12-2010

Evolutionary Strategies for Data Mining

Rose Lowe
Clemson University, rlowe@clemson.edu

Evolutionary Strategies for Data Mining

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Computer Science

---

by
Rose M. Lowe
December 2010

Accepted by:
Dr. Dennis E. Stevenson, Committee Chair
Dr. Edward Page
Dr. A. Wayne Madison
Dr. Christopher Cox

Abstract

Learning classifier systems (LCS) have been successful in generating rules for solving classification problems in data mining. The rules are of the form IF condition THEN action. The condition encodes the features of the input space and the action encodes the class label. What is lacking in those systems is the ability to express each feature using a function that is appropriate for that feature. The genetic algorithm is capable of doing this but cannot because only one type of membership function is provided. Thus, the genetic algorithm learns only the shape and placement of the membership function, and in some cases, the number of partitions generated by this function. The research conducted in this study employs a learning classifier system to generate the rules for solving classification problems, but also incorporates multiple types of membership functions, allowing the genetic algorithm to choose an appropriate one for each feature of the input space and determine the number of partitions generated by each function. In addition, three membership functions were introduced. This paper describes the framework and implementation of this modified learning classifier system (M-LCS). Using the M-LCS model, classifiers were simulated for two benchmark classification problems and two additional real-world problems. The results of these four simulations indicate that the M-LCS model provides an alternative approach to designing a learning classifier system. The following contributions are made to the field of computing: 1) a framework for developing a

learning classifier system that employs multiple types of membership functions, 2) a model, M-LCS, that was developed from the framework, and 3) the addition of three membership functions that have not been used in the design of learning classifier systems.

Dedication

This work is dedicated to my late grandson, Tyler Damonte McKenzie.

Acknowledgments

First and foremost, I thank God for inspiring and anointing me to complete this dissertation. I thank my advisor, Dr. Dennis Stevenson for his patience, encouragement, guidance, and support during the course of this research. I thank Dr. Ed Page for serving as my initial advisor and for his encouragement, advice, and countless short stories that were so full of wisdom. I express my appreciation to the other members of my committee, Dr. Wayne Madison and Dr. Chris Cox, for reading this dissertation and for their guidance. I thank Dr. Hugh Spitler for providing the real-world alcohol data. I thank Dr. Larry Hodges and Dr. Mark Smotherman for their support. I thank Mrs. Carleathea (Lea) Benson for her continuous words of encouragement and for all that she did to help me remain focused.

My family has always been a source of strength and encouragement. I am grateful to my husband, Larry, for his love and support. My children, Yolonda, Jason and his spouse Micah, Lauren and her spouse Paul, and Daniel, helped me to keep things in perspective. I thank my granddaughter, Jasmine, for always keeping a song in my heart.

My friends, Sharon Howell, Linda Greene, Mr. Theo Grate and Mrs. Ruby Grate were always available to listen and encourage. I am grateful to them for their prayers. Finally, I thank the US Army Research Office and the Southern Regional Education Board for financial support during my graduate studies.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Data mining is a relatively new discipline that emerged in the 1980s out of a need to develop tools and methods for processing large data sets and databases. Although the field is relatively new, the building blocks of current data mining tools and techniques have been in existence much longer. Tools and techniques used in artificial intelligence, machine learning, information retrieval, database systems, regression analysis, artificial neural networks, evolutionary algorithms, and others are employed in data mining systems. Research in data mining seeks to adapt these systems to work with massive amounts of data as well as develop new ones. The field has experienced rapid growth during the past three decades and currently data mining is applied to a variety of problem domains, including bioinformatics [24, 10, 11], medical applications [6, 77, 48], and business applications [36, 27, 68]. Data mining tasks include retrieval, analysis, summarization, prediction, and classification. Of these, Peng reported that classification is the most used data mining task [63].

There are several approaches to solving classification problems. Learning classifier systems (LCS) have been successful in solving traditional classification problems and researchers have investigated their use in coping with large amounts of data for

solving classification problems in data mining. A learning classifier system is a machine learning model that maintains a population of IF-THEN rules, called classifiers, of the form IF condition THEN action. The condition of the rule may be represented in different ways, one of which is a membership function. Systems that use membership functions generally use one type of membership function for encoding all features of the input space, varying the placement and shape, but not the type. In all known classifiers to date, no one membership function has performed significantly better on all problems. It may be the case that a single type of membership function does not perform well for all features of the input space. A different approach is to allow each feature to be encoded using a membership function appropriate for that feature. The research reported in this dissertation explores the use of learning classifier systems for data mining but investigates the efficacy of employing more than one type of membership function for generating the condition component of the classifiers, and allowing the rule discovery component to choose an appropriate one for each feature of the input space. With this approach, instead of generating the same type of membership function for feature x_i as in Figure 1.1, the membership functions generated for x_i by the classifier developed in this research may be of different types as shown in Figure 1.2. Similarly, the condition component of the rules generated by the former will be comprised of the same function, as in Figure 1.3; whereas, those of the latter may be comprised of multiple membership functions as in Figure 1.4. Allowing a function for each feature has the potential of improving the overall performance of this modified LCS. To implement this modification, the LCS must maintain a set of membership functions from which to choose. In addition to the traditional triangular and trapezoidal functions, this research introduced three membership functions to be used in the LCS.

In learning classifier systems, a genetic algorithm is usually employed in the

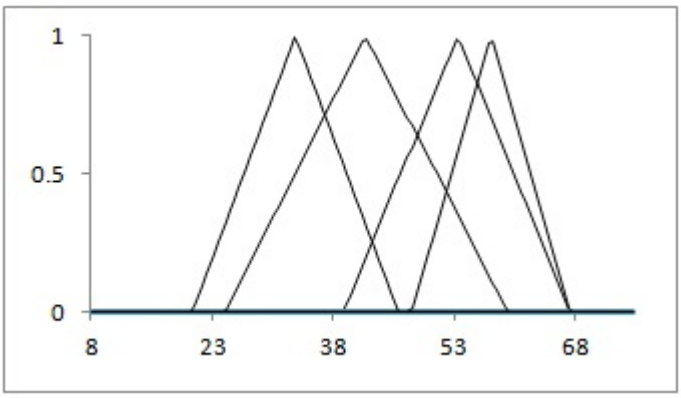

Figure 1.1: Membership functions for x_i using one type

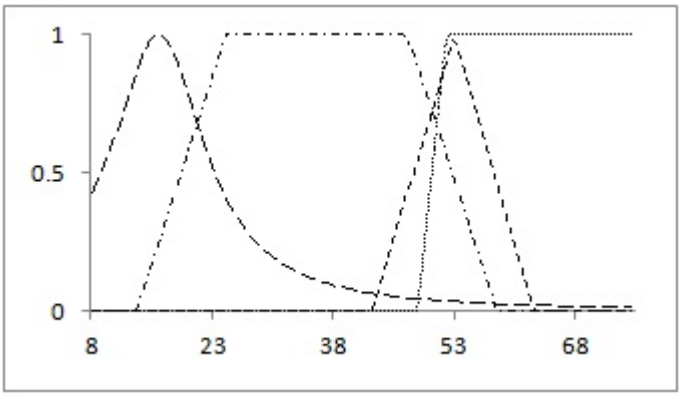

Figure 1.2: Membership functions for x_i using multiple types

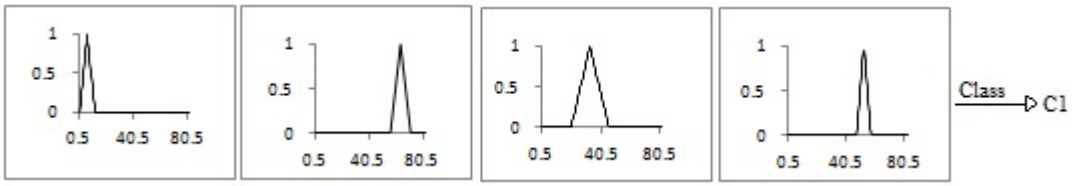

Figure 1.3: Rule generated by using one type of membership function

rule discovery component of the classifiers. A genetic algorithm is a stochastic heuristic search algorithm based on the theory of evolution and the survival of the fittest. It maintains a population of potential solutions to the problem and employs genetic

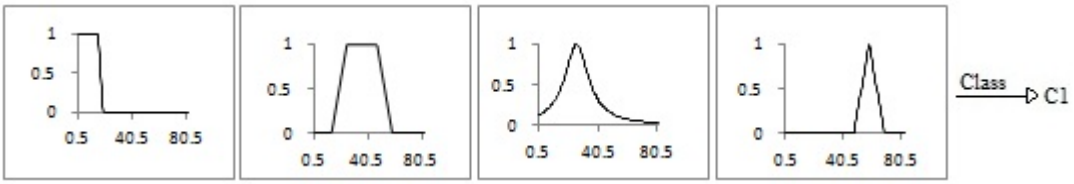

Figure 1.4: Rule generated by using multiple types of membership function

operators to evolve an optimal solution. Although additional operators have emerged over the years, the selection, mutation, and crossover operators are a part of almost every GA in some form. The modified LCS presented in this research uses a genetic algorithm that maintains a population of real-valued, variable length chromosomes, with the following modifications:

1. The mutation and crossover operators were adapted to work with variable length chromosomes consisting of multiple types of membership functions

2. Three additional genetic operators were included

3. The inclusion of three membership functions not previously used with learning classifier systems

1.1 Dissertation Objectives

The objectives of this dissertation were as follows:

1. To develop a framework for a modified learning classifier system that incorporates multiple types of membership functions

2. To include three functions not previously used with learning classifier systems to assist in generating interpretable rules.

3. To use the framework to develop a modified LCS that exhibits a classification accuracy comparable to those reported in the literature.

The remainder of the dissertation is organized in the following manner. Chapter 2 presents background information on data mining, learning classifier systems, genetic algorithms, and search, which are all central to the framework presented in this dissertation. Chapter 3 presents the framework for a learning classifier system that employs multiple types of membership functions, including the representation of the rules, membership functions, genetic operators, learning algorithm, and classification algorithm. The utility of the framework is demonstrated by using it to simulate classifiers and applying the classifiers to two benchmark classification problems and two additional real-world problems. The results of the simulations are reported in Chapter 4. Chapter 5 concludes the dissertation by identifying the contributions of this research and future directions.

Chapter 2

Background

2.1 Introduction

Concepts from data mining, classification, learning classifier systems, genetic algorithms, and search are central to this research. Classification is an essential data mining task. A learning classifier system is just one classification model and the model of choice for this research. Central to most learning classifier systems is a genetic algorithm for generating the classifiers. This chapter provides background information for each of these.

2.2 Data Mining

Advances in data capture and storage technologies during the past three decades have made it possible to capture and store large amounts of data easily, creating a need for tools and techniques for processing the data to provide information that is understandable and useful to the data owner. One field that has emerged as a result is data mining (DM), which is the process of extracting useful information

from large data sets and databases by incorporating tools and techniques from several disciplines, including statistics [5, 65, 74], artificial intelligence (AI) [8, 18], machine learning [39, 58, 67], database technology [85, 37, 12, 51, 78], high-performance computing [14], and data visualization [35, 40]. Data mining techniques can be applied to a wide variety of data types including databases, images, spatial data, temporal data, and text, graphs, streams, Web, biological, and multi-media [63]. Research and applications in this area include methods for data analysis, descriptive modeling, knowledge discovery, retrieval, frequent pattern mining, exception detection, clustering, data visualization, prediction, and classification. In his data mining framework, Peng has identified eight categories of data mining to provide a concise description of the research activities of data mining and knowledge discovery [63]. These categories, described below, are not mutually exclusive.

1. DM tasks include basic functions of data mining and knowledge discovery such as classification, clustering, time series analysis, and exception detection.

2. Learning methods and techniques include disciplines that involve data mining and knowledge discovery or contribute to the area, such as databases, machine learning, artificial intelligence, statistics and optimization. Unsupervised learning, reinforcement learning, artificial neural networks, evolutionary computation, rough sets, fuzzy logic, inductive logic programming, decision trees, support vector machines, data warehousing, online analytical processing (OLAP), data mining query languages, indexing, regression, Bayesian analysis, principle component analysis (PCA), Markov chain Monte Carlo (MCMC), hidden Markov model and discriminant analysis are subcategories of these.

3. Mining complex data involves issues related to mining diverse data types, such as text, graph, temporal, spatial, stream, biological, and multi-media.

4. Foundations of DM involve theoretical and fundamental issues or data mining and knowledge discovery, such as data mining theories, frameworks, taxonomies, measures, privacy, security, and social impact issues.

5. DM software and systems include various aspects of data mining software and systems, such as software maintenance, software development environment, and data mining systems.

6. High-performance and distributed DM focus on designing algorithms and techniques for increasing efficiency in handling massive data sets, for example, parallel algorithms, parallel processing, and distributed data mining algorithms.

7. DM applications describe current data mining application domains and related issues, such as fraud detection, e-commerce, basket analysis, and marketing.

8. DM process and project includes steps in the knowledge discovery process and data mining project related issues, such as process models, preprocessing techniques, model and algorithm selection, post-processing techniques, and visual data mining.

In highlighting the results of his survey of data mining and knowledge discovery research, Peng reported that classification is the most used task of data mining [63].

There are several approaches to classification as outlined in Section 2.3, one being learning classifier systems. Several researchers have investigated the application of learning classifier systems to the classification task of data mining. For example, Bagnall studied the potential of the XCS [80] learning classifier system as a data mining tool by comparing its performance to several other classifier techniques [9], while Wilson investigated the use of this system in classifying oblique data [81]. Bernado applied two learning classifier systems, XCS and GALE, to several data

sets, comparing their performances to six well-known learning algorithms [52]. In all experiments, no classifier performed significantly better than XCS and GALE. In further research, Holmes investigated the use of a learning classifier system for knowledge discovery in epidemiological surveillance [34]. The research reported here also focuses on the classification task of data mining, investigating the efficacy of applying a modified learning classifier system to such problems in data mining.

2.3 The Classification Problem

Classification of objects into different classes is central to most application domains. For example, in biology, large molecules are classified according to their structure and function, such as carbohydrates, lipids or fats, proteins, and nucleic acid. In medical diagnostic systems, classification is central to associating a set of symptoms with a particular disease. In chemistry, classification of molecules is essential in determining the family of molecules that participate in a reaction [43], in business, classification is essential in assessing the credit risk of its customers. For literature mining systems such as MEDLINE, document classification is essential.

The classification problem may be viewed as a problem of finding a function (or a model) that maps a vector of measurements x to a categorical variable Y, for the purpose of using the function or model to predict the class of data objects whose class label is unknown or has not been seen. A block diagram of the classification problem is illustrated in Figure 2.1.

The particular model or function used to generate the classifier depends on the problem to be solved. These include IF-THEN rules, decision trees, artificial neural networks, evolutionary algorithms such as genetic algorithms, learning classifier systems, or mathematical formulas. Since manual classification is seldom feasible for

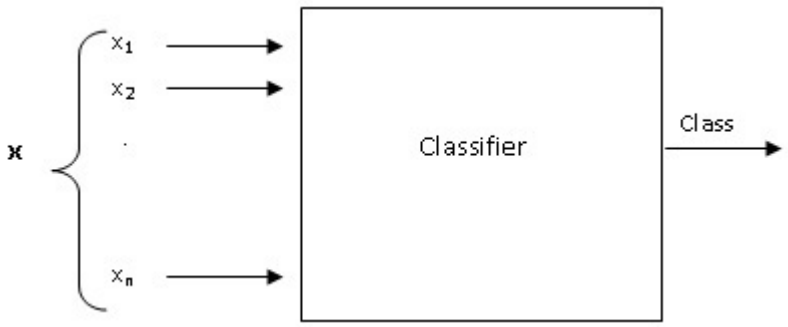

Figure 2.1: Block diagram of the classification problem

complex problems, several automatic methods for deriving classification systems have been developed. These include decision tree induction, statistical classifiers such as Bayesian classification, artificial neural networks, k-nearest neighbor, genetic algorithms, rough sets, fuzzy sets, and cased-based reasoning, regression, and clustering.

If the resulting class is all that is required, without any explanation of how the class label was determined, then either of the above techniques will suffice. For situations requiring an explanation of the reasoning, learning classifier systems have been very successful because they represent the knowledge in rules that are easily interpretable by humans. This research uses learning classifier systems for learning the rules. To test the utility of the derived classifier, the classifier was applied to four problems and the results were compared to those reported in the literature, if possible. The comparison is based on the holdout classification accuracy method, although there are other criteria established for comparing classifier systems.

2.3.1 Comparing and Evaluating Different Classifier Systems

Criteria for comparing and evaluating different classifier systems include classification accuracy, speed, robustness, scalability, and interpretability [32]. Of these, accuracy is the most important and the easiest to assess. Classification accuracy is a

measure of the classifier's ability to classify a new pattern or one that has never been seen. This is an important measure, allowing a developer to evaluate how accurately a given classifier will label new data. Common techniques for assessing classifier accuracy include the holdout method, k-fold cross-validation, and leave-one-out. The holdout method separates the data set into two independent sets, one for training and the other for testing. The training set is used during the learning phase to derive the classifier. Afterwards, the classifier is used to classify patterns in the test set. Classification accuracy is the percentage of test set samples that are correctly classified. In k-fold cross-validation, the initial data set is randomly partitioned into k mutually exclusive subsets of approximately equal size. The training and testing cycle is performed k times, each time reserving one of the subsets for the test set and using the remaining subsets to derive the classifier. Accuracy is computed as the average of the correctly classified patterns. The leave-one-out method is a special case of k-fold cross-validation with k being the number of initial samples.

For binary classification problems, those with classification as either has or does not have (0 or 1), an accuracy measure based only on the number of correct classifications may not depict how well the classifier is correctly classifying the data. For example, if only 4% of the training data are actually 1, an accuracy of 90% may not be acceptable because the classifier could be correctly labeling only the class 0 patterns. For the binary classification problems, the sensitivity and specificity measures can be used to assess the performance. Formulas for these measures are given in Equation 2.1 and Equation 2.2, respectively, where t_{pos} is the number of positive patterns that were classified as positives, pos is the number of positive (1) patterns, t_{neg} is the number of true negative (0) patterns that were correctly classified

as 0, and neg is the number of negative patterns.

$$sensitivity = \frac{t_{pos}}{pos} \tag{2.1}$$

$$sensitivity = \frac{t_{neg}}{pos} \tag{2.2}$$

Accuracy can then be computed as indicated in Equation 2.3

$$accuracy = sensitivity\frac{t_{pos}}{(pos + neg)} + specificity\frac{neg}{(pos + neg)} \tag{2.3}$$

Sensitivity and specificity measures are useful when the main class of interest is in the minority.

For systems that have acceptable accuracy measures, the other measures are helpful in providing additional means of comparing the them. Speed can be viewed as the cost involved in deriving and using the classifier model, with one measure being the amount of machine time it takes to derive the model. For real-world problems it is not uncommon for a system to process data sets that have missing or imprecise data values or irrelevant or meaningless data (noise). Robustness is the measure of the classifier's ability to correctly predict the class label given imprecise, incomplete, or noisy data. Scalability is the ability of the system to derive the model and maintain an acceptable level of performance given increasingly larger data sets. This criterion can be evaluated by assessing the number of I/O operations on such data sets. Interpretability is a measure of how well the model generates understandable rules. This is a subjective criterion; however, it can be evaluated from objective data. For example, in a rule-based system, the measure can be based on the number of rules or the number of features represented in a rule. Interpretability can be measured based

on the number of hidden units for neural networks and the number of nodes for a decision tree.

2.4 Learning Classifier Systems

2.4.1 Introduction

A learning classifier system (LCS), whose fundamental component is a set of condition-action rules called classifiers, is a machine learning model representing the current knowledge of a problem. These systems, first introduced by John Holland [34], which have been in existence for more than thirty years, were originally designed to evolve rules online to model complex adaptive systems [45]. They were widespread during the 1980s, but due to the complexity of the architecture and the online adaptation process, interest in them declined to almost non-existence by the mid 1990s. Interest in the field increased again following the work by Wilson in the design and implementation of XCS [82], which simplified the original design of Holland's system in ways that made the system more easily duplicated.

Currently, learning classifier systems are being successfully applied to a number of machine learning and data mining classification problems. Their success can be attributed to their ability to adapt, generalize, and scale when applied to complex problems.

2.4.2 Learning Rules

The most significant attribute of a learning classifier system is its ability to learn by interacting with the environment. Learning is normally accomplished by utilizing a learning algorithm. Learning algorithms can be categorized as supervised,

unsupervised, or reinforcement. The first classifiers used reinforcement learning; however, supervised learning is also used in current systems. In supervised learning, each input pattern has an associated desired target class. During training, the predicted class is compared to the target class and the fitness value is based on how well the chromosome predicts the target class. Unsupervised learning, on the other hand, does not have knowledge of desired targets. It involves clustering the data into categories to discover features or regularities in the training data. In reinforcement learning, the credit assignment system receives feedback from the environment and distributes the reward among the chromosomes based on their contributions to successful behavior. This differs from supervised learning in that feedback is not in the form of the correct answer. The bucket brigade algorithm is usually employed to manage credit assignment.

2.4.3 Categories of Learning Classifier Systems

Learning classifier systems can be categorized into two types based on the level of learning determined by the form of the rule base: the Michigan model and the Pittsburgh model. In the first, a population consists of a number of condition-action rules, each encoded by a single chromosome in the population and representing a portion of the overall solution. The entire population of chromosomes, thus, constitutes the rule base and represents the overall solution to the target problem. This model was first described by John Holland in 1978 and the first LCS based on it, Cognitive System One (CS-1) devised by Holland and Reitman [23], consisted of a performance component, a reinforcement component known as the credit assignment, and a rule discovery component. The performance component, which consists of the rules, interacts directly with the environment. It specifies how to construct a map-

ping from the rule base to a class label. Since all rules in the population will not be useful nor perform at the same level, the credit assignment component is responsible for assessing the performance of each rule, with the rule discovery component being used to generate new rules to replace those less fit. This rule discovery component is usually implemented using a genetic algorithm. An illustration of a learning classifier system can be seen in Figure 2.2.

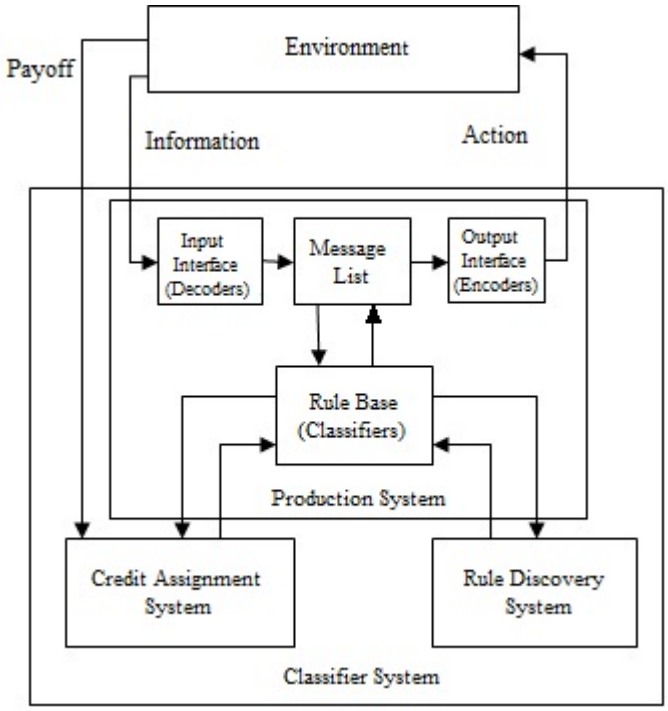

Figure 2.2: Architecture of a Michigan Model LCS

The classifier system interacts with the environment, receiving input and sending actions to be performed. The input interface (detectors) receives messages from the environment and, based on its sensations, sends messages (actions) through the output interface (effectors), thus affecting actions in the environment, such as arm movements of a robot, game moves, or buying shares. Depending on the effectiveness of the actions, the environment may reward the system. The credit assignment

component then determines the reward to assign each classifier based on its level of contribution to the solution. The basic execution of Holland's classifier is given below:

1. Post all input messages on the message list
2. Compare each message with each rule and record all rules whose condition matches the message
3. Merge each matched message with the action part of the satisfied classifier. Replace the old message list with the new message list generated in this way
4. Process the message list through the output interface to produce messages that affect the environment

While the Michigan model was designed to work online, it is also capable of working offline. [75].

In the Pittsburgh style LCS, LS-1, devised by Smith [61], each chromosome encodes a complete rule base (i.e. a set of condition-action rules) instead of one rule as is the case in the Michigan model. The population consists of a number of chromosomes each encoding a rule base representing a complete solution to the target problem. Since a chromosome represents an entire rule base, individual rules do not compete during evolution, meaning a credit assignment component is not needed in this model. The GA searches for the optimal set of rules for solving the problem.
The two models are similar in the following ways:

1. Both models maintain a population or chromosomes.
2. Both consist of a rule discovery component that usually employs a genetic algorithm.

3. A performance measure is used by both to guide the GA in exploring the search space.

4. Reinforcement learning was the original learning mode for both.

The two models differ in the following ways:

1. In the Michigan model, each chromosome encodes a single rule representing a partial solution to the problem and the rule base consists of the entire population of chromosomes; whereas, in the Pittsburgh model, each chromosome encodes a complete rule base representing the complete solution to the problem.

2. In the Michigan model, evaluation consists of both the strength and fitness of a rule. The credit assignment component adjusts the strength of a rule based on the payoff received from the environment and the contribution of the rule to the solution. The Pittsburgh model, on the other hand, uses a purely GA approach to assign fitness to the rule base in terms of the number of patterns it correctly classifies. Consequently, it does not need a credit assignment component for distributing credit to individual rules.

3. The Michigan model uses fixed length chromosomes to represent the classifiers; whereas, the Pittsburgh model uses variable-length chromosomes and applies modified genetic operators for coping with these chromosomes.

4. The Michigan model was designed to work online; whereas, the Pittsburgh model was designed to work offline.

The two rule base configurations are illustrated in Figure 2.3.

Each model has advantages and disadvantages. It is easier to design the rules in the Michigan model since a single chromosome encodes a single rule. In addition, learning is faster and recombination simpler because the recombination operator operates

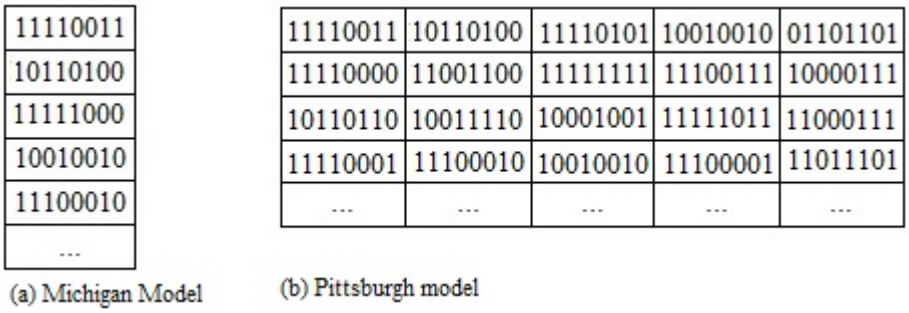

Figure 2.3: Rule base models

on shorter chromosomes and the population requires less memory storage than in the Pittsburgh model. One disadvantage is the conflict between maintaining a population of rules that cooperate to solve the problem and, at the same time, compete to participate in the next generation. Having the entire population represent the complete rule base also causes the rule base to contain a larger number of rules.

Although the design of a chromosome is more difficult in the Pittsburgh approach, its implementation is simpler than that of the Michigan model. Since a single chromosome represents a complete rule base, it is easier to generate a complete set of rules using this model. However, this model requires more memory storage for the population since each chromosome of the population encodes an entire rule base. For the same reason, the Pittsburgh model is more computationally intensive.

In both models, the genetic algorithm is usually employed in the implementation of the rule discovery component of the system. Although the genetic operators are similar in both models, there may be differences in the characteristics of the rules selected for the next generation. Since the selection operator chooses a chromosome based on its fitness, individuals with high values have a higher probability of participating in the next generation. When the entire rule base is represented by a single chromosome, as in the Pittsburgh model, selection is based on the overall fitness of

the entire rule base, not on an individual rule. As a result, individual rules with high fitness values might be replaced and not participate in the next generation if the overall fitness of the rule base is relatively low. This is not the case with the Michigan model. With the Michigan approach, every rule competes for selection, meaning if an individual rule has a high fitness value, then it is likely to participate unchanged in the next generation.

2.5 Genetic Algorithms and Biological Evolution

The origin of GAs dates back to 1975 with the work of John Holland [33], a professor of psychology and computer science at the University of Michigan. Holland devised a genetic code of binary digits that could be used to represent any type of computer program, using 1 to represent true and 0 false. Holland showed that, given a long enough string, it is possible to represent any object by the right combination of binary digits. This approach, called a genetic algorithm (GA), is a computational model that is conceptually based on Darwin's theory of evolution, which is based on the concept that the genetic information on human chromosomes is passed from one generation to the next. As these traits change, or evolve, over time, the individuals expressing these variants exhibit a greater chance of surviving and reproducing, supporting the principle of survival of the fittest [21].

GAs refer to a class of stochastic search techniques that emulate the mechanics of natural evolution [13]. This type of algorithm is comprised of a population of potential solutions and a corresponding set of genetic operators that guide the search through it to identify a string representing a suitable solution to the problem. The components of a GA include

1. A suitable representation

2. A method for creating an initial population of solutions

3. A function that assigns a fitness measure to each solution in the population

4. Genetic operators that are applied to the population of solutions in a probabilistic manner

The overall procedure of a basic genetic algorithm is as follows: An initial population of potential problem solutions is randomly generated and subsequently evaluated for fitness. Then, the selection operator is applied to create an intermediate population. Next, the genetic operators are applied to this population to create a population of new and possibly more fit solutions consisting of chromosomes that have survived the fitness test and offspring produced from applying the genetic operators. The process of moving from the current population to the next population constitutes one generation in the execution of the GA. As solutions alter and combine, the worst solutions are discarded, and the ones that have higher fitness survive to participate in the next generation to produce solutions that have even higher fitness values. As a result, the new generation usually contains strings superior to those from the previous generation. A flowchart of this procedure is illustrated in Figure 2.4.
GAs have been successfully applied to a variety of problem domains, including problems in optimization, product design, and monitoring industrial systems. Commercial applications of GAs continue to emerge.

To use a GA, the developer makes a number of crucial architectural decisions to determine the encoding scheme, the fitness function, and the genetic operators, including the probabilities associated with each. The GA devised by Holland, referred to as the canonical GA, uses fixed length chromosomes, binary encoding, and a fixed set of genetic operators. Variations of the canonical GA use variable length chromosomes, different encodings, and additional genetic operators.

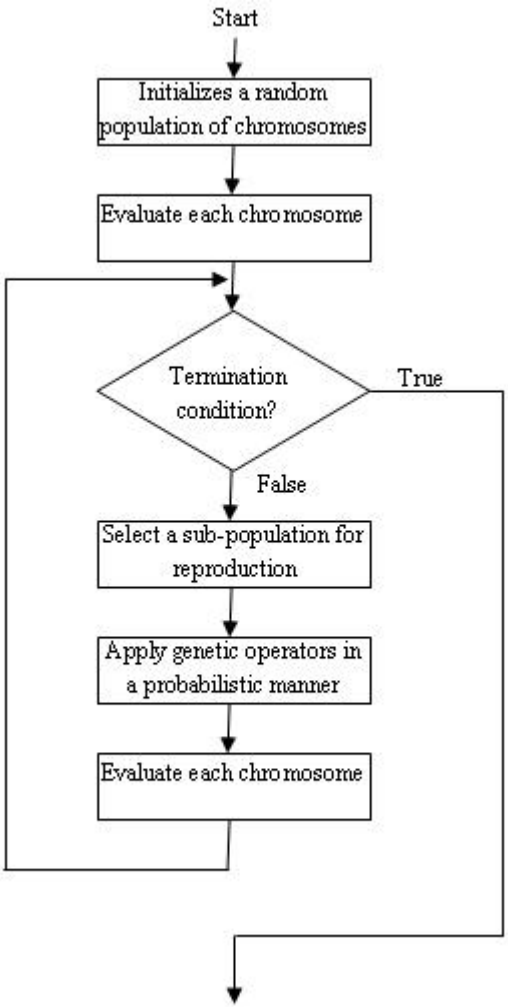

Figure 2.4: Genetic Algorithm

2.5.1 Encoding Techniques

In natural genetics, every organism has a set of rules describing how that organism is generated from the tiny building blocks of life. These rules are encoded in the genes of the organism. The genes are in turn connected into long strings called chromosomes, with each gene representing a specific trait of the organism such as height or eye color. Similarly, in a GA, the potential solution to a problem is encoded in a chromosome-like structure, with each gene representing a feature of the input. The encoding of the solutions depends on the problem to be solved. For example, for an optimization problem, the chromosome can be used to encode the values for the parameters being optimized, while in a classification problem, it can be used to represent either a single rule or an entire rule base. Choosing a suitable representation of a solution to the problem is critical to the success and performance of the GA [2]. Methods typically used for this encoding are bit string, value, and a permutation. In bit string encoding, each chromosome is represented by a binary string of a fixed length as illustrated in Figure 2.5.

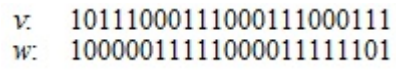

Figure 2.5: Example of a chromosome with binary encoding

This encoding scheme works well if the solutions do not involve integer, real-valued data, or character data. If a binary string is used to represent integer data or real-valued data, each value must first be encoded with the chromosome consisting of a binary string for each value and decoding is necessary to evaluate the string. Michalewicz [54] gives one example of this scheme. Given a variable x whose domain has length 3 and precision requirement of six places after the decimal point, the range $[-1..2]$ should be divided into at least 3 * 1000000 equal size ranges. A chromosome

length of 22 bits is required for the encoding, since

The following procedure was used to decode the binary string to obtain its equivalent real number:

1. Convert the binary string from base 2 to base 10

2. Use the following Equation 2.4 to find the corresponding real number x:

$$x = -1.0 + x' \frac{3}{2^{22}} - 1 \tag{2.4}$$

j where -1.0 is the left boundary of the domain and 3 is the length of the domain. For example, the chromosome

1000101110110101000111

represents the number 0.637197, since

$x' = (1000101110110101000111)_2 = 2288967$

and

$x = -1.0 + 2288967 \frac{3}{4194303} = 0.637197$

With this representation, the length of the chromosome is directly proportional to the number of features that must be encoded, meaning a bit string representation can become long for a problem involving several features. To address this issue, researchers have used value-encoded chromosomes to represent the data along with specialized genetic operators to manipulate the chromosomes [23, 54]. In value encoding, a chromosome is represented as a fixed-length string of values. The values used depend on the type of data, for examples, integer, real, or character. In a permutation encoding, each chromosome is a string of numbers representing the numbers in a sequence. This type of encoding is suitable for problems that involve ordering things,

such as ordering the cities to be visited as in the traveling salesman problem or ordering tasks to be performed. Examples of value and permutation encodings are illustrated in Table 2.1. In the permutation example, if the values represent tasks to be performed, then Task 3 should be done first, followed by Task 5, then Task 1, etc.

Table 2.1: Examples of alternate encoding schemes

integer:	3	1	3	6	5	2	7
real:	7.4	0.3	2.1	2.5	0.1	4.1	8.3
character:	A	B	C	D	A	B	C
permutation:	3	5	1	4	6	7	2

No matter which encoding is chosen, there must be an evaluation function to evaluate the chromosome and assign it a fitness value.

2.5.2 Fitness Function

One of the most difficult and critical tasks in developing a GA-based solution is the design of the fitness function. This function determines both the success of the GA and the speed at which it converges. A GA operates on a population of potential solutions to a problem, not the problem itself, relying on the fitness of the chromosome to determine how close it is to the solution. Chromosomes with superior characteristics should be assigned a higher fitness value than those that are less fit; otherwise, the GA will take longer to evolve a solution that performs well. The choice depends on the problem to be solved. For a function approximation problem, the fitness function may be the evaluation of the function at the given values. For a classification problem, it may be necessary to develop a simulation

and evaluate a chromosome based on the its outcome. Similarly, for a model design, fitness may involve developing a simulation of the model. For learning classifier systems applying reinforcement learning, fitness is usually based on the strength of the rule; however, accuracy-based fitness is used with more recent systems employing supervised learning.

2.5.3 Genetic Operators

Genetic operators, in concert with the fitness measure, improve the population of solutions from one generation to the next. They are applied to the chromosomes in a probabilistic manner. The three genetic operators of the canonical GA, selection, crossover, and mutation, are a part of almost every GA in some form. The inversion operator is also used with position independent encoding.

2.5.3.1 Selection

Selection in a GA operates in a manner similar to Darwin's natural selection. Darwin recognized that an advantageous trait first appearing in only one or a few individuals of a population gives them and their descendants superiority over others in the population. This process of selecting an advantageous trait and transmitting it to a larger proportion of individuals in the population in each successive generation is referred to as natural selection. Ultimately, many generations of natural selection produce a population of members all expressing this trait. At the same time, variant forms of other traits arise in each generation, and natural selection will, for each, determine which survives over time. In a GA, the fitness function assigns a fitness value to each chromosome. The selection operator works in conjunction with the fitness value to choose chromosomes for the next generation by making copies of the

more successful ones and deleting the less successful, ensuring better chromosomes in the next generation. However, this selection operator must be chosen with care. Although the selection operator should be biased toward the better individuals in the population, it should also choose those that do not have high fitness values but rather those having genetic material that will help the population to improve; otherwise, the population will rapidly converge to one individual, a result that is not desirable.

Selection schemes include truncation, fitness-proportionate, linear ranking, tournament, sigma scaling, and steady state selection. Truncation selection is the simplest form. With this method, individuals are sorted according to their fitness, from highest to lowest and the top n% are selected as parents for the next generation. These individuals are duplicated to maintain the population size. For example, given a population size of 200, truncation might select 25% of these as parents and then create four copies of each to maintain a population of 200 individuals. Although this scheme is easy to implement, it can result in premature convergence since less fit individuals are not given an opportunity to evolve into ones more fit. To overcome this, fitness-based approaches give every individual a chance to participate in the next generation with fitter individuals more likely to be chosen than weaker ones. Fitness-proportionate selection chooses individuals to participate in the next generation based on the ratio of their individual fitness values to the average fitness of the population. The roulette wheel algorithm [23], described in Table 2.2, is most commonly used to implement fitness-proportionate selection; however, stochastic universal sampling is another. For example, given the chromosomes and corresponding fitness values in Table 2.3, the algorithm will choose chromosome 5 if the random fitness value is 3.0. Linear ranking selection sorts individuals according to their fitness values, choosing them based on their rank. The rank of the least fit is zero while the rank of the most fit is $N-1$, where N is the size of the population. Unlike fitness-proportionate selection,

Table 2.2: Roulette wheel algorithm

1. Compute the sum S of the fitness values of all members of the population,
2. Generate a random number r from the interval $(0, S)$
3. initialize partial sum s to zero
4. While r is greater than s
   4.1. Add the fitness of this chromosome to s
   4.2. If r is less than s, return this chromosome.

Table 2.3: Examples of roulette wheel algorithm

Chromosome:	1	2	3	4	5	6	7
Fitness:	0.75	0.32	0.8	0.55	0.7	0.92	0.36
Partial Sum:	0.75	1.07	1.87	2.42	3.12	4.04	4.4

no individual generates an excessive number of offspring in the next generation, thus overcoming the problem of stagnation or premature convergence.

Tournament selection chooses a number of individuals randomly from a population and selects the best from this group for further genetic processing, repeating this process as often as desired. For selection, an individual must win a competition, or a tournament, with a randomly selected set of individuals in the population. In a k-ary tournament, the best of k strings is selected for the next generation. One advantage of this scheme is the potential for massive parallelism. An example of tournament selection is illustrated in Figure 2.6. Steady state selection chooses an individual according to linear ranking, then chooses the currently worst individual for replacement.

Although selection can increase the number of superior individuals in the population, in *The Origin of Species* Darwin cautioned that it is not the exclusive means of genetic modification [21]. Selection alone cannot create improved offspring. To

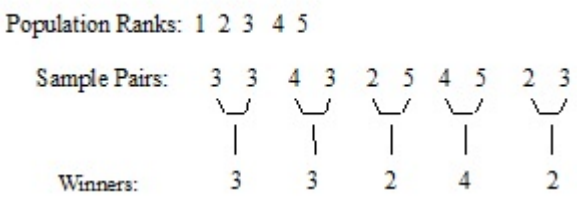

Figure 2.6: Tournament example. More fit individuals have lower rank values.

achieve that, variation operators are required, the two common ones being crossover and mutation.

2.5.3.2 Crossover

In natural genetics when two parents mate, the resulting offspring receive half its genes from one parent and half from the other. Before mating, the exchange of chromosomal material occurs between each identical pair of parental chromosomes by breakage and reunion during the production of mating cells. This process, referred to as crossing over, is the basis of genetic recombination. In a similar manner, the crossover operator in a genetic algorithm forms two new chromosomes by combining select parts of two parent chromosomes and then exchanging these parts beyond this point. While crossover conserves the genetic information present in the chromosomes crossed, it creates chromosomes superior to the two originals. The different types of crossover operators include one-point, two-point, uniform, and average crossover.

In one-point crossover, the chromosomes are randomly selected in pairs. Given two chromosomes in generation t:

$v^t = (v_1, v_2, \ldots, v_N)$ and

$w^t = (w_1, w_2, \ldots, w_N)$

where N is the length of the chromosome and a point $k \in \{1, 2, \ldots, N\}$ generated

randomly with v^t and w^t being crossed at the kth position, the resulting offspring, produced for generation $t+1$ by exchanging the genetic material of the two parents after the k-th position, are illustrated in 2.7.

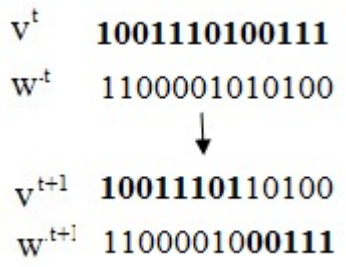

Figure 2.7: One-point crossover

The offspring are composed of the head of one parent's chromosome combined with the tail of the other.

The other three types function similarly. In two-point crossover, two points $j, k \in \{1, 2, \ldots, N\}, j < k$, are randomly generated and two offspring are produced by exchanging the genetic material of the parents between j and k and leaving unchanged the parts preceding j and those following k as illustrated in 2.8.

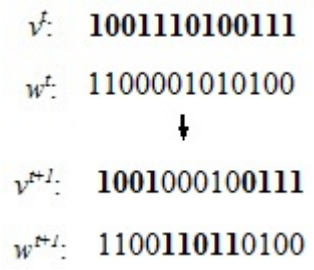

Figure 2.8: Two-point crossover

Uniform crossover, on the other hand, randomly chooses each bit position of the two parent chromosomes and exchanges the two bits if applicable. The last type, an arithmetic crossover, is generally used with real-valued chromosomes [70],

one example being whole arithmetic crossover which produces offspring by computing linear combinations of the two parents. Given two chromosomes at generation t:

$v^t = (v_1, v_2, \ldots, v_N)$ and

$w^t = (w_1, w_2, \ldots, w_N)$

the resulting offspring at generation t+1 are

$v^{t+1} = r * v^t + (1 - r) * w^t$ and

$w^{t+1} = r * w^t + (1 - r) * v^t$

This type of arithmetic crossover is illustrated in 2.9.

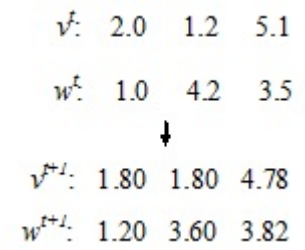

Figure 2.9: Arithmetic Crossover

Although crossover is capable of producing chromosomes that are superior to others, it cannot introduce new genetic material into the population. For this, the mutation operator is necessary.

2.5.3.3 Mutation

Mutations serve as the foundation for the evolution of species and the survival of organisms. The balance between the continuous introduction of new advantageous mutations and the loss of deleterious ones provides a selective advantage to individuals carrying them. Just as mutations occur by chance in natural evolution to alter the genetic information, a mutation operator is used in a genetic algorithm to alter the genetic information of a gene at random. This mutation operator iterates over

the chromosome and, if applicable, changes a gene of the chromosome, thereby introducing new genetic material into the population. In the case of binary-encoded chromosomes, mutation simply changes the bit, while for value-encoded chromosomes, mutation replaces the value of the gene with a randomly generated value within the desired range. A binary-encoded chromosome with mutation applied to its third, seventh, and twelfth genes is illustrated in Figure 2.10, with the gene to be mutated and the mutated genes in bold.

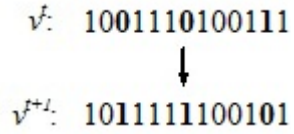

Figure 2.10: Mutation of third, seventh, and twelfth genes

2.5.4 Inversion

The inversion operator is another variation operator, but only suitable for a permutation encoding. It selects two random points of the chromosome and reverses the order of the genes between these points as illustrated below, with the genes to be inverted and the inverted genes in bold.

Original chromosome: 15**7392**84

Inverted chromosome: 15**2937**84

2.6 Search

For a number of problems, there exists a known algorithm for solving it and a computer program can be developed to implement the solution. Such algorithms are referred to as deterministic algorithms. For many real-world problems no known

algorithm exists for solving the problem. For others, an algorithm may exist, but the time or storage required prohibits its implementation. For these problems, finding a solution reduces to a search problem. It is necessary to search the space of possible solutions to find one that is suitable. The type of search used depends on the problem to be solved. Search techniques can be categorized as blind random, exhaustive, or heuristic search.

A blind random search algorithm is an unsystematic search technique that does not use any domain-specific knowledge to direct it even if knowledge is available. Exhaustive techniques, such as breadth-first and depth-first searches, systematically search through the search space until a solution is found. Although they do not use domain-specific knowledge about the problem to guide the search, they are guaranteed to find a solution if one exists. For small problems, using either blind random search or exhaustive search techniques may be quite feasible; however, these techniques are rarely feasible for non-trivial problems. To address these issues, heuristic search techniques were developed.

Heuristic search seeks to reduce the search space by using information about the problem domain. In their early development heuristic search methods were applied to a single domain and domain-specific knowledge was closely intertwined with techniques for using this method. Heuristics were not easily accessible for study nor adaptation to new problems. Because of the tight coupling of the search technique with the domain-specific knowledge, it was likely that similar heuristic techniques had to be repeatedly developed for different applications. Beginning in the mid 1960s, researchers began developing generalized heuristic search algorithms whose properties could be studied independently of the particular programs that might use them, thus making it possible to apply the heuristic algorithm to a wide variety of problems. The genetic algorithm is one such heuristic search technique; others include hill climbing

and simulated annealing. The latter two will not be included in this research.

Chapter 3

A Modified Learning Classifier System

3.1 A Framework for M-LCS

Chapter 2 presented two types of learning classifier systems: the Pittsburgh model and the Michigan model. This research uses the Pittsburgh model because its implementation is simpler and it is easier to generate a complete set of rules using it since a single chromosome encodes a complete rule base.

A learning classifier system based on the Pittsburgh model is an algebra characterized by a search space $\mathcal{S}$, a heuristic function $\mathcal{F}$, and a membership function $\mathcal{M}$. In a classical GA approach, the heuristic function consists of selection, crossover, and mutation operators. The M-LCS extends this algebra by using a set of five types of membership functions $\mathcal{M}'$ instead of one. The heuristic function $\mathcal{F}'$ consists of three additional genetic operators – a creep operator, an insert operator and a delete operator. In developing a framework for the M-LCS, the following seven aspects of the model need to be addressed:

1. A set of membership functions for deriving the rules
2. Representation of the membership function
3. Representation of a single rule
4. Representation of a rule base
5. A set of genetic operators
6. Constraint satisfaction
7. An evaluation function.

3.1.1 A set of membership functions for deriving the rules

Chapter 2 presented several approaches for representing the rules in a rule base, one being membership functions. This research uses membership functions for the following reasons: 1) they do not require crisp clearly defined boundaries, calculating instead a degree of membership of a feature in the interval [0, 1], and 2) they make it possible to express rules in a form easily interpreted by humans. Selecting an appropriate one can be challenging because of the number available: triangular [20, 28, 29], trapezoidal [42, 29], ellipsoidal [3, 84], Gaussian [30, 15], and sigmoidal functions [76, 25], to name a few. Current classifier systems employing membership functions use a single type to derive the rules; however, since no membership function performs best for all domains [83, 66], the performance of the resulting classifier system is limited by the performance of the one chosen. To address this limitation, the research reported here examines the efficacy of employing multiple types of membership functions to derive the classifiers, with the GA choosing an appropriate one from the available set for each feature, producing rules that may consist of two or more

membership functions. In addition, the M-LCS also employs membership functions that have not been used in previous classification problems in an attempt to generate rules that are easily expressed. Although the system is capable of using any number of membership functions in its set, this research focuses on the use of five: triangular (Equation 3.1), trapezoidal (Equation 3.2), two ramp functions (the γ function and the L-function) [64]. described in Equation 3.4 and Equation 3.5, and a fuzzy membership function [45] described in Equation 3.3.

The triangular and the trapezoidal functions were chosen because of their widespread use in deriving classification rules, having been employed in the design of fuzzy logic controllers [71, 16] as well as with evolutionary algorithms in the design of classification systems [42, 29]. The ramp functions have simple forms and work well for representing situations in which a condition is true up to or past a certain value. For example, in a classification problem involving age and weight, age greater than 50 and weight less than 160 can be easily represented using a ramp function. Since the fuzzy membership has been successfully applied to fuzzy classification problems involving controllers, this research investigates the efficacy of using this membership function in conjunction with a genetic algorithm. While all of these functions have been previously used in the design of fuzzy logic classification systems, there is no known use of the last three with evolutionary algorithms. The graphical representations of the functions are given in Figures 3.1 - 3.5.

$$f(x;\alpha,\beta) = \begin{cases} 0 & x < \alpha \\ \frac{2(x-\alpha)}{\beta-\alpha} & \alpha \leq x < \frac{\alpha+\beta}{2} \\ \frac{2(x-\beta)}{\alpha-\beta} & \frac{\alpha+\beta}{2} \leq x \leq \beta \\ 0 & x > \beta \end{cases} \tag{3.1}$$

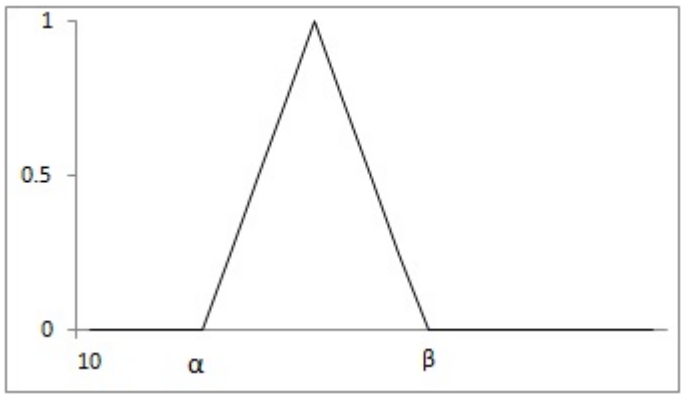

Figure 3.1: Illustration of triangular function for Equation 3.1

$$f(x;\alpha,\beta) = \begin{cases} \frac{4(x-\alpha)}{\beta-\alpha} & \alpha \leq x < \frac{3\alpha+\beta}{4} \\ 1 & \frac{(3\alpha+\beta)}{4} \leq x < \frac{3\beta+\alpha}{4} \\ \frac{4(x-\beta)}{\alpha-\beta} & \frac{3\alpha+\beta}{4} \leq x \leq \beta \\ 0 & otherwise \end{cases} \tag{3.2}$$

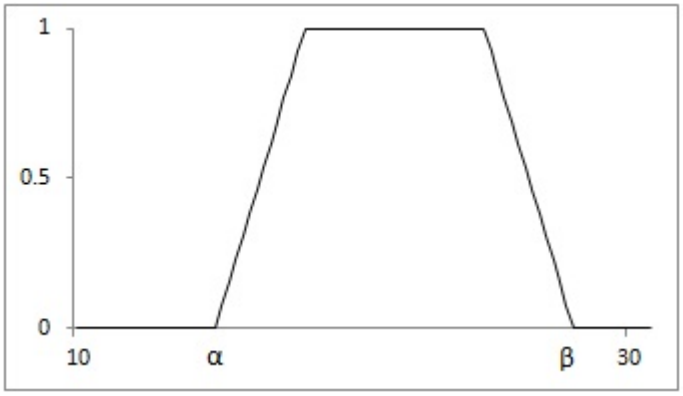

Figure 3.2: Illustration of trapezoidal function for Equation 3.2

$$f(x;\alpha,\beta) = \frac{1}{1+(\frac{\alpha-x}{\beta})^2} \tag{3.3}$$

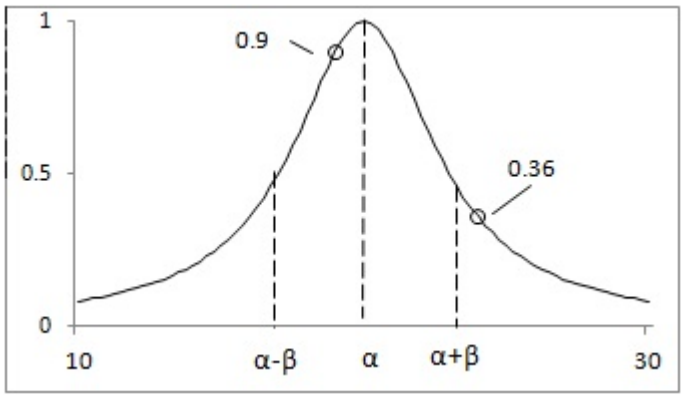

Figure 3.3: Illustration of Equation 3.3 for $\alpha = 20$ and $\beta = 3.0$

$$f(x;\alpha,\beta) = \begin{cases} 0 & x \leq \alpha \\ \frac{x-\alpha}{\beta-\alpha} & \alpha < x < \beta \\ 1 & x \geq \beta \end{cases} \tag{3.4}$$

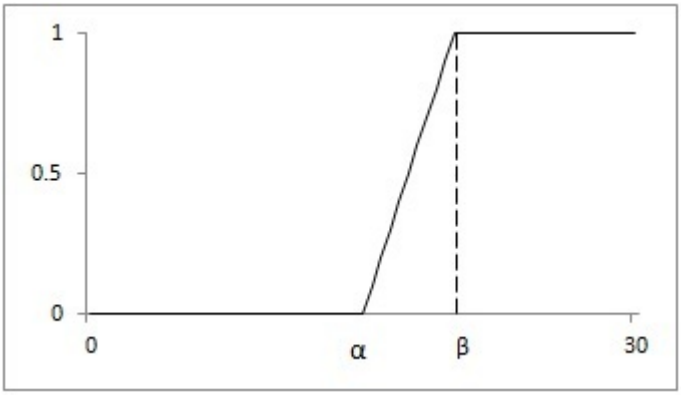

Figure 3.4: Illustration of γ-function for Equation 3.4 for $\alpha = 20$ and $\beta = 23$

$$f(x;\alpha,\beta) = \begin{cases} 1 & x \leq \alpha \\ \frac{x-\beta}{\alpha-\beta} & \alpha < x < \beta \\ 0 & x \geq \beta \end{cases} \tag{3.5}$$

In this research, the GA determines the placement of each membership func-

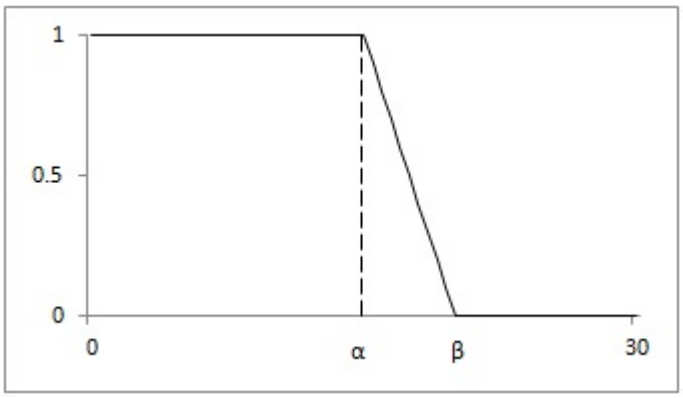

Figure 3.5: Illustration of L-function for Equation 3.5 for $\alpha = 20$ and $\beta = 23$

tion by learning the values of the associated α's and the β's. Incorporating multiple functions adds to the complexity of the representation. To address this issue, this research assigns a numeric code to each membership function in the set and then simplifies the application of the genetic operators by using a consistent representation for each. This representation of a membership function and its parameters is given by a 3-tuple $< fn\_code, \alpha, \beta >$, where fn_code is the predefined code for the function and α and β are its parameters. The triangular and trapezoidal functions are restricted to isosceles triangles and trapezoids to conform to this 3-tuple notation. In the representation scheme used in this research, a 3-tuple constitutes a single gene of the chromosome.

3.1.2 Representation of a Single Rule

A classification rule consists of two components, the condition and the associated action, taking the form IF condition THEN action. The condition component of the rule can be represented using strings over the alphabet {0, 1, ♯} [80, 56], disjunctions of intervals [47, 60], symbolic expressions [44, 46, 55], first-order logic expressions [49, 53], or membership functions [20, 29]. The action component can be represented

by continuous functions [82, 17], membership functions [41], or a set of labels [7, 79]. This research uses membership functions to represent the condition and a set of labels to represent the action. A single rule encodes a membership function for each feature of the input space, using the 3-tuple notation described above, followed by a discrete value for the action. In the M-LCS, the action is a value representing the class associated with the patterns that satisfy the condition. Given N input patterns, m input features, and K predefined classes, the ith rule is represented by a fixed length array of real values as illustrated in Figure 3.6:

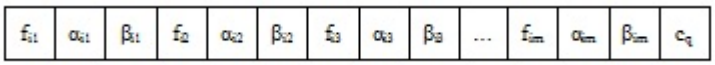

Figure 3.6: Rule i

If μ_{ij} represents the membership function for feature j of rule i, then the rule can be interpreted as

IF μ_{i1} *and* μ_{i2} *and* ... *and* μ_{im} THEN c_q

For example, in a garment sizing system, the input may consist of height, weight, and chest size, and the output the size of the garment. For a 3-tuple representation of height as 0.0 60.0 5, 1.0 110.0 140.0 for weight, and 1.0 32.0 36.0, for age and a garment size of 34, the rule is represented in Figure 3.7.

0.0	60.0	5.0	1.0	110.0	140.0	1.0	32.0	36.0	34

Figure 3.7: Rule i

Each 3-tuple is a gene and each rule represents a small part of the overall solution to the problem, with the complete solution consisting of one or more rules called a rule base.

3.1.3 Representation of the Rule Base

The number of rules in a rule base is problem dependent, either specified by the developer or learned by the system. Requiring the developer to specify the number of rules is a concern since not all problems require the same number of rules. If the developer overestimates the number needed, the system may not generalize well. Too few rules may cause the system to fail to learn the necessary ones. The M-LCS addresses this problem by adopting the Pittsburgh model to represent a rule base and allowing the length of the chromosome to vary based on the number of rules in it. Figure 3.8 illustrates a rule base consisting of three rules and Figure 3.9 and one consisting of five. The GA then learns the appropriate number of rules for solving the problem.

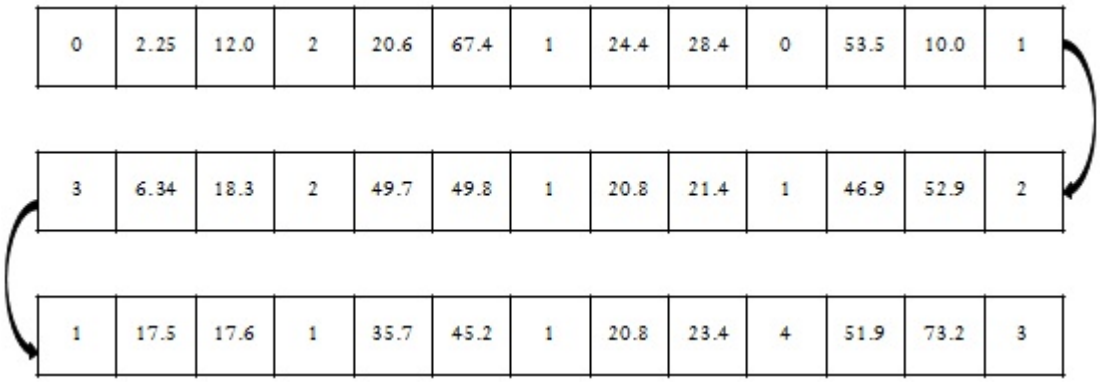

Figure 3.8: Chromosomes consisting of 3 rules

3.1.4 Genetic Operators

Given a rule base consisting of chromosomes of variable lengths, the next issues to be resolved involve 1) the choice of operators that generate meaningful offspring and 2) the manner in which the operators are applied during reproduction. For this study, the classical crossover and mutation operators were modified and additional operators

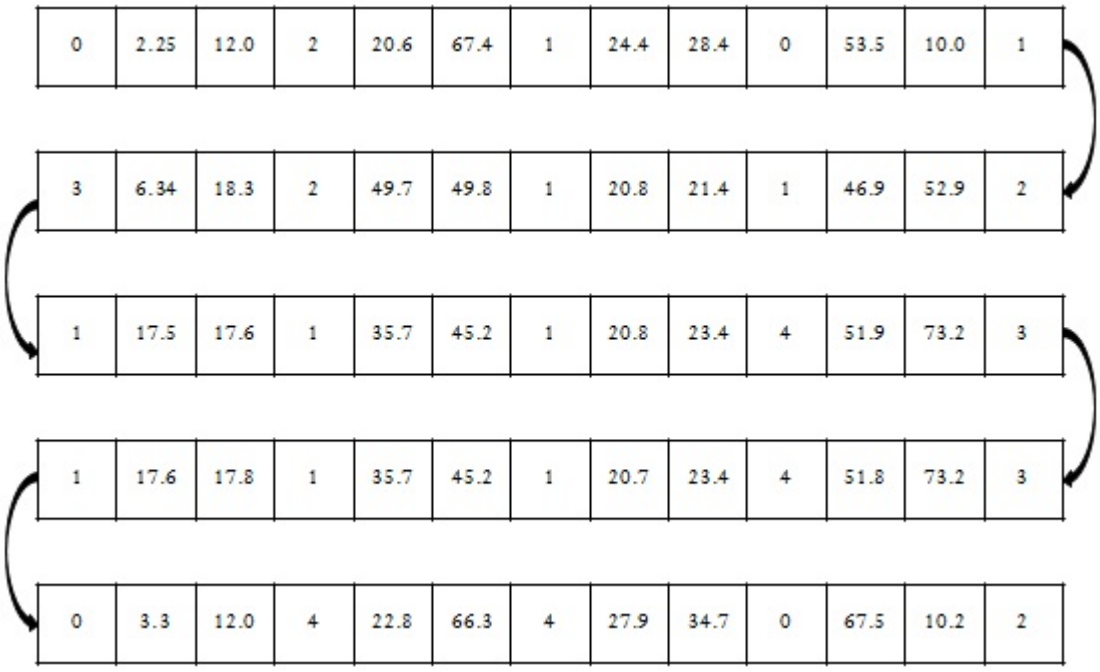

Figure 3.9: Chromosomes consisting of 5 rules

used by Davis [23] were added to accommodate a rule base comprised of multiple types of membership functions, variable-length chromosomes, and real-valued genes. Specifically, the M-LCS uses a selection operator, a no operation (NOP) operator, and five variation operators: one point crossover, mutation, creep [23], insert [23], and delete [23].

3.1.4.1 Selection Operator

The selection operator works in conjunction with the fitness function to select an intermediate population of chromosomes to participate in reproduction. The M-LCS uses fitness proportionate selection, implemented using the roulette wheel selection algorithm [23] described in Chapter 2. The intermediate population is selected, operators applied to it and the resulting chromosomes are placed in the next generation. Using this technique, it is possible that the best chromosome will not sur-

vive to participate in the next generation. To guarantee having the best chromosome in the next generation, the elitist method, which first copies the best chromosome to the next generation, is used. With this technique the best chromosome is always preserved in the population.

3.1.4.2 NOP Operator

It is desirable to have some chromosomes participate in the next generation without any modifications. The NOP operator is designed for this purpose. NOP does not perform an operation on the chromosome, thus allowing it to be included in the next generation unchanged.

3.1.4.3 Crossover Operator

The M-LCS uses one-point crossover modified to work with variable-length rules containing multiple types of membership functions and real-valued genes. For this operator, the chromosomes are randomly selected in pairs (v^t, w^t). Given two randomly selected chromosomes in generation t, v of length M and w of length N:

$v^t = (v_1, v_2, \ldots, v_M)$ and

$w^t = (w_1, w_2, \ldots, w_N)$, with v and w

being crossed at the j^{th} and k^{th} positions, respectively. The resulting offspring are

$v^{t+1} = (v_1, \ldots, v_j, w_{k+1}, \ldots, w_N)$ and

$w^{t+1} = (w_1, \ldots, w_k, vj + 1, \ldots, v_M)$

where $j \in \{1, 2, \ldots, M\}$ is the position of an element in chromosome v , and $k \in \{1, 2, \ldots, N\}$ is the position of an element in chromosome w.

Crossovers may occur either at a gene boundary or at a rule boundary. If the former, they must occur at a feature boundary within points matching on the same feature. This restriction is imposed since the parameters of multiple-membership

functions have different meanings. Thus, restricting the crossover points to a feature boundary helps to maintain legitimate rules.

3.1.4.4 Mutation Operator

The mutation operator used in the M-LCS, which is similar to the one used in other systems, produces a rule that is the same as the parent in all locations except for one or more randomly selected parameters of the membership functions. The selected parameters are replaced by randomly generated values in the allowable range for the feature. Given a randomly selected chromosome,

$$v^t = (v_1, v_2, \ldots, v_i, v_{i+1}, \ldots, v_j, v_{j+1}, \ldots, v_k, v_{k+1}, \ldots, v_M)$$

mutated at the i^{th}, j^{th} and k^{th} positions, respectively, the resulting offspring is

$$v^t = (v_1, v_2, \ldots, v_i^{'}, v_{i+1}, \ldots, v_j^{'}, v_{j+1}, \ldots, v_k^{'}, v_{k+1}, \ldots, v_M)$$

where $v_i^{'}, v_j^{'}$, and $v_j^{'}$ represent the new values. Only the parameters of the membership functions may be modified; the mutation operator does not modify the function codes.

3.1.4.5 Insert and Delete Operators

The main goal of the insert and delete operators is to assist in maintaining variable-length chromosomes. They are included because of their success in other systems employing variable-length chromosomes [23]. The insert operator introduces new genetic material into the population. It generates a new rule and inserts it in the rule base, thus increasing the size of the rule base. The deletion operator deletes a randomly chosen rule, thus decreasing the number of rules in the rule base by one.

3.1.5 Constraint Satisfaction

During evolution, the mutation, insert, and creep operations may produce illegal offspring. In this research, it is desirable to omit illegal offspring in order to avoid generating meaningless rules. Past research offers four solutions to this problem. One is to introduce a penalty term that adjusts the fitness functions if a constraint is violated. Using this approach, the genetic algorithm may have to spend time evaluating illegal individuals. A second solution is to use decoders to avoid building an illegal individual, while the third repairs the chromosome in such a way that the chromosome fulfills all the constraints and constraint-preserving operators. The M-LCS investigated here uses the fourth approach, constraint-preserving operators, to avoid producing illegal chromosomes. To implement this solution, a procedure used by Setnes [69]was adapted for this study. The search constraints are coded in two vectors, $v^{max} = [v_1^{max}, v_2^{max}, \ldots, v_N^{max}]$ and $v^{min} = [v_1^{min}, v_2^{min}, \ldots v_N^{min}]$, representing the upper and lower bounds of each condition and each action. If the mutation, creep, or insert operators generate a value violating either of the bounds for v_k, it is replaced by

$$max(v_k^{min}, min(v_k, v_k^{max}))$$

where

v^{max} represents the upper bounds and v^{min} the lower bounds.

Unlike the classical GA, which can apply both crossover and mutation to the same individual during reproduction, the M-LCS uses an operator-based procedure [23] for applying operators. With this procedure, each reproduction event applies exactly one genetic operator based on some probability. Empirically, the following probabilities were found to work well: 0.02 for NOP, 0.65 for crossover, 0.1 for mutation, 0.05 for creep, 0.1 for delete and 0.08 for insert.

3.1.6 Fitness Function

This research uses accuracy-based fitness as described in Chapter 2. The fitness function used in this research was empirically determined. There are no restrictions placed on the size of a rule base. To discourage the generation of excessive rules, fitness is based on the accuracy of the rule base and the number of rules.

Accuracy, the percentage of correctly classified patterns, is computed using Eq. 3.6.

$$accuracy = \frac{1}{p} * n \tag{3.6}$$

where n is the number of correctly classified patterns. The fitness of the rule base is computed using Equation 3.7.

$$fitness = \nu * accuracy + (1 - \nu) * (\frac{1.0}{r}) \tag{3.7}$$

where ν is a weight for accuracy and and r is the number of rules in the rule base. In this research, the value for ν is 0.98.

3.1.7 Learning Algorithm

A supervised learning algorithm is used during the training mode to generate a solution to the problem. Consider a system with n inputs (conditions) and q outputs (actions). Given a training set $\mathcal{S}$ of p input-output patterns,

$$\mathcal{S} = \{(x_1, y_1), \ldots, (x_p, y_p)\}$$

where x $\in \Re^n$ and y $\in$ {1, 2, ..., q}. For this case, the algorithm consists of the following steps:

1. Generating an initial population of rule bases of variable sizes using the constraints for each condition and each action, then randomly generating the associated parameter values.

2. Evaluating the population by

   (a) Processing $\mathcal{S}$ to compute the membership degree for each rule in the rule base. For the ith rule R_i, expressed as If x_1 is A_1, ... x_n is A_n, then y_1 is C_1, ... y_q is C_q, the degree of the rule, denoted by D(R_i), is defined as follows

   $$D(R_i) = \frac{1}{n} \sum_{k=1}^{n} m_{A_{ik}}(x_k) \tag{3.8}$$

   for all patterns (x, y), where $m_{A_{ij}}(x_j)$ represents the membership function value for the jth feature of pattern x and n is the number of features.

   (b) Removing the duplicate rules from the rule base

   (c) Removing the rule with the lowest degree if two rules are inconsistent

   (d) Processing $\mathcal{S}$ to compute the fitness of each chromosome (rule base) by

      i. Selecting the next input pattern (x, y) from the training set

      ii. Computing the degree of the rule using Equation 3.8

      iii. Choosing the rule with the highest degree of data fitness, assigning this degree to f_d, and choosing the action of this rule as the output class

      iv. Calculating the error for this output: $\sqrt{(out_{actual} - out_{calculated})}^2$

      v. Determining if the error is within a specified range to ascertain if the pattern is correctly classified and adding one to the sum of correct output if it is, $s_{output} = s_{output} + 1$.

vi. Returning to (a) for the next pattern

(e) Calculating the output fitness using Equation 3.9

$$fitness_{output} = \frac{1}{p} s_{out} \tag{3.9}$$

(f) Computing the fitness using Equation 3.7 in section 3.1.6

Once the classifier has been trained, it is used in classification mode to classify unseen patterns.

3.1.8 Classification Algorithm

Let S represent the rule base generated by the learning algorithm. The following procedure is used to classify a pattern by:

1. Computing the membership degree of each rule R_i in the rule base using Equation 3.8
2. Choosing the rule with the maximum degree. The output class is the action of this rule

This mode computes the accuracy of the classifier in classifying patterns. During the development phase, the classification algorithm is used to compute accuracy for both the training and test sets.

Chapter 4

Applications of M-LCS

4.1 Introduction

To evaluate the performance of the M-LCS developed here, the framework was used to design learning classifier systems for the Fisher iris classification problem, the Pima Indians diabetes forecasting problem, the military garment sizing problem, and the alcohol classification problem. Although there are several available problems from which to choose, these were used because they represent different levels of difficulty and data set sizes. In addition, all except the alcohol problem have published results for comparison. As a first step the framework developed here was used to design classifier systems that employed only one membership function as well as an M-LCS. The results of these classifiers were compared to those reported in the literature. In addition, the results of the one-membership function classifiers were used to evaluate the relative performance of the M-LCS. The following general procedure was used for each classifier:

1. Starting with a population of randomly-initialized chromosomes, the classifier was trained until a desired fitness value was obtained or a maximum number of

generations was reached, whichever came first.

2. The trained classifier was then used in classification mode on both the training and the test sets.

3. Steps 1 - 2 were repeated ten times.

4. The average performance of the ten iterations of the training set and the average of the test set were computed.

This chapter presents the results obtained from these experiments.

4.2 Application of the M-LCS to the Fisher Iris Data

The iris classification problem is a common benchmark in classification and recognition studies [2, 31, 38, 57, 70]. This data set, obtained from the UCI Machine Learning Repository, is based on the one used by R. A. Fisher in his pioneering work on linear discriminant analysis [26]. It consists of 150 data values with four input features — sepal length, sepal width, petal length, and petal width — and three classes, each referring to a type of iris plant — *iris setosa*, *iris versicolor*, or *iris virginica*. The problem involves classifying the patterns into one of the three classes, a simple classification problem because the *setosa* class is linearly separable from the other two classes while the latter classes overlap and cannot be linearly distinguished [2].

For this study, the training and the test sets were constructed using the same method as those used by Simpson [72]. Specifically, the first twenty-five patterns of each class were selected for the training set and the remaining twenty-five for the

test set, yielding seventy-five training patterns and seventy-five test patterns. To apply the M-LCS, the population size was set to 100 chromosomes, each representing a complete rule base of if-then rules, the number of rules in a rule base ranging from one to five. The maximum number of generations was 1500. Most of the classifiers were completely trained before reaching this number, and training beyond 1500 generations caused over-training of the training set and, consequently, produced lower generalization capabilities. During training, the GA learned the number of rules, and for each, it learned the membership function for each input feature, the parameters of each membership function, and the output class for the rule.

The membership functions obtained by using the best fuzzy function classifier are shown in Figure 4.1. Only three membership functions were needed for the sepal length. Although the LCS generated five membership functions for sepal width, two of them were almost identical and could be easily combined. Similarly, two of the four functions generated for petal length and petal width could be combined. These results closely match those reported by other researchers. For example, the VISIT algorithm devised by Chang *et al.*[19] utilized two membership functions for sepal length, three for petal length, and two for petal width.

The rule base obtained by the best fuzzy function classifier is shown in Figure 4.2. The number of rules generated by this classifier also fits within the range of rules reported by other researchers. The membership functions obtained from the M-LCS are shown in Figure 4.3

The M-LCS generated the following membership functions: one trapezoidal, two γ functions, and an L function for sepal length; a trapezoidal, triangular, γ, and L function for sepal width; a γ, an L, and two fuzzy functions for petal length; and a γ, an L, a triangular, and a fuzzy function for petal width. The GA chose each membership function in its set.

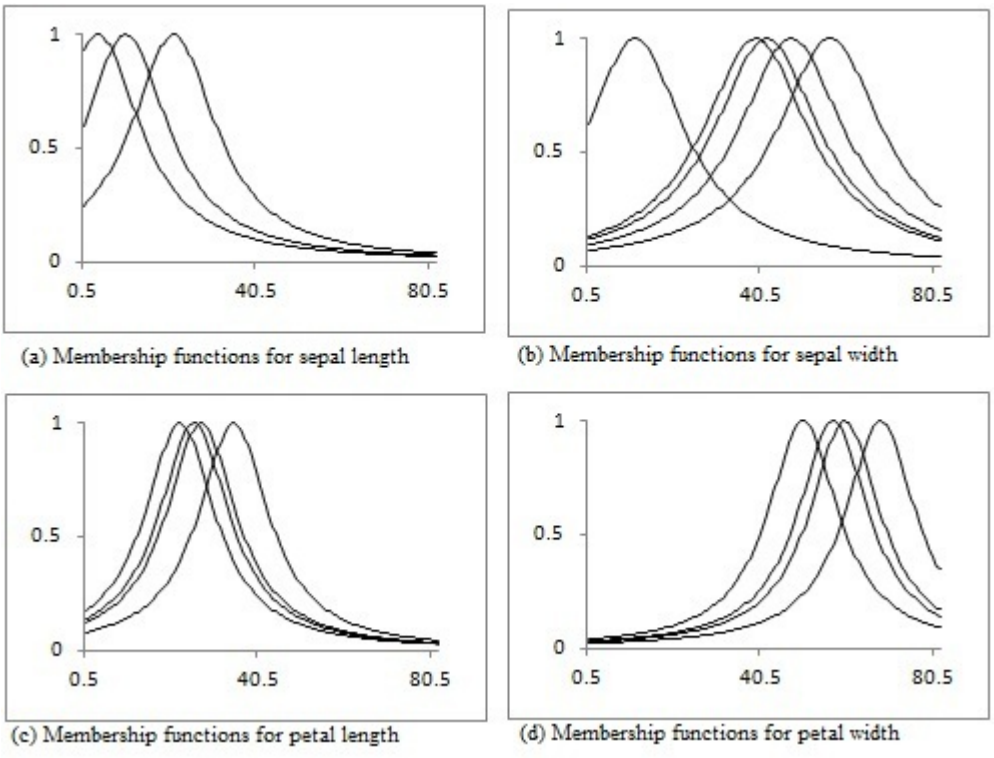

Figure 4.1: Membership sets generated by fuzzy function classifier for iris

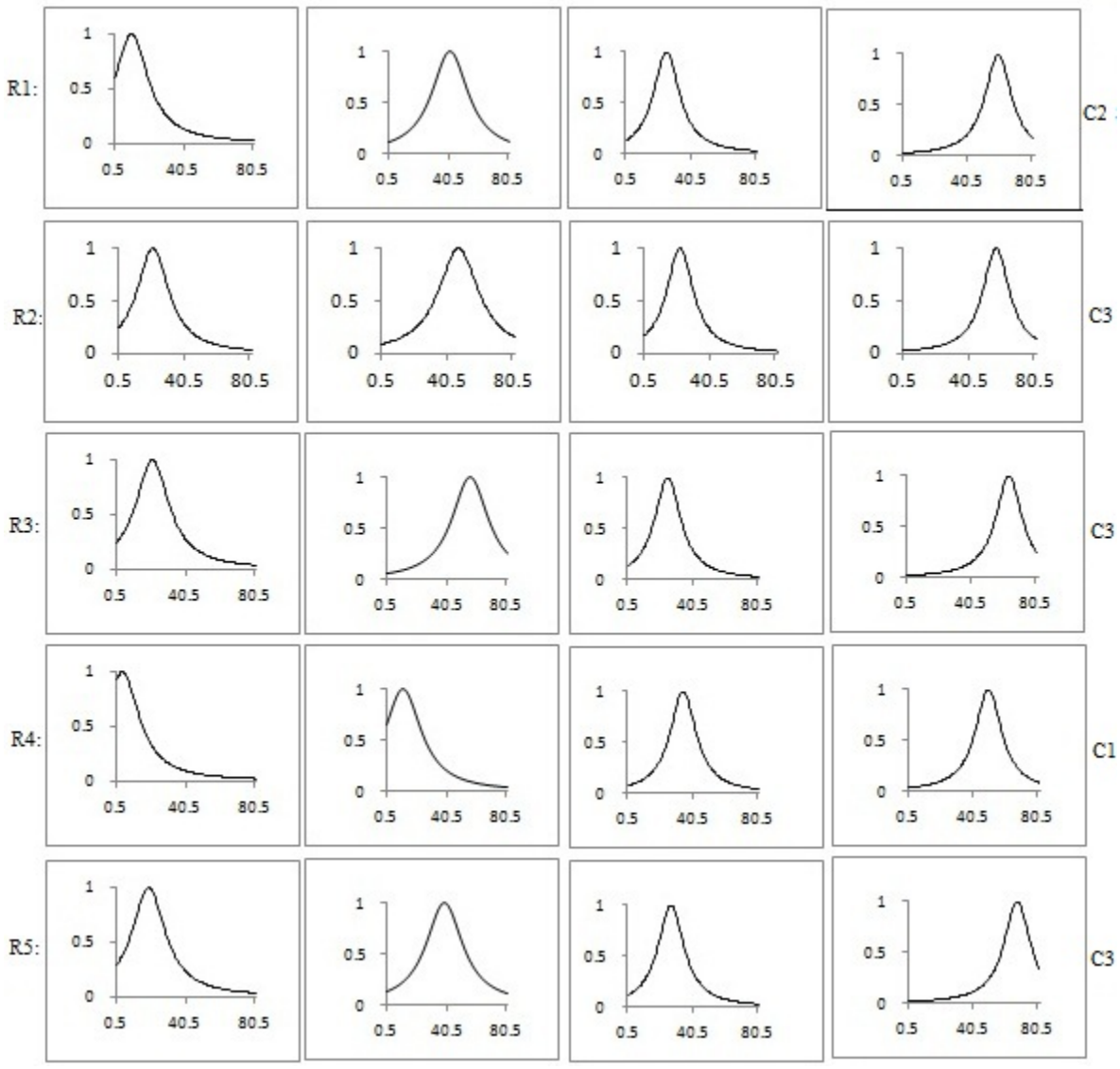

Figure 4.2: Rule base for the fuzzy function classifier for iris

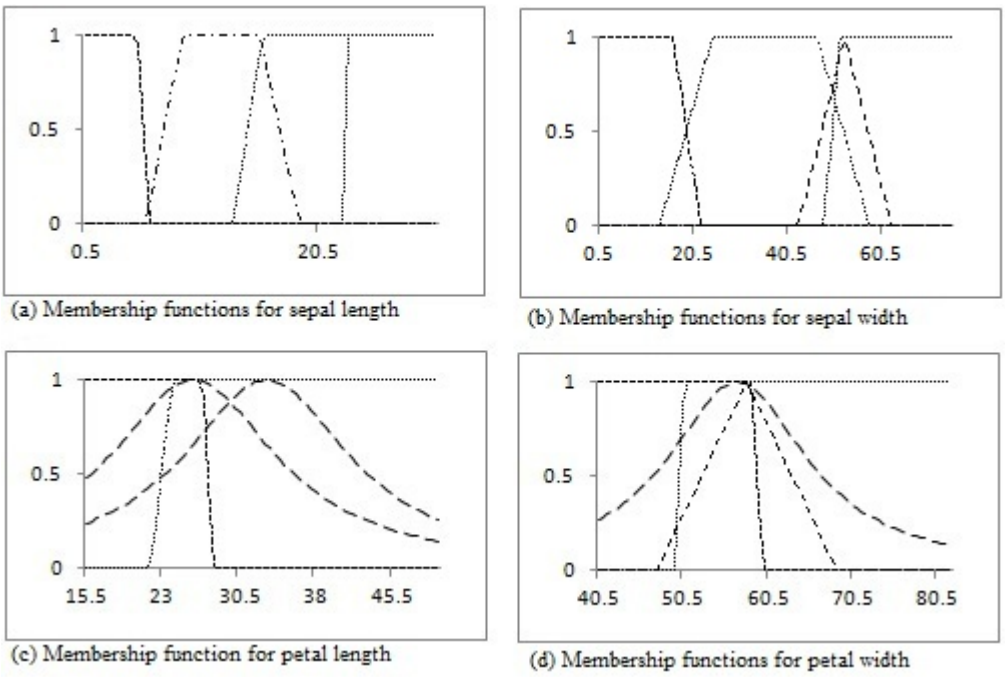

Figure 4.3: M-LCS membership functions for iris problem

The rule base having the highest average classification rate out of ten runs is shown in Figure 4.4.

As this figure shows, the best rule base generated by the M-LCS consists of four rules; however, during training, the number of rules varied from three to twelve. This number of rules is consistent with those reported by other researchers. The graphical representation of the rules in Figure 4.4 can be written in the form given in Table 4.1, where SL is sepal length, SW is sepal width, PL is petal length and PW is petal width. If linguistic sets are assigned to the membership sets, then the rules can be written in a form that is more understandable as demonstrated in Table 4.2.

To evaluate the performance of the classifiers used in this research, the performance of each was compared to those listed in Table 4.3.

The average of the ten runs for each of the single membership functions and the M-LCS for the training set and the test set at the end of the classifier training are

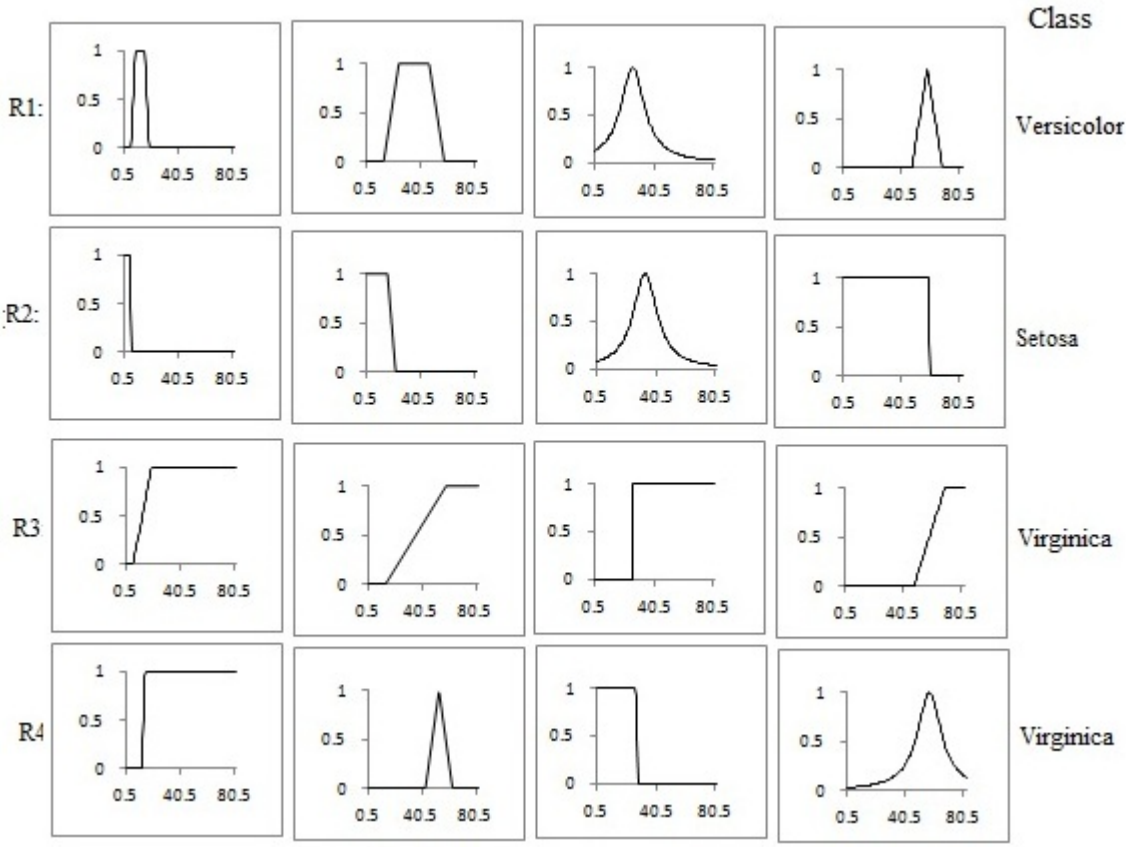

Figure 4.4: M-LCS rule base for iris problem

Table 4.1: M-LCS rules for classifying the iris plant

R1:	if SL is in the interval $[5.6, 18.77]$ and SW in the interval $[13.3, 58.1]$ and PL is close to 25.9 and PW is close to 58.0, then the class is *Versicolor*.
R2:	if SL is less than 5.0 and SW is less than 16.1 and PL is close to 33.2 and PW is less than 58.6, then the class is *Setosa*.
R3:	if SL is greater than 15.7 and SW is greater than 51.7 and PL is greater than 23.9, and PW is greater than 50.6, then the class is *Virginica*.
R4:	if SL is greater than 14.1 and SW is close to 23.0 and PL is less than 26.9 and PW is close to 56.8, then the class is *Verginica*.

Table 4.2: M-LCS linguistic rules for classifying the iris plant

R1:	if SL is narrow and SW is narrow and PL is short and PW is narrow then the class is *Versicolor*.
R2:	if SL is short and SW narrow and PL is medium and PW is narrow then the class is *Setosa*.
R3:	if SL is long and SW wide and PL is medium or long, and PW is medium or wide, then the class is *Virginica*.
R4:	if SL is medium or long and SW is narrow and PL is short and PW is narrow then the class is *Verginica*.

Table 4.3: Classification accuracy reported in literature for iris

Source	#Rules	Train Data	Test Data	Overall
Abe	-	-	97.3	98.0
Aboyni	3	-	-	96.1
Chang	5	-	-	99.3
Guidi	-	-	-	96.0
Halgamuge	13	-	98.7	-
Ishibuchi	13	-	-	100.0
Nauck	7	-	97.3	-
Liu	6	-	-	95.3
Russo	5	-	100.0	100.0
Shi	4	-	96.0	-
Setnes	3	-	98.7	-

presented in Table 4.4.

Table 4.4: Classification results obtained for the iris data

Membership Function	All Data Mean	Train Data Mean	Test Data Mean	Test Data Min	Test Data Max
Fuzzy	96.8	100	93.6	92.0	94.7
γ	96	99.1	92.9	90.7	94.7
L	94.6	99.9	89.3	76.0	94.7
Triangular	95.3	99.6	91.1	86.7	93.3
Trapezoidal	96.3	99.7	92.9	90.7	96.0
M-LCS	96.3	100	93.1	90.7	94.7

As the results indicate, the performance of each classifier is close to those reported in the literature. Although the average classification accuracy on the test data is lower than some, it is not significantly lower and, with the exception of the triangular function, the overall performance of the best classifier for each function is at least 97.3%. The overall performance of the M-LCS, having only four misclassifications, is 97.3%. Thus, the best classifiers in this research perform at or near the level of those

reported in the literature. The results of the test set provide an indication of the generalization ability of the classifiers, that is, how well the classifier will correctly classify unseen patterns. On average, the L-function classifier exhibited the lowest generalization ability, while the fuzzy-function exhibited the highest. The M-LCS performed 0.5% below the fuzzy-function classifier.

The fuzzy function classifier comparison with the average convergence speed of the different classifiers based on the percent accuracy during training can be seen in Figure 4.5. If the classifier is trained offline, then the convergence speed may not be

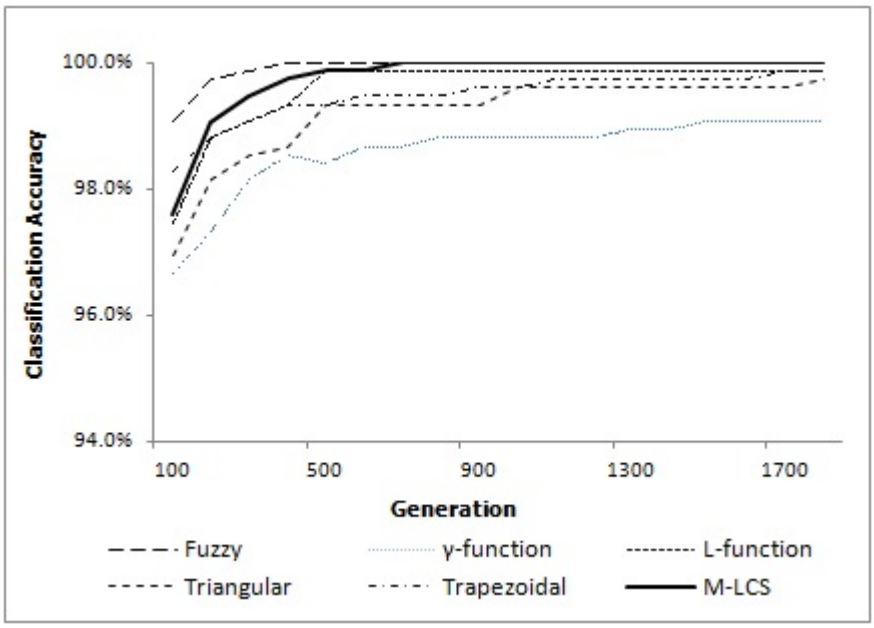

Figure 4.5: Convergence speed of M-LCS for iris

a factor; however, if training has to be done in real time, then the convergence speed of the classifier is important. On average, the fuzzy function classifier converged to the optimum solution at 400 generations, while the M-LCS converged at 700. The L-function classifier converged to its optimum solution faster than the M-LCS; however, its accuracy is lower. This function also did not generalize well, perhaps implying

overfitting of the training data. The average performance of each classifier when applied to the test set is illustrated in Figure 4.6.

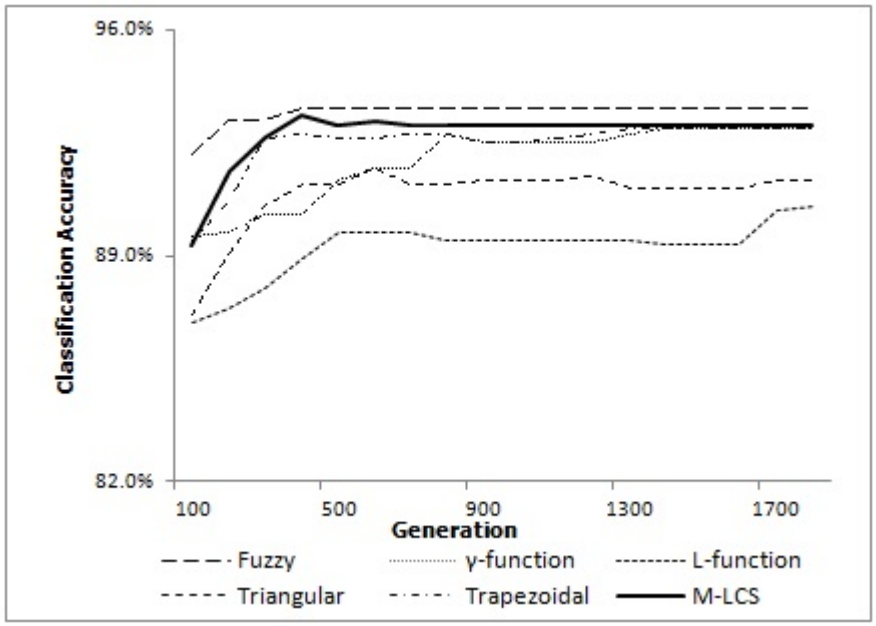

Figure 4.6: Average convergence of M-LCS for iris test data

4.3 Application of the M-LCS to the Pima Indians Diabetes Data

The Pima Indians diabetes data set, obtained from the machine-learning database at the University of California, Irvine, has been used to design systems for predicting the onset of diabetes in Pima Indians. The data consists of 768 input-output patterns. In this research, 576 of these patterns were used for training and 192 for testing. The eight features of the input are listed in Table 4.5

This classification problem, previously studied by Smith *et al.* [73] in 1988, by Abdelbar [1] in 1996, and more recently by others, has proven to be difficult. Using a

Table 4.5: Features of the Pima Indians Diabetes data set

Number of previous pregnancies
Plasma glucose concentration after 2 hours — Glucose Tolerance Test (GTT)
Diastolic blood pressure
triceps skin fold thickness
2-hour serum insulin
body mass index (weight divided by the square of the height)
diabetes pedigree function — evaluates subject's family history of diabetes
age

neural network algorithm called ADAP, Smith *et al.* obtained a classification rate of 76% on the test set; Abdelbar *et al.* used a feedforward High Order Network to obtain a classification rate of 83%. Results obtained more recently are presented in Table 4.6

Table 4.6: Literature results for Pima Indians Diabetes data

Source	Train Data	Test Data	Overall
Chang	75.8	78.2	77.0
Aboyni	-	-	73.05
Orriols	-	74.88	
MLP	-	75.8	-
RBF	-	75.7	-
Bayes	-	72.2	-
C4.5	-	72.2	-
SVM	-	77.5	-

The results of the research reported here are compared to the results obtained by other learning classifier systems as well as those based on a different learning paradigm. Chang [19], Aboyni [4], and Orriols [59] used a learning classifier system; MLP, RBF, and Bayes are based on artificial neural networks, while C4.5 is a decision tree based classifier, and SVM is a support vector machine style classifier. The results

are presented in Table 4.7.

Table 4.7: Results for the Pima Indians Diabetes data

Membership Function	Overall Mean	Train Mean	Test Mean	Test Min	Max
Fuzzy	79.7	80.6	77.2	74.5	80.7
γ	78.8	79.8	75.6	72.4	78.1
L	79.6	81.3	74.4	73.4	76.6
Triangular	78.9	80.5	74.4	72.4	76.6
Trapezoidal	80.1	81.6	75.6	72.4	77.6
M-LCS	79.3	80.4	75.9	73.4	79.2

Although M-LCS did not have the highest classification accuracy, its overall performance was only 0.8% below that of the fuzzy-function classifier, which performed the highest. On the training set, M-LCS performed 1.2% below the trapezoidal-function classifier, 1.3% below the fuzzy-function classifier on the test set, but had the highest maximum test performance. Compared to the results reported by other researchers, the classifier systems developed in this research were found to be very competitive. Although the classification accuracy of M-LCS was below that of Chang, the performance of the best M-LCS classifier was higher. The M-LCS membership functions are presented in Figure 4.8

The convergence speed of the classifiers during training is given in Figure 4.8. The graph indicates that the fuzzy function classifier peaked near generation fastest convergence speed and also the highest performance. Figure 4.9 illustrates the behavior of the classifiers on the test set during training. Although the classifiers continued to slowly improve in classifying the training data, the performance on the test data remained relatively constant. Since this is a has/has not type problem, meaning the classification is either diabetes predicted or diabetes is not predicted, it may be

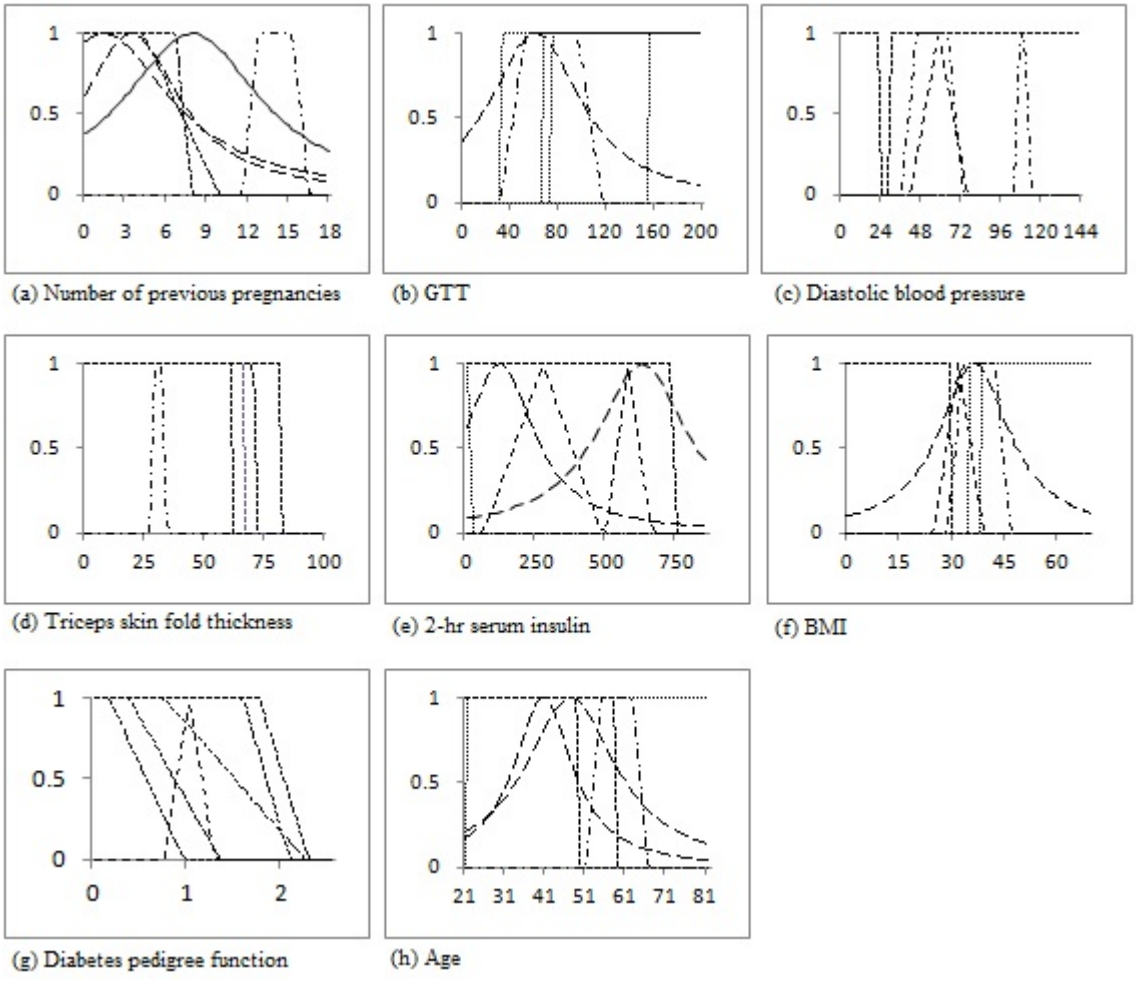

Figure 4.7: Membership sets generated by fuzzy function classifier for diabetes

Table 4.8: M-LCS rules for classifying the diabetes data

R1:	If previous pregnancy is less than 4.3 and GTT is in $[31.2, 117.13]$ and diastolic BP is greater than 28.7 and triceps skin fold thickness is less than 61.4 and 2-hour serum insulin is less than 734.8 and MI is less than 29.7 and DPF is less than 0.16 and age is greater than 21, then onset is not predicted.
R2:	If previous pregnancy is less than 7 and GTT is greater than 154.8 and diastolic BP is greater than 28.7 and triceps skin fold thickness is less than 80.3 and 2-hour serum insulin is close to 572.5 and BMI is close to 36.5 and DPF is less than 1.8 and age is close to 47, then onset is predicted.
R3:	If previous pregnancy is close to 8 and GTT is greater than 32.2 and diastolic BP is between 36.0 and 73.3 and triceps skin fold thickness is less than 70.9 and 2-hour serum insulin less than 4.7 and BMI is close to 30 and DPF is greater than 2.1 and age is close to 41, then onset is predicted.
R4:	If previous pregnancy is close to 3 and GTT is greater than 67.4 and diastolic BP is less than 22.8 and triceps skin fold thickness is less than 66.1 and 2-hour serum insulin close to 115.3 and BMI is in $[28.7, 46.9]$ and DPF is greater than 1.3 and age is in $[51, 67]$, then onset is predicted.
R5:	If previous pregnancy is close to 1.3 and GTT is close to 61.0 and diastolic BP is 58.5 and triceps skin fold thickness is less than 81.1 and 2-hour serum insulin close to 624.9 and BMI is greater than 38.5 and DPF is close to 1.0 and age is less than 49, then onset is not predicted.
R6:	If previous pregnancy is between 12 and 17 GTT is greater than 74.1 and diastolic BP is in $[104.4, 113.2]$ and triceps skin fold thickness is in $[27.6, 34.3]$ and 2-hour serum insulin close to 270.7 and BMI is greater than 35.0 and DPF is less than 0.75 and age is less than 59, then onset is predicted.

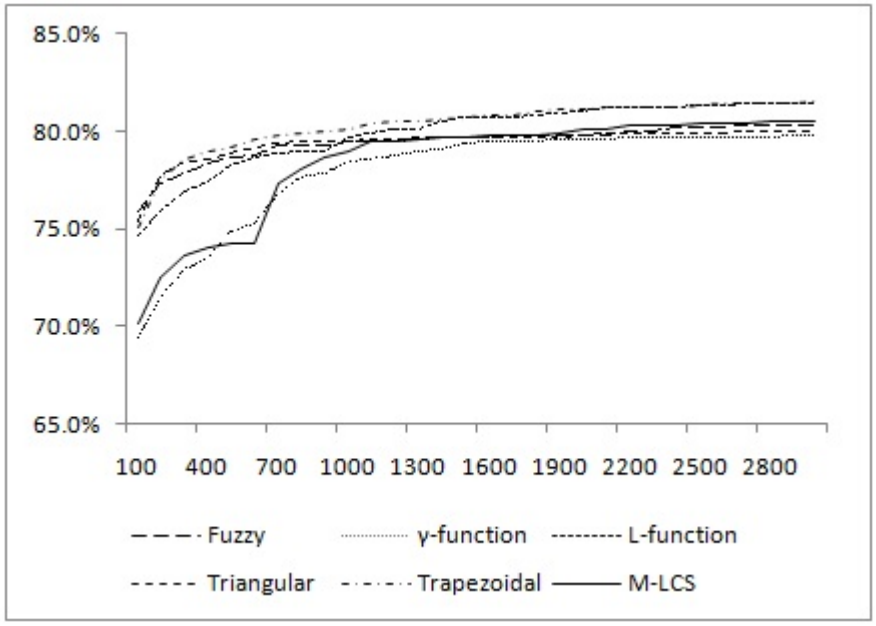

Figure 4.8: Convergence speed during training of diabetes data

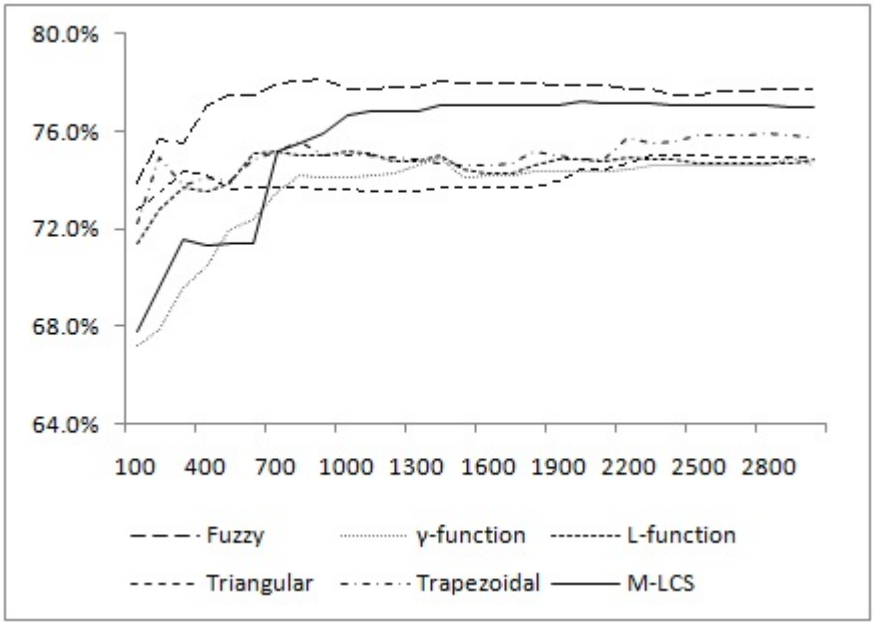

Figure 4.9: Behavior on diabetes test data during training

possible to use sensitivity and specificity measures to improve accuracy.

4.4 Application of M-LCS to the Army Data

The manual procedure for issuing clothing to military personnel, which is very labor-intensive and highly susceptible to human error, includes two phases: to the process: 1) Taking the necessary body measurements, and 2) Mapping the set of measurements to the appropriate size. Errors that occur may include the improper placement of the measuring tape, varying the tension of the tape, and human fatigue. If measurements are inaccurate, then a soldier may receive garments that do not fit, requiring additional measurements or extensive alterations. Even with accurate measurements, a fitter is still faced with the task of mapping a set of measurements to a chart to determine the correct garment size. When the set of actual measurements do not correspond exactly to the measurements on the charts, the fitter determines the correct size based on priorities among body measurements, another potential source of human error. An automated procedure can potentially provide a more accurate means of fitting army personnel and, therefore reduce, both the time involved and the number of alterations required. This problem was studied by Lowe *et al.* [50], Davis *et al.* [22], and Pargas *et al.* [62]. Lowe *et al.* provide another approach, implementing a neural network solution to the problem, testing four encoding techniques. The results of that research are presented in Tables 4.9 and 4.10.

To test the performance of the M-LCS for forecasting garment sizes, a data set obtained from the Clothing Initial Issue Facility at Fort Jackson in Columbia, SC, was used. A database of 1778 input/output patterns was separated by randomly choosing 1200 vectors for training and using the remaining vectors for testing. This

Table 4.9: Army data classification accuracy

Accuracy (%)		
Method	Train Data Mean	Test Data Mean
Frequency Distribution	98.4	66.4
Ensemble Encoding	87.3	54.4
Normalization	23.0	-
Linear Scaling	5.0	-

Table 4.10: Army data classification accuracy (within one size)

Method	Train Data Mean	Test Data Mean
Frequency Distribution	99.0	81.4
Ensemble Encoding	88.2	74.2
Normalization	28.3	-
Linear Scaling	6.7	-

research used the same eight measurements as those previously used by Lowe *et al.*: height, head size, neck size, chest size, waist size, hip size, sleeve length, and weight. During the developmental stages, classifiers were designed using different population sizes and maximum generations; however, the final version used a population size of 100 and a maximum of 2000 generations. For this number of generations, the classification accuracy increased and there was no degradation in performance on the test set. Since data representation was not the main focus of this research, the results obtained in this research are based on the raw data. The results of applying the single-function classifiers and M-LCS are presented in Table 4.11 and Table 4.12.

For this problem, the trapezoidal-function and triangular-function classifiers performed best on the training data and the complete data set. The M-LCS performed the worst, with an accuracy 4.0% below that of the triangular-function classifier.

Table 4.11: Results obtained for army data

Membership Function	All Data Mean	Train Set Mean	Test Set Mean	Test Set Min	Test Set Max
Fuzzy	38.2	38.7	37.2	31.3	46.2
γ	36.9	36.8	38.3	31.7	44.7
L	38.1	37.7	39.1	34.6	44.3
Triangular	40.0	40.0	40.1	35.8	43.3
Trapezoidal	38.1	37.2	40.0	37.3	41.4
M-LCS	37.7	36.0	41.1	35.6	48.3

In addition, as the results show, the classification accuracies of the single-function classifiers and M-LCS are low compared to the results obtained by the frequency-based and ensemble encoding networks. However, since the classifiers in this research used raw data, not pre-processed data, it is more appropriate to compare the performances to those of the networks that used linear scaling. When compared to this network, the classifiers performed 31% - 35% higher on the training data, a significant difference. Compared to the network that used normalized data, the classifiers performed 7.7% - 27.7% higher. There were no values for comparison of the performance on the test data. Comparing the results obtained for classification within one size, those of the classifiers were closer to those of the frequency and ensemble networks than the ones using linear scaling or normalization.

The convergence speed of the classifiers during training is illustrated in Figure 4.10 and the behavior of the classifiers during testing in Figure 4.11.

Membership functions generated by the M-LCS are shown in Figure 4.12. The triangular function performed the best for this application and the M-LCS chose four triangular function for weight. other function was

The relative performances of the fuzzy function classifier and the M-LCS were lower on the army data set than any other, a situation that may be attributable to the nature

Table 4.12: Results within one size for army data

Membership Function	All Data Mean	Train Set Mean	Test Set Mean	Test Set Min	Test Set Max
Fuzzy	76.5	75.8	77.9	70.5	84.6
γ	74.5	73.5	76.5	69.3	81.5
L	75.8	74.7	77.9	71.7	84.8
Triangular	77.1	76.7	77.8	73.8	81.3
Trapezoidal	78.4	77.1	81.0	80.1	82.5
M-LCS	76.5	74.0	81.6	77.9	88.2

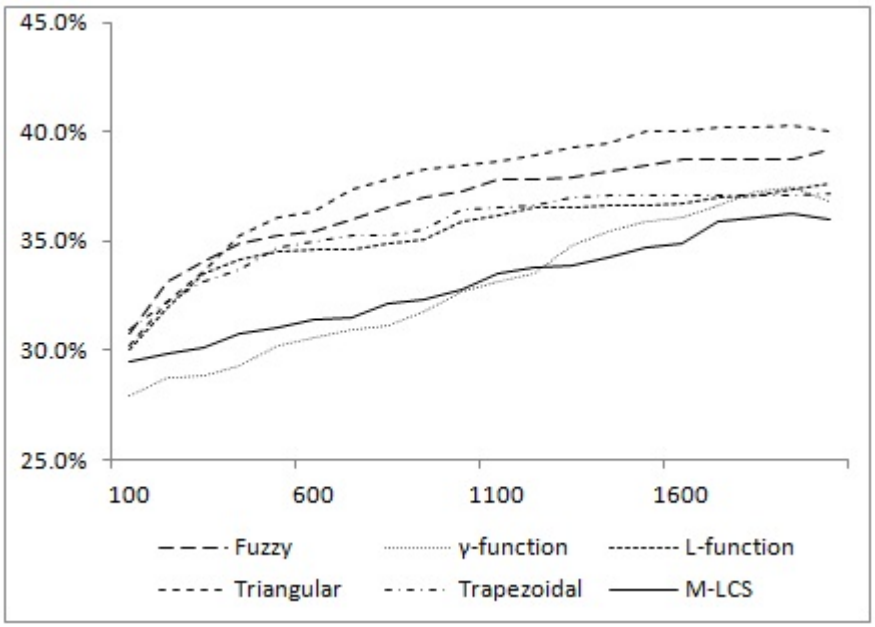

Figure 4.10: Convergence speed on army data during training

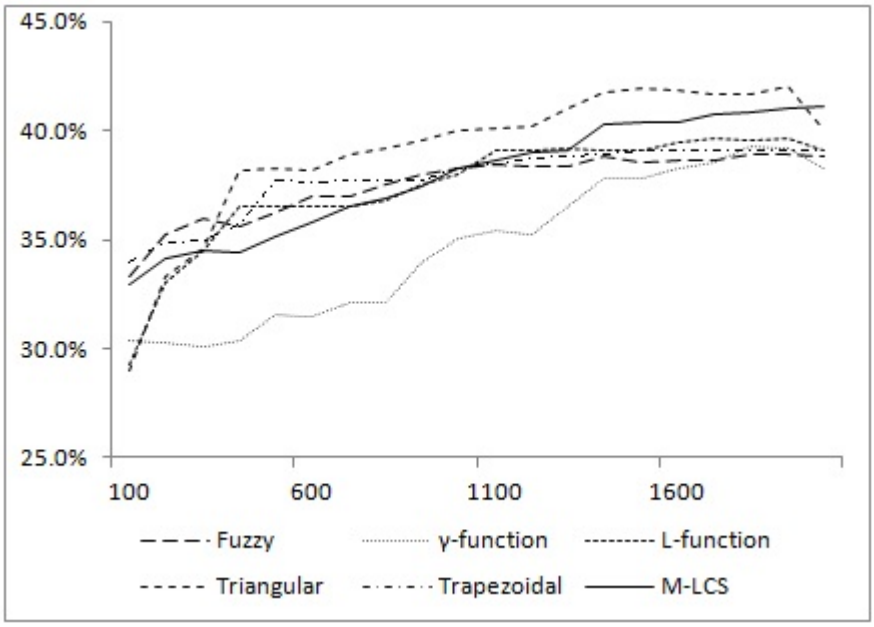

Figure 4.11: Behavior on army test set during training

of the data. Lowe *et al.* discovered that the data set contains several inconsistent patterns, which makes it difficult to classify the data. Further study is needed to determine the cause.

4.5 Application of the M-LCS to the Alcohol Data

Data for the alcohol problem was obtained from the College of Health, Education, and Human Development (HEHD) at Clemson University. This problem involves identifying specific high-risk contexts and peer reference groups at Clemson University, promoting alcohol abuse among first-year students. Data collection for the problem was supported by a National Institute on Alcohol Abuse and Alcoholism Rapid Response to College Student Drinking Cooperative Agreement Grant. This research used 633 of the patterns (475 for the training set and 158 for the test set) to classify freshmen students into one of two classes: those who will abuse alcohol

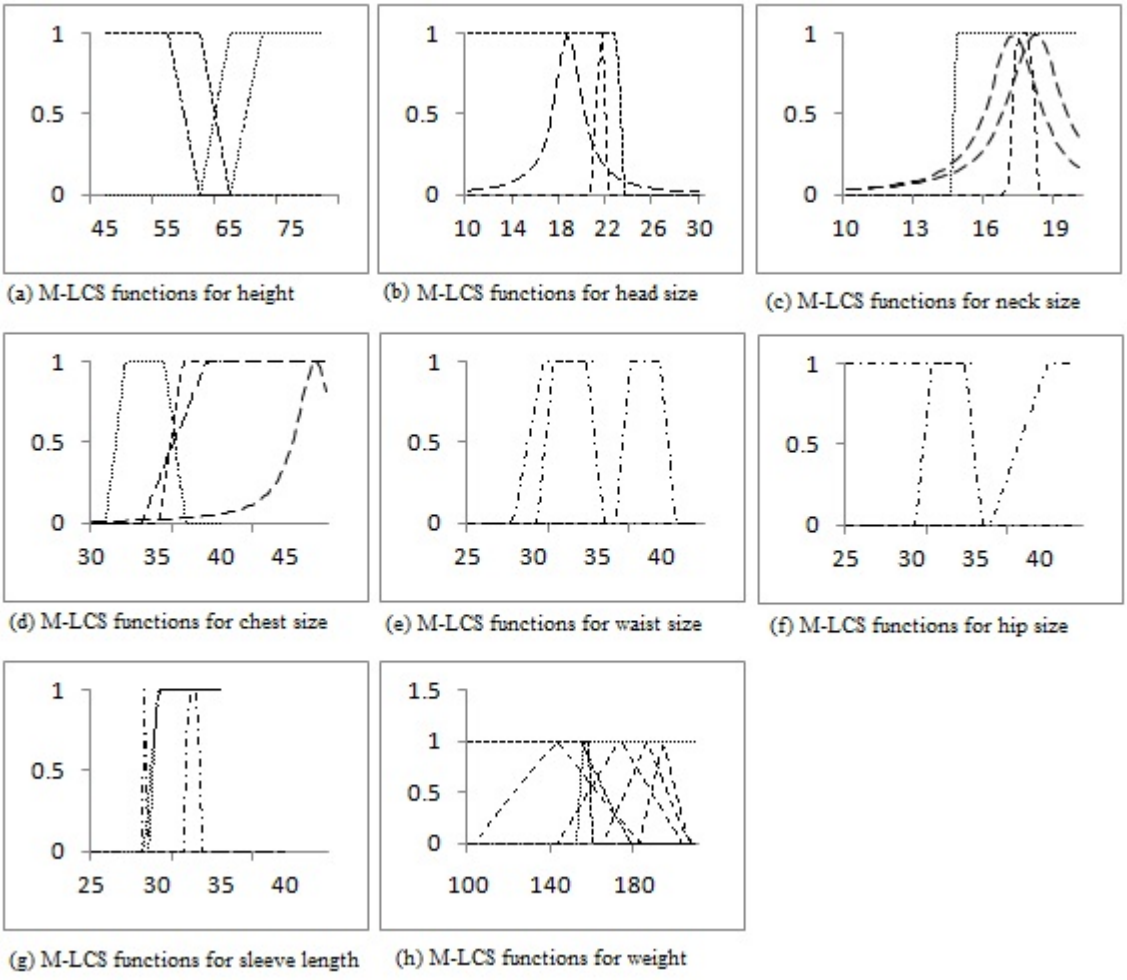

Figure 4.12: Membership functions for the army data

and those who will not. The data for the this problem consisted of 32 input features listed in Table 4.13.

Table 4.13: Input features for the alcohol problem

Age	Will student drink in college
Student drinks beer	Do parents drink
Student drinks wine	Do parents drink beer
Student drinks wine coolers	Do parents drink wine
Student drinks hard liquor	Do parents drink wine coolers
Age student started drinking	Do parents drink hard liquor
How often student drinks	How often parents drink
How many drinks when socializing	Parents average social drinks
Student gets alcohol at home	Student plans to play intramural sports
Student gets alcohol from a friend	Student plans to join a fraternity/sorority
Student purchases alcohol	Student plans to join an honorary society
Student drinks at home	Student plans to join a religious organization
Student drinks at friend's home	Student plans to play collegiate athletics
Student drinks at social events	Student plans to join campus clubs
Student drinks at other occasions	Gotten in trouble while drinking
Gender	Race/ethnicity

The problem is to classify each pattern into one of two classes: alcohol abuse and not alcohol abuse

Since this real-world problem has not been used in classification studies, there are no reported results for comparison. The performance of the M-LCS was evaluated by comparing it to the single-membership function classifiers. The data consisted of 633 input-output patterns, 475 of which were randomly selected for the training set and the remaining 158 were used for the test set. The results that were obtained through this research are reported in Table 4.14.

The alcohol problem did not present a challenge for any of the classifiers used in this study, a situation that may be attributed to the nature of the data. For many of the features, the values are either 0 or 1, while others have no more than

Table 4.14: Results of alcohol data

Membership Function	All Data Mean	Train Data	Test Data	Test Min	Test Max
Fuzzy	99.7	100.0	99.0	97.5	100
γ	99.8	100	99.1	96.8	99.4
L	99.4	99.8	98.2	96.8	99.4
Triangular	99.3	99.9	97.5	94.9	99.4
Trapezoidal	99.1	99.8	97.1	94.9	99.4
M-LCS	99.8	100	99.1	97.5	100

five discrete values. For example, do parents drink beer and does student plan to join a club have a 0 or 1 value. The feature, how often do parents drink, has five possible values: daily, weekly, monthly, less than monthly. Although all classifiers had classification accuraces above 96% on the test set, the table shows that the triangular and trapezoidal classifiers performed almost 2% below the others.

The convergence speed during training is given Figure 4.13 and the behavior during testing is in Figure 4.14. Figure 4.13

The M-LCS, γ function, and fuzzy-function classifiers were able to classify all training data after 25 generations and had a 99% accuracy on the test pattern after 25 generations. The L-function and the triangular functions reached 99.8% and 99.9% respectively at generation 500, while the trapezoidal took 800 generations to reach a peak of 99.8%. The classification accuracy was constant on the test set for the M-LCS, γ, and fuzzy functions. The other functions stabalized after generation 700.

4.6 Discussion

The results of this research, which are very encouraging, indicate that the M-LCS provides an alternative approach to designing a learning classifier system based

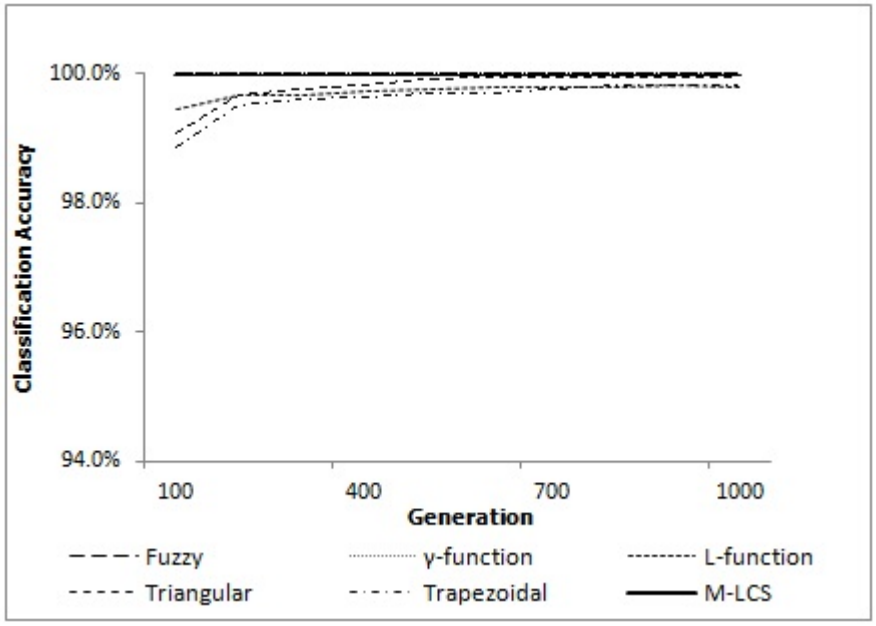

Figure 4.13: Convergence speed on alcohol data during training

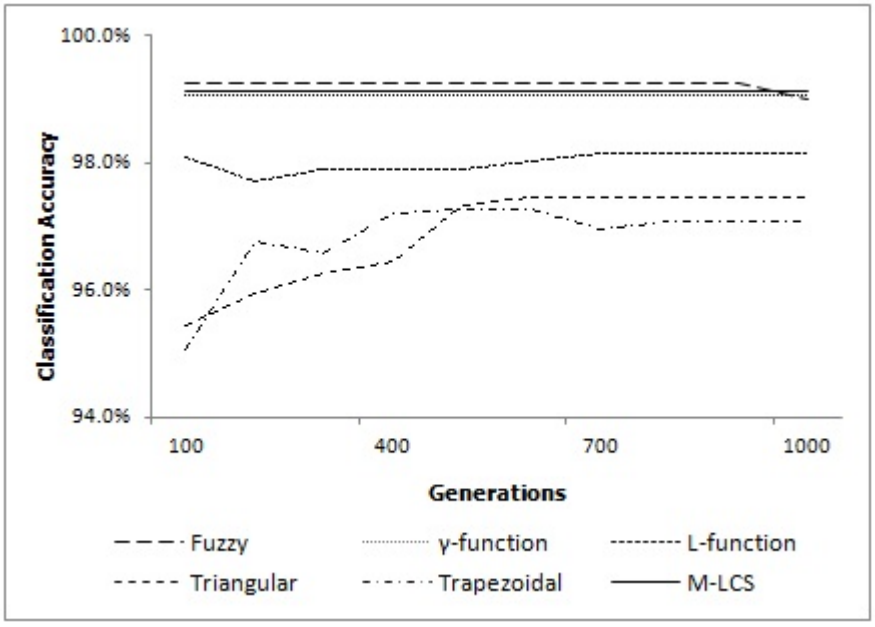

Figure 4.14: Behavior on alcohol test set during training

on the Pittsburgh model and that the fuzzy, γ, and L functions provide additional membership functions for use in the condition component of the classifiers. The fuzzy-function classifier performed well on all applications. The accuracy of the γ-function classifier was consistently lower than the others; however, the best γ classifier performed better relative to the other classifiers. The L-function classifier performed well on the training data but did not always generalize well. Even so, performance of the best L classifier generated was equal to or higher than that of the others. Although the γ- and L-function classifier did not have high accuracy, they were chosen by M-LCS for each of the classification problems used in this research. Moreover, to determine if they had a negative effect on the performance of the M-LCS, it was trained with these two functions omitted from the function pool. There was no noticeable change in the performance of the M-LCS. This finding suggests that including these functions in the pool did not cause a degradation in performance. Since a number of parameters must be determined when using a GA, altering them may improve the performance of all of the functions.

Although M-LCS did not always perform the highest on average, the best M-LCS classifier was generally the highest and when the best M-LCS classifier performed below the others, the difference was no more than 1.3%. Based on the results obtained The M-LCS reduces the need for the developer to select a membership function for representing classifiers. M-LCS was tested with five functions, but it is flexible; consequently, other functions may be substituted or added to the function pool.

Chapter 5

Conclusions and Future Work

The objective of this research has been to design and develop a framework for a classifier system that employs multiple membership functions and then use the framework to develop a learning classifier system for use in data mining. Incorporating multiple membership functions in the development of a classifier system presented new and challenging problems.

In Chapter 3, a framework for the design of a multiple membership functions learning classifier system was presented. The classifier system expands the algebra of current Pittsburgh classifier systems in two ways: 1) by increasing the number of membership functions from one to five and 2) by modifying the genetic operators to operate on chromosomes that may contain a variety of membership functions in one rule. Of the four membership functions, two ramp functions and a fuzzy membership function were introduced as part of this research. Although heuristic search does not always produce the best result, heuristic search has been successful in solving problems when an exhaustive search or a traditional method is not feasible. A genetic algorithm is the particular heuristic search algorithm used to generate the rule base for the multiple membership functions learning classifier system. The system uses the

Pittsburgh approach to represent the rule base, and variable length chromosomes, since the number of rules necessary to solve a problem depends on the problem to be solved.

To test the efficacy of the classifier system, it was applied to four problems of different sizes and complexities. Three of the problems have been used by other researchers and the fourth problem is a real-world problem that has not been used in the past. In Chapter 4 the results were presented and, when possible, the performance was compared to those of other results reported in the literature. In all cases, the results of the classifiers were encouraging.

One observation of this research is that even when the M-LCS did not improve in performance when multiple types of membership functions were used, the GA chose the membership function that performed best when used alone. This implies that it may be possible to use the M-LCS to assist in selecting a membership function when a developer desires to use only one membership function.

As a result of this research, the following contributions have been made to the field of computing:

1. a framework for developing a learning classifier system consisting of multiple types of membership functions

2. a modified learning classifier system that employs multiple types of membership functions to improve performance of the classifiers

3. the addition of three membership functions to be used in the condition part of the condition-action rules

There are several areas that have not been investigated in this work.

1. Only the Pittsburgh model was used. Future work should use the Michigan model to compare the two approaches for the M-LCS.

2. The representation is very restrictive, allowing a membership function to have only two parameters. Such a restriction forces triangular and trapezoidal functions to be designed in ways that generate regular triangle and trapezoidal shapes. General triangles and trapezoids may yield better performance. Future work will focus on alternative representation schemes, such as intervals, for representing the rules.

3. Incorporate different membership functions and experiment to determine the optimal number of functions to use.

4. Investigate the effects of using different selection and crossover algorithms and different fitness functions.

5. Apply the framework to a different suite of problems to study the characteristics of problems that lend themselves well to the framework and those that are not as suitable.

6. The framework presented here is suitable for other application areas. Future work will apply the framework to the selection of a a facial recognition algorithm, given a set of algorithms rather than using the same algorithm for recognizing each instance.

Bibliography

[1] A. Abdelbar and G. Tagliarini. Honest: a new high order feedforward neural network. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 2, pages 1257–62, New York, NY, USA, 3-6 June 1996 1996. IEEE.

[2] S. Abe and M. Lan. A method for fuzzy rules extraction directly from numerical data and its application to pattern classification. *IEEE Transactions on Fuzzy Systems*, 3(1):18–28, 02 1995.

[3] S. Abe and R. Thawonmas. Fast training of a fuzzy classifier with ellipsoidal regions. In *Part 3 (of 3), September 8, 1996 - September 11*, volume 3 of *Proceedings of the 1996 5th IEEE International Conference on Fuzzy Systems*, pages 1875–1880, New Orleans, LA, USA, 1996 1996. IEEE.

[4] J. Abonyi, J. A. Roubos, and F. Szeifert. Data-driven generation of compact, accurate, and linguistically sound fuzzy classifiers based on a decision-tree initialization. *International Journal of Approximate Reasoning*, 32(1):1–21, 01 2003.

[5] H. Ahammer, J. M. Kropfl, C. Hackl, and R. Sedivy. Image statistics and data mining of anal intraepithelial neoplasia. *Pattern Recognition Letters*, 29(16):2189–96, 12/01 2008.

[6] F. Alonso, J. Caraca-Valente, A. L. Gonzalez, and C. Montes. Combining expert knowledge and data mining in a medical diagnosis domain. *Expert Systems with Applications*, 23(4):367–75, 11 2002.

[7] P. P. Angelov and X. Zhou. Evolving fuzzy-rule-based classifiers from data streams. *IEEE Transactions on Fuzzy Systems*, 16(6):1462–1475, 2008.

[8] H. Bae, S. Kim, Y. Kim, H. L. Man, and B. W. Kwang. e-prognosis and diagnosis for process management using data mining and artificial intelligence. In *29th Annual Conference of the IEEE Industrial Electronics Society*, volume 3 of *IECON'03*, pages 2537–42, Piscataway, NJ, USA, 2-6 Nov. 2003 2003. IEEE.

[9] A. J. Bagnall and G. C. Cawley. Learning classifier systems for data mining: a comparison of xcs with other classifiers for the forest cover data set. In *2003*

International Joint Conference on Neural Networks, volume 3, pages 1802–7, Piscataway, NJ, USA, 20-24 July 2003 2003. IEEE.

[10] A. L. C. Bazzan. Agents and data mining in bioinformatics: joining data gathering and automatic annotation with classification and distributed clustering. In *4th International Workshop, ADMI 2009*, Agents and Data Mining Interaction, pages 3–20, Berlin, Germany, 10-15 May 2009 2009. Springer-Verlag.

[11] P. Bertone and M. Gerstein. Integrative data mining: the new direction in bioinformatics. *IEEE Engineering in Medicine and Biology Magazine*, 20(4):33–40, 07 2001.

[12] V. Bogorny, B. Kuijpers, and L. O. Alvares. St-dmql: a semantic trajectory data mining query language. *International Journal of Geographical Information Science*, 23(10):1245–76, 10 2009.

[13] L. B. Booker, D. E. Goldberg, and J. H. Holland. Classifier systems and genetic algorithms. *Artificial Intelligence*, 40(1-3):235–282, 1989.

[14] T. A. Braun, T. E. Scheetz, G. Webster, A. Clark, E. M. Stone, V. C. Sheffield, and T. L. Casavant. Identifying candidate disease genes with high-performance computing. *Journal of Supercomputing*, 26(1):7–24, 08 2003.

[15] O. Buchtala and B. Sick. Goodness of fit: Measures for a fuzzy classifier. In *2007 IEEE Symposium on Foundations of Computational Intelligence, FOCI 2007, April 1, 2007 - April 5*, pages 201–207, Honolulu, HI, United states, 2007 2007. Inst. of Elec. and Elec. Eng. Computer Society.

[16] L. Bull. On lookahead and latent learning in simple lcs. In *9th Annual Genetic and Evolutionary Computation Conference, GECCO 2007, July 7, 2007 - July 11*, pages 2633–2636, London, United kingdom, 2007 2007. Association for Computing Machinery.

[17] L. Bull, P. L. Lanzi, and T. O'hara. Anticipation mappings for learning classifier systems. In *2007 IEEE Congress on Evolutionary Computation, CEC 2007, September 25, 2007 - September 28*, pages 2133–2140, Singapore, 2007 2008. Inst. of Elec. and Elec. Eng. Computer Society.

[18] M. G. Ceruti. The relationship between artificial intelligence and data mining: application to future military information systems. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, volume 3, page 1875 vol.3, Piscataway, NJ, USA, 8-11 Oct. 2000 2000. IEEE.

[19] X. Chang and J. H. Lilly. Evolutionary design of a fuzzy classifier from data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(4):1894–1906, August 2004 2004.

[20] R. Cheng, M. Gen, and T. Tozawa. Vehicle routing problem with fuzzy due-time using genetic algorithms. *Japanese Journal of Fuzzy Theory and Systems*, 7(5):665–79, 1995.

[21] C. Darwin. *The origin of species by means of natural selection.* London, Watts, 1929.

[22] J. S. Davis, N. Staples, and R. Pargas. Design of a system to predict garment sizes. *New Review of Applied Expert Systems*, 4:17–31, 1998.

[23] L. Davis. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991.

[24] S. Dixon and X. Yu. Bioinformatics data mining using artificial immune systems and neural networks. In *2010 International Conference on Information and Automation (ICIA 2010)*, pages 440–5, Piscataway, NJ, USA, 20-23 June 2010 2010. IEEE.

[25] C. J. Enderli. Feature extraction using polynomial and sigmoidal kernels for classification of radar sar images. In *2006 CIE International Conference on Radar, ICR 2006, October 16, 2006 - October 19*, Shanghai, China, 2006 2007. Institute of Electrical and Electronics Engineers Inc.

[26] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(part II):179–188, 1936.

[27] O. Folorunso and A. O. Ogunde. Data mining as a technique for knowledge management in business process redesign. *Information Management and Computer Security*, 13(4):274–80, 2005.

[28] L. Y. Fong and K. Y. Szeto. Fuzzy adaptive rules in the forecasting of short memory time series. In *Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, volume 1, pages 598–603, Piscataway, NJ, USA, 25-28 July 2001 2001. IEEE.

[29] Y. Francis, H. Y. Chan, F. K. Lam, and Hung Nguyen. New fuzzy classifier with triangular membership functions. In *Part 4 (of 4), June 9, 1997 - June 12*, volume 1 of *Proceedings of the 1997 IEEE International Conference on Neural Networks*, pages 479–482, Houston, TX, USA, 1997 1997. IEEE.

[30] C. Franke, F. Hoffmann, J. Lepping, and U. Schwiegelshohn. Development of scheduling strategies with genetic fuzzy systems. *Applied Soft Computing Journal*, 8(1):706–21, 01 2008.

[31] S. K. Halgamuge and M. Glesner. Neural networks in designing fuzzy systems for real world applications. *Fuzzy Sets and Systems*, 65(1):1–12, 07/11 1994.

[32] D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. MIT Press, Cambridge, Massachusetts, 2001.

[33] J. Holland. Adaptation in natural and artificial systems, 1975.

[34] J. H. Holmes, D. R. Durbin, and F. K. Winston. The learning classifier system: An evolutionary computation approach to knowledge discovery in epidemiologic surveillance. *Artificial Intelligence in Medicine*, 19(1):53–74, 2000.

[35] M. L. Huang and Q. V. Nguyen. Context visualization for visual data mining. In *Visual Data Mining - Theory*, volume 4404 LNCS, pages 248–263, Tiergartenstrasse 17, Heidelberg, D-69121, Germany, Techniques and Tools for Visual Analytics 2008. Springer Verlag.

[36] B. Indranil and R. K. Mahapatra. Business data mining-a machine learning perspective. *Information and Management*, 39(2):211–25, 12 2001.

[37] N. Inuzuka and T. Makino. Implementing multi-relational mining with relational database systems. In *13th International Conference, KES 2009*, Knowledge-Based and Intelligent Information and Engineering Systems, pages 672–80, Berlin, Germany, 28-30 Sept. 2009 2009. Springer.

[38] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka. Construction of fuzzy classification systems with rectangular fuzzy rules using genetic algorithms. *Fuzzy Sets and Systems*, 65(2-3):237–53, 08/10 1994.

[39] L. Jikeng, W. Xudong, and S. K. Tso. Ann-based data mining for the detection of the most influential variables causing mismatch of super-heater outlet temperatures in a thermo-plant. In *CyberC 2009*, 2009 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, pages 5–11, Piscataway, NJ, USA, 10-11 Oct. 2009 2009. IEEE.

[40] H. Jin and H. Liu. Research on visualization techniques in data mining. In *2009 International Conference on Computational Intelligence and Software Engineering, CiSE 2009, December 11, 2009 - December 13*, Wuhan, China, 2009 2009. IEEE Computer Society.

[41] C. Joung, H. Kang, and K. Sim. Fuzzy classifier system for edge detection. In *1999 IEEE International Conference on Systems, Man, and Cybernetics 'Human Communication and Cybernetics', October 12, 1999 - October 15*, volume 4, pages IV–911 – IV–915, Tokyo, Jpn, 1999 1999. IEEE.

[42] P. R. Kersten. The fuzzy quadratic classifier. In *Proceedings of IEEE 5th International Fuzzy Systems*, New York, NY, USA, 8-11 Sept. 1996 1996. IEEE.

[43] B. Krishnamurthy, T. Malik, S. Stamatis, V. Venkatasubramanian, and J. Caruthers. Rule-based classification systems for informatics. In *2008 IEEE Fourth International Conference on eScience*, pages 420–1, Piscataway, NJ, USA, 7-12 Dec. 2008 2008. IEEE.

[44] P. L. Lanzi. An analysis of generalization in xcs with symbolic conditions. In *2007 IEEE Congress on Evolutionary Computation, CEC 2007, September 25, 2007 - September 28*, pages 2149–2156, Singapore, 2007 2008. Inst. of Elec. and Elec. Eng. Computer Society.

[45] P. L. Lanzi. Learning classifier systems: Then and now. *Evolutionary Intelligence*, 1(1):63–82, 2008.

[46] P. L. Lanzi, S. Rocca, and S. Solari. An approach to analyze the evolution of symbolic conditions in learning classifier systems. In *9th Annual Genetic and Evolutionary Computation Conference, GECCO 2007, July 7, 2007 - July 11*, pages 2795–2800, London, United kingdom, 2007 2007. Association for Computing Machinery.

[47] H. Lee, K. Chen, and I. Jiang. A neural network classifier with disjunctive fuzzy information. *Neural Networks*, 11(6):1113–1125, 1998.

[48] M. Lee and C. Yang. A novel data mining algorithm and knowledge-based diagnostic rules for medical thermograph. *Biomedical Engineering, Applications Basis Communications*, 21(1):1–8, 02 2009.

[49] H. Lodhi, S. Muggleton, and M. J. Sternberg. Learning large margin first order decision lists for multi-class classification. In *12th International Conference on Discovery Science, DS 2009, October 3, 2009 - October 5*, volume 5808 LNAI, pages 168–183, Porto, Portugal, 2009 2009. Springer Verlag.

[50] R. M. Lowe and E. W. Page. Application of neural networks to garment size prediction. In *Joint Conference of Information Sciences*, pages 176–179, March 1997 1997.

[51] S. A. Maelainin and A. Bensaid. Fuzzy data mining query language. In *KES '98*, volume 1 of *Proceedings of Second International Conference on Conventional and Knowledge-Based Intelligent Electronic Systems*, pages 335–40, New York, NY, USA, 21-23 April 1998 1998. IEEE.

[52] S. Z. Mahmoodabadi, A. Ahmadian, M. D. Abolhassani, J. Alireazie, and P. Babyn. Ecg arrhythmia detection using fuzzy classifiers. In *NAFIPS 2007: 2007 Annual Meeting of the North American Fuzzy Information Processing Society, June 24, 2007 - June 27*, pages 48–53, San Diego, CA, United states, 2007 2007. Institute of Electrical and Electronics Engineers Inc.

[53] D. Mellor. A first order logic classifier system. In *GECCO 2005 - Genetic and Evolutionary Computation Conference, June 25, 2005 - June 29*, pages 1819–1826, Washington, D.C., United states, 2005 2005. Association for Computing Machinery.

[54] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Stringer-Verlag, New York, 1994.

[55] V. Mikovic and M. Milosavljevic. Application of symbolic inductive learning methods to gene expression analyses. In *9th Symposium on Neural Network Applications in Electrical Engineering, NEUREL 2008, September 25, 2008 - September 27*, pages 99–102, Belgrade, Rs, 2008 2008. Inst. of Elec. and Elec. Eng. Computer Society.

[56] J. Moreno-Torres, X. Llora, and D. E. Goldberg. Binary representation in gene expression programming: towards a better scalability. In *2009 Ninth International Conference on Intelligent Systems Design and Applications (ISDA 2009)*, pages 1441–4, Piscataway, NJ, USA, 30 Nov.-2 Dec. 2009 2009. IEEE.

[57] D. Nauck and R. Kruse. A neuro-fuzzy method to learn fuzzy classification rules from data. *Fuzzy Sets and Systems*, 89(3):277–88, 08/01 1997.

[58] N. Navaroli, D. Turner, A. I. Concepcion, and R. S. Lynch. Performance comparison of adrs and pca as a preprocessor to ann for data mining. In *2008 Eighth International Conference on Intelligent Systems Design and Applications*, volume 1, pages 47–52, Piscataway, NJ, USA, 26-28 Nov. 2008 2008. IEEE.

[59] A. Orriols-Puig, J. Casillas, and E. Bernado-Mansilla. Fuzzy-ucs: A michigan-style learning fuzzy-classifier system for supervised learning. *IEEE Transactions on Evolutionary Computation*, 13(2):260–83, 04 2009.

[60] A. Orriols-Puig, X. Llora, and D. E. Goldberg. How xcs deals with rarities in domains with continuous attributes. In *12th Annual Genetic and Evolutionary Computation Conference, GECCO-2010, July 7, 2010 - July 11*, pages 1–8, Portland, OR, United states, 2010 2010. Association for Computing Machinery.

[61] C. Ouyang and S. Lee. Knowledge acquisition from input-output data by fuzzy-neural systems. In *Part 2 (of 5), October 11, 1998 - October 14*, volume 2 of *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1928–1933, San Diego, CA, USA, 1998 1998. IEEE.

[62] R. P. Pargas, N. J. Staples, and J. S. Davis. Automatic measurement extraction for apparel from a three-dimensional body scan. *Optics and Lasers in Engineering*, 28(2):157–172, 1997.

[63] Y. Peng, G. Kou, Y. Shi, and Z. Chen. A descriptive framework for the field of data mining and knowledge discovery. *International Journal of Information Technology and Decision Making*, 7(4):639–682, 2008.

[64] H. Polheim. Geatbx: Genetic and evolutionary toolbox for use with matlab. Technical report, 1995-2005.

[65] V. Rayward-Smith. Statistics to measure correlation for data mining applications. *Computational Statistics and Data Analysis*, 51(8):3968–82, 05/01 2007.

[66] C. Reddy and M. Doshi. Effects of combining classifiers.

[67] D. Rivero, J. Rabunal, J. Dorado, and A. Pazos. Automatic design of anns by means of gp for data mining tasks: iris flower classification problem. In *Proceedings, Part I*, Adaptive and Natural Computing Algorithms. 8th International Conference, ICANNGA 2007, pages 276–85, Berlin, Germany, 11-14 April 2007 2007. Springer-Verlag.

[68] J. Seng and T. C. Chen. An analytic approach to select data mining for business decision. *Expert Systems with Applications*, 37(12):8042–57, 12 2010.

[69] M. Setnes and H. Roubos. Transparent fuzzy modeling using fuzzy clustering and gas. In *Proceedings of NAFIPS-99: 18th International Conference of the North American Fuzzy Information Processing Society*, pages 198–202, Piscataway, NJ, USA, 10-12 June 1999 1999. IEEE.

[70] M. Setnes and H. Roubos. Ga-fuzzy modeling and classification: complexity and performance. *IEEE Transactions on Fuzzy Systems*, 8(5):509–22, 10 2000.

[71] M. Shetty-Wagoner, K. S. Rattan, and R. Siferd. Membership function generator circuit for a fuzzy logic controller. In *Proceedings of NAECON '93 - National Aerospace and Electronics Conference*, volume 1, pages 48–53, New York, NY, USA, 24-28 May 1993 1993. IEEE.

[72] P. K. Simpson. Fuzzy min-max neural networks. i. classification. *IEEE Transactions on Neural Networks*, 3(5):776–86, 1992.

[73] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings - Twelfth Annual Symposium on Computer Applications in Medical Care, November 6, 1988 - November 9*, pages 261–265, Washington, DC, USA, 1988 1988. Publ by IEEE.

[74] C. Soares, A. M. Jorge, and M. A. Domingues. Monitoring the quality of metadata in web portals using statistics, visualization and data mining. In *12th*

Portuguese Conference on Artificial Intelligence, EPIA 2005 - Progress in Artificial Intelligence, December 5, 2005 - December 8, volume 3808 LNCS, pages 371–382, Covilha, Portugal, 2005 2005. Springer Verlag.

[75] K. Taboada, S. Mabu, E. Gonzales, K. Shimada, and K. Hirasawa. Fuzzy classification rule mining based on genetic network programming algorithm. In *SMC 2009*, 2009 IEEE International Conference on Systems, Man and Cybernetics, pages 3860–5, Piscataway, NJ, USA, 11-14 Oct. 2009 2009. IEEE.

[76] K. Toh. Training a reciprocal-sigmoid classifier by feature scaling-space. *Machine Learning*, 65(1):273–308, 10 2006.

[77] S. Tsumoto, K. Matsuoka, and S. Yokoyama. Application of data mining to medical risk management. In *Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2008*, volume 6973, pages 697308–1, USA, 03/16 2008. SPIE - The International Society for Optical Engineering.

[78] M. Usman, S. Asghar, and S. Fong. A conceptual model for combining enhanced olap and data mining systems. In *2009 Fifth International Joint Conference on INC, IMS and IDC*, pages 1958–63, Piscataway, NJ, USA, 25-27 Aug. 2009 2009. IEEE.

[79] M. Wang, X. Zhou, and T. Chua. Automatic image annotation via local multi-label classification. In *2008 International Conference on Image and Video Retrieval, CIVR 2008, July 7, 2008 - July 9*, pages 17–26, Niagara Falls, ON, United states, 2008 2008. Association for Computing Machinery.

[80] S. W. Wilson. Generalization in the xcs classifier system. In *Proceedings of Genetic Programming Conference (GP-98)*, pages 665–74, San Francisco, CA, USA, 22-25 July 1998 1998. Morgan Kaufmann Publishers.

[81] S. W. Wilson. Mining oblique data with xcs. In *Revised Papers*, Advances in Learning Classifier Systems. Third International Workshop, IWLCS 2000, pages 158–74, Berlin, Germany, 15-16 Sept. 2000 2001. Springer-Verlag.

[82] S. W. Wilson. Classifiers that approximate functions. *Natural Computing*, 1(2-3):211–33, 2002.

[83] H. Xu, S. Mannor, and C. Caramanis. Sparse algorithms are not stable: a no-free-lunch theorem. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 1299–303, Piscataway, NJ, USA, 23-26 Sept. 2008 2008. IEEE.

[84] L. Yao, K. Weng, and R. Chang. A fuzzy classifier with directed initialization adaptive learning of norm inducing matrix. In *2009 First Asian Conference on*

Intelligent Information and Database Systems, ACIIDS, pages 226–31, Piscataway, NJ, USA, 1-3 April 2009 2009. IEEE.

[85] S. Zhang and J. Zhang. A new classification mining model based on the data warehouse. In *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics*, volume 1, pages 168–71, Piscataway, NJ, USA, 2-5 Nov. 2003 2003. IEEE.