issue 1, 16 , pp. 1-25, 2011
[7_4] J. C. Bezdek, ***Pattern Recognition with Fuzzy Objective Function Algorithms***, Plenum Press, N. York, 1981.
[7_5] U. Bodenhofer, P. Bauer, "A formal model of interpretability of linguistic variables," In: ***Interpretability Issues in Fuzzy Modeling***, J. Casillas, O. Cordón, F. Herrera, and L. Magdalena, Eds. Berlin, Germany: Springer-Verlag, pp. 524–545, 2003.
[7_6] M.Y. Chen, D.A. Linkens, Rule-base self-generation and simplification for data-driven fuzzy models, ***Fuzzy Sets and Systems***, Volume 142, Issue 2, pp. 243-265, 2004.
[7_7] M. J. Gacto, R. Alcalá, and F. Herrera, "Integration of an index to preserve the semantic interpretability in the multiobjective evolutionary rule selection and tuning of linguistic fuzzy systems," ***IEEE Trans. Fuzzy Syst***., vol. 18, no. 3, pp. 515–531, 2010.
[7_8] H. Ishibuchi, T. Yamamoto, Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining, ***Fuzzy Sets and Systems***, Volume 141, Issue 1, pp. 59-88, 2004.
[7_9] A. Krone, H. Krause, and T. Slawinski, "A new rule reduction method for finding interpretable and small rule bases in high dimensional search spaces," in ***Proc. 9th IEEE Int. Conf. Fuzzy Syst***., San Antonio, TX, pp. 693–699, 2000.
[7_10] Y. Yam, P. Baranyi, and C. T. Yang, "Reduction of fuzzy rule base via singular value decomposition," ***IEEE Trans. Fuzzy Syst***., vol. 7, pp.120–132, 1999.

Chapter 9 Conclusions and future work

In this study, we focus on efficient approaches for constructing fuzzy rule-based systems and reducing their complexity. This chapter summarizes the research completed in this study and identifies the future tasks as the extensions of the current work.

9.1 Conclusions

The presented experimental results proof that the fuzzy rule-based models built using FNN and fuzzy c-means together with complexity reduction methods are able to provide concise rule sets and sound prediction accuracy. This has been observed during testing phases for the cases of synthetic data as well as the real world control problems.

Fuzzy neural networks and fuzzy clustering are powerful tools for data modeling. Our research employed these two methods in constructing the accurate and easy understandable fuzzy rule based systems.

First part of our research is dedicated to the construction of FRBS using FNN and evolutionary intelligence techniques. The main contributions of this work are:

- Construction process of FRBSs based on FNNs, where FNNs are built using the genetic algorithm (GA).

- Reduction of dimensionality of FRBS rules and the improvement of their quantity via a pruning process based on threshold values identified for AND neurons and OR neurons. The optimal threshold values for both neurons are determined so there exists a balance between simplicity of rules and accuracy of FRBS. Optimal model satisfying both competing objectives are built using a multi-objective optimization algorithm.

- The constructed FRBSs together with the pruning approaches are verified on datasets from well-known data repositories and for the real world application – deployment of wireless sensor networks.

The second part of the work focuses on the construction of FRBS based on fuzzy clustering. The reduction of complexity of the system is done by providing a system with an up-front determined number of input variables, or by finding isolated variables inside the rules. This part presents the following contributions:

- Construction method of FRBS based on fuzzy clustering realized using the linear approximation functions defined on clusters.

- Reduction of a number of variables in each rule. Two methods are proposed here. One of them is to cut off the input variables based on a provided number of input variables to keep. Another one is to determine isolated input variables in each rule.

- The FRBSs built based on fuzzy clustering and optimized with the two reduction approaches are evaluated with the datasets from well known data repositories.

The two approaches enhancing the interpretability of rule-based models are directly applied to the already constructed Takagi-Sugeno fuzzy models realized with the use of fuzzy clustering. Both the formation of the input subspace of conditions as well as the isolation of the input variables are the methods refining multivariable fuzzy sets produced through fuzzy clustering. The proposed approaches are quantifiable in terms of the level of the interpretability abilities offered by them (expressed either in terms of the number of variables eliminated or the variables isolated). This aspect is helpful in determining how much the interpretability could be enhanced without any significant sacrifice of accuracy of the model. Furthermore in this way one could reveal input variables (or their combinations) that are essential in rule-based modeling.

The originality in our research is manifested by:

- Novel way to configure FNNs for modeling purposes;
- Novel way to organise inputs to each AND neuron in a hidden layer;
- Usage of Pareto front in a pruning process to reduce complexity of the FRBSs with FNN;
- Two methods of complexity reduction for FRBSs built using FCM:
    - via reduction of input space
    - via isolation of input variables
- Construction of FRBSs based on FNN for WSN node deployment problem and the application of a simple pruning and Pareto front pruning processes to reduce the complexity of constructed models.

9.2 Suggestions for future work

Observations and the conclusions stated in Chapters 6, 7 and 8 confirmed the effectiveness and the consistency of constructing FRBS using evolutionary approach in modelling of a given problem. Moreover, the different techniques in reducing the FRBS complexity bring future research tasks in this area.

The presented research results prove that the approaches we have proposed are capable of providing useful tools for data modeling/mining and real world applications. However, we have noticed a few limitations on the approaches. Some suggestions for FRBS constructed based on FCM have been stated in summary of Chapter 8. Here are some additional issues to consider.

Overhead processing when constructing the models

For both FNN and fuzzy clustering, current construction processes take into account all attributes. When given data sets have a large number of input variables, the processing time becomes a big burden. To reduce the performance requirement, we can introduce preliminary feature selection techniques to discount some of the less important variables.

Previous knowledge on some parameter setting

For FNN, before constructing a model, we need to consider how many AND nodes in the hidden layer in order to provide best results for a given problem. Currently, based on extensive experiments and expertise knowledge, we chose nine AND neurons for all problems. This selection needs some verification so that a suitable number might be found. For fuzzy clustering, we need to determine an optimal number of clusters before proceeding to the complexity reduction stage with models. In both cases, there is a need for automatic ways of selecting proper/optimal numbers of nodes and clusters.

The optimal kept ratio when cutting off variables in rules

Currently, we select the number of variables in the input space to be kept ourselves. However, it is possible and desirable to apply GA to determine the optimal number of variables to keep. When a number of clusters and/or input variables become larger, the application of GA to select the optimal number of kept variables seems unavoidable. In the future work, we will apply multi-objective optimization to find the best tradeoff between the model interpretability and its accuracy.

The optimal number of isolated variables

Similarly to the number of kept input variables, the number of isolated variables requires an automatic method of determination. When the number of input variables increases, the application of GA to find the optimal number of isolated variable imposes some performance constrains. In the next step of the work, it is possible to apply multi-objective optimization approach to determine numbers of isolation variables.

Application of other evolutionary optimization algorithms

In our research, we used GA and MOGA as main optimization method. The other evolutionary optimization methods could also be efficient in construction and complexity reduction of FRBS based on FNN or FCM. The possible evolutionary optimization methods could be PSO or DE, and the modified algorithms for multi-objective optimization requirement.

Other improvement for FRBS with FCM

The approach offers a certain new view at the enhancement of fuzzy rule-based models. There could be several avenues worth pursuing in the future including (a) development of a hybrid arrangement of the formation of subspaces of conditions of the rules associated with some further isolation of variables from such subspaces, (b) construction of interpretability measures quantifying various facets of the interpretation mechanisms.

All in all, the proposed methods are able to construct FRBS modeling given data sets with simple rule sets when the rule complexity reduction algorithms are used. The above-mentioned areas for future improvements should lead to even better results.