Neural Fuzzy Techniques
In Vehicle Acoustic Signal Classification

Somkiat Sampan

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Electrical Engineering

Hugh F. VanLandingham, Chair
William T. Baumann
John S. Bay
Robert D. James
Jeffrey H. Reed
John F. Rossi

April 23, 1997
Blacksburg, Virginia

Keywords: Acoustic Signal Classification, Circular Array, Modified Genetic Algorithm, Multilayer Perceptron, Adaptive Fuzzy Logic System, Balance of Area Defuzzification

Neural Fuzzy Techniques in Vehicle Acoustic Signal Classification

by

Somkiat Sampan

(ABSTRACT)

Vehicle acoustic signals have long been considered as unwanted traffic noise. In this research acoustic signals generated by each vehicle will be used to detect its presence and classify its type. Circular arrays of microphones were designed and built to detect desired signals and suppress unwanted ones. Circular arrays with multiple rings have an interesting and important property that is constant sidelobe levels. A modified genetic algorithm that can work directly with real numbers is used in the circular array design. It offers more effective ways to solve numerical problems than a standard genetic algorithm.

In classifier design two main paradigms are considered: multilayer perceptrons and adaptive fuzzy logic systems. A multilayer perceptron is a network inspired by biological neural systems. Even though it is far from a biological system, it possesses the capability to solve many interesting problems in variety fields. Fuzzy logic systems, on the other hand, were inspired by human capabilities to deal with fuzzy terms. Its structures and operations are based on fuzzy set theory and its operations. Adaptive fuzzy logic systems are fuzzy logic systems equipped with training algorithms so that its rules can be extracted or modified from available numerical data similar to neural networks. Both fuzzy logic systems and multilayer perceptrons have been proved to be universal function approximators. Since there are approximations in almost every stage, both of these system types are good candidates for classification systems.

In classification problems unequal learning of each class is normally encountered. This unequal learning may come from different learning difficulties and/or unequal numbers of training data from each class. The classifier tends to classify better for a well-learned class while doing poorly for other classes. Classification costs that may be different from class to class can be used to train and test a classifier. An error backpropagation algorithm can be modified so that the classification costs along with unequal learning factors can be used to control classifier learning during its training phase.

Acknowledgments

I would like to express my deepest gratitude and thanks to my advisor, Dr. Hugh F. VanLandingham, for his time, advice, support, and guidance throughout my dissertation research at Virginia Tech.

I would like to give my sincere thanks to Mr. Robert James, at The Center for Transportation Research, for his constant supports throughout my research. I would also like to thank staffs at CTR for great supports and opportunities. I also want to thank Mr. Clark and signal crews of VDOT for constant supports. Without their helps, the sensor installation would be impossible.

I would like to extend my gratitude to Dr. W.T. Baumann, Dr. J.S. Bay, Dr. J.H. Reed and Dr. J.F. Rossi for their excellent lectures during my program, helpful advice and comments for developing this dissertation, and for serving on my committee.

I would like to give my special thanks to The Royal Thai Army for sponsorships throughout my undergraduate and graduate studies. Without the supports, I would not have any chance to come and study abroad.

I want to dedicate all my works for my mother who passed away last August. You sacrificed everything including your life for my success. I will dedicate and do good works for you for the rest of my life. I would like to give my special thanks to my mother and father in law and Mr. and Mrs. Ittipatanun for great supports.

Finally, I would like to share this success with my very special people, my spouse and son, who provided me the encouragement and companion throughout this research. Thank to both of you for great supports and understandings.

Table of Contents

List of Figures viii

List of Table xii

1. Introduction 1

1.1 Background and motivation 1

1.2 Survey of previous works 4

1.2.1. Acoustic pattern classification 4

1.2.2. Conventional and artificial neural networks classifiers 9

1.3 Objectives and Scope of Research 13

1.3.1 Scope of the research 13

1.4 Contributions 14

2. Acoustic Sensor Design and Data Acquisition 15

2.1 Acoustic Sensor Design 16

2.1.1 Circular Array of Microphones 17

2.1.2 Genetic Algorithms (GAs) 22

2.1.3 Modified Genetic Algorithm 25

2.1.4 Two-Dimensional Optimization Results 30

2.1.5 Circular Array Design Using Modified Genetic Algorithms 35

2.1.6 Sensitivity Study 40

2.1.7 Prototype Circular Array Implementation 44

2.1.8 Approximate Field Test Beamforming 52

2.2 Data Collection 56

3. Feature Extraction and Analysis **58**

3.1 Preprocessing (Filtering) 58

3.2 Vehicle Detection Algorithm 62

3.2 Feature Analysis and Extraction 69

3.2.1 Feature Extraction 70

3.2.2 Principal Component Analysis 73

3.3 Training and Testing Data 75

4. Classifier Designs **76**

4.1 Pattern Classification with Discriminant Functions 76

4.2 *k*-Nearest Neighbor (*k*-NN) 77

4.3 Multilayer Perceptrons (MLP) 80

4.3.1 Overview 80

4.3.2 Artificial Neuron 81

4.3.3 Training Algorithm 85

4.4 Fuzzy Logic Systems (FLS) 87

4.5 Adaptive Fuzzy Logic Systems (AFLS) 93

4.5.1 MIMO Adaptive Fuzzy Logic System with Center Average Defuzzification 94

4.5.1.1 Overview 94

4.5.1.2 Training Procedure MIMO-AFLS with Center Average Defuzzification 96

4.5.2 MIMO Adaptive Fuzzy Logic System with Balance of Area Defuzzification 98

4.5.2.1 Overview 98

4.5.2.2 Training procedure for MIMO-AFLS with BOA defuzzifier 106

5. Unequal Learning of Classifiers **110**

5.1 Overview 110

5.2 Unequal Learning of each Class 110

5.3 Unequal Number of Data 112

5.4 Unequal Misclassification Costs among Different Classes 113

5.5 Learning Factor Due to Unequal Learning, Unequal Number and Different Classification Costs 115

6. Training and Testing Results **120**

6.1 Computational Expense of Different Classifiers 120

6.2 Iris Data 123

6.2.1 Iris Data with Equal Number of Training Data Problem 123

6.2.2 Iris Data with Two Unequal Training Data 127

6.3 Vehicle Acoustic Data 130

6.3.1 Five-Class Problem 130

6.3.2 Four-Class Problem 139

6.3.3 Small versus Large Vehicle Classification Problem 144

6.4 Dimension Reduction Study 149

6.5 Conclusions 154

7. Summary and Future Research Directions **155**

7.1 The Main Contributions of the Dissertation 158

7.2 List of Future Research Directions 158

Bibliography **161**

Vita **172**

List of Figures

Figure 1.1 Typical pattern recognition system 3

Figure 1.2 Block Average method used in feature extraction process 6

Figure 1.3 A pattern classification system using discrimant functions 9

Figure 2.1 Vehicle acoustic classification systems 15

Figure 2.2 Several sub-tasks in vehicle acoustic classification study 16

Figure 2.3 Magnitude response of tractor trailer truck's acoustic signals 18

Figure 2.4 Geometry of N element circular of the r^{th} ring 20

Figure 2.5 Testing fitness function 31

Figure 2.6 Average of the best individual found of 100 runs 34

Figure 2.7 Average of the best individual found of 100 runs 35

Figure 2.8 Circular arrays of microphones 38

Figure 2.9 Array gains (Across-the-Lane) corresponding to frequency at 9000 Hz 38

Figure 2.10 Array gains (Across-the-Lane) corresponding to frequency at 5000 Hz 40

Figure 2.11 Angle errors versus maximum sidelobe levels (dB) 41

Figure 2.12 Angle errors versus 30 dB beamwidth 41

Figure 2.13 Radius errors versus maximum sidelobe Levels (dB) 42

Figure 2.14 Radius errors versus 30 dB beamwidth 42

Figure 2.15 Gain errors versus maximum sidelobe levels (dB) 43

Figure 2.16 Gain errors versus 30 dB beamwidth (dB) 43

Figure 2.17 Overall array system 44

Figure 2.18 (a) Microphone part, (b) 1st order highpass filter 45

Figure 2.19 Energy for each windows for all 152 microphones 47

Figure 2.20 Histogram of gains for all 153 Microphones 48

Figure 2.21 Example of each branch circuits testing 49

Figure 2.22 Energy for each windows for all branches of all summers 49

Figure 2.23 Histogram of gains for all branches of all summers 50

Figure 2.24 Energy for each windows for all branches and microphones of all summers 51
Figure 2.25 Histogram of gains for all branches and microphones of all summers 51
Figure 2.26 Array gains (Across-the-Lane) corresponding to frequency at 9000 Hz. 53
Figure 2.27 Array gains (Across-the-Lane) corresponding to frequency at 9000 Hz. 54
Figure 2.28 Magnitude response of testing signals 54
Figure 2.29 Across-the-Lane direction 55
Figure 2.30 Along-the-Lane direction 55
Figure 2.31 Roadside data collection diagram 57
Figure 2.32 Playback process 57
Figure 3.1 Passenger car information (a) the original signals sampled with 44.1 kHz, (b) energy (*E*), (c) filtered signals, (d) average energy (*avE*) 60
Figure 3.2 Two-axle truck information (a) the original signals sampled with 44.1 kHz, (b) energy (*E*), (c) filtered signals, (d) average energy (*avE*) 60
Figure 3.3 Energy patterns of the same vehicle passing the sensor four different times in one hour span (2700 - 5400 Hz) 61
Figure 3.4 Energy patterns of the same vehicle passing the sensor four different times (4500 to 5400 Hz) 62
Figure 3.5 Detection and feature extraction diagram 64
Figure 3.6 Fuzzy membership functions of number of windows from starting point to the peak (Nwd) 67
Figure 3.7 Fuzzy membership functions of the peak value (P) 67
Figure 3.8 Fuzzy membership functions of Nwd-after 68
Figure 3.9 Feature extraction regions 70
Figure 3.10 Feature extraction regions of tractor trailer truck (Class 9, FHWA) 72
Figure 4.1 Artificial neuron (McCulloch-Pitts neuron model) 81
Figure 4.2 Examples of activation functions 83
Figure 4.3 Multilayer perceptrons (MLP) network 84

Figure 4.4 Fuzzy logic systems with fuzzifier and defuzzifier 87
Figure 4.5 Fuzzy inference engine 89
Figure 4.6 Minimum inference 90
Figure 4.7 Product Inference 90
Figure 4.8 The overall output fuzzy set obtained by using Max-Min and Max-Product inference 91
Figure 4.9 The approximated output of different defuzzifiers 93
Figure 4.10 MIMO-AFLS with center average defuzzification 95
Figure 4.11 Symmetric membership function 100
Figure 4.12 Triangular shape membership function 100
Figure 4.13 Consequent fuzzy sets placed on massless beam 101
Figure 4.14 AFLS with balance of area (BOA) defuzzification method 103
Figure 4.15 Results of different defuzzifiers 104
Figure 4.16 Example of Gaussian shape consequent membership functions and different 105
Figure 4.17 Example of triangular shape consequent membership functions and different defuzzification methods 105
Figure 6.1 Average sum-squared-error of each class WITHOUT learning factor of MLP 131
Figure 6.2 Average sum-squared-error of each class WITH learning factor of MLP 131
Figure 6.3 Average sum-squared-error of each class WITHOUT learning factor of AFLS-BOA 132
Figure 6.4 Average sum-squared-error of each class WITH learning factor of AFLS-BOA 132
Figure 6.5 Average sum-squared-error of each class WITHOUT learning factor of AFLS-CA 133
Figure 6.6 Average sum-squared-error of each class WITH learning factor of AFLS-CA 133

Figure 6.7 Percentage of total and average correct of the BEST performance in five-class problem 137
Figure 6.8 Percentage of total and average correct of the MEDIAN performance in five-class problem 138
Figure 6.9 Computational expense during training (flops) in vehicle 5-class problem 139
Figure 6.10 Percentage of total and average correct of the BEST performance in four-class problem 141
Figure 6.11 Percentage improvement of BOA over other classifiers for BEST performance in four-class problem 142
Figure 6.12 Percentage of total and average correct of the MEDIAN performance in four-class problem 143
Figure 6.13 Percentage improvement of BOA over other classifiers for MEDIAN performance in four-class problem 144
Figure 6.14 Percentage of total and average correct of the BEST performance in small vs. Large vehicle problem 146
Figure 6.15 Percentage improvement of BOA over other classifiers for BEST performance in small vs. Large vehicle problem 146
Figure 6.17 Percentage improvement of BOA over other classifiers for MEDIAN performance in small vs. Large vehicle problem 148
Figure 6.18 Example of some eigenvalues of correlation matrix 153
Figure 7.1 Fusion of classifiers 159
Figure 7.2 Sub-task classifiers 159
Figure 7.3 Classifier and predictor systems 160

List of Tables

Table 1.1 FHWA recommended vehicle classification scheme 2

Table 2.1 Crossover-point operation (the crossover-point is 7 in this case) 25

Table 2.2 Example of mutation process 27

Table 2.3 Absolute difference between the *best* solutions in 100 runs and ideal solutions of each algorithm 33

Table 2.4 Absolute difference between the *worst* solutions in 100 runs and ideal solutions of each algorithm 33

Table 2.5 Absolute difference between the *average* solutions of 100 runs and ideal solutions of each algorithm 34

Table 2.6 Results of GAs (dverage), MGAs (dverage) and MGAs (best) for circular arrays design problem 39

Table 3.1 Fuzzy rule base for number of windows after the peak approximation 66

Table 3.2 Feature descriptions 71

Table 4.1 Numerical results of different defuzzifiers 104

Table 4.2 Results of Gaussian shape membership functions 106

Table 4.3 Results of triangular shape membership functions 106

Table 5.1 Example of classification results confusion matrix 114

Table 5.2 Example of classification cost confusion matrix 114

Table 6.1 Computational expense during single-epoch of training of each classifier 121

Table 6.2 Computational expense of trained classifier 122

Table 6.3 Classification costs in Iris data with the same number of training data 124

Table 6.4 Hidden nodes determination of MLP in Iris data with equal training data 125

Table 6.5 Number of rules determination of BOA in Iris data with equal training data 125

Table 6.6 Determination of *k* of *k*-NN in Iris data with equal training data 126

Table 6.7 Comparison of the BEST performance of each classifier in Iris data with equal training data 126

Table 6.8 Comparison of the MEDIAN performance of each classifier in Iris data with equal training data 127
Table 6.9 Classification costs used in Iris two-class problem 128
Table 6.10 Hidden nodes determination of MLP in Iris data with unequal training data 128
Table 6.11 Number of rules determination of BOA in Iris data with unequal training data 128
Table 6.12 Determination of *k* of *k*-NN in Iris data with unequal training data 128
Table 6.13 Comparison of the BEST performance of each classifier in Iris data with unequal training data 129
Table 6.14 Comparison of the MEDIAN performance of each classifier in Iris data with unequal training data 129
Table 6.15 Classification costs in five-class problem 134
Table 6.16 Hidden nodes determination of MLP in five-class problem 135
Table 6.17 Number of rules determination of BOA in five-class problem 135
Table 6.18 Determination of *k* of *k*-NN in five-class problem 136
Table 6.19 Comparison of the BEST performance of each classifier in five-class problem 136
Table 6.20 Percentage difference of the BEST performance from BOA in five-class problem 137
Table 6.21 Comparison of the MEDIAN performance of each classifier in five-class problem 138
Table 6.22 Percentage difference of the MEDIAN performance from BOA in five-class problem 139
Table 6.23 Classification costs used in four-class problem 140
Table 6.24 Comparison of the BEST performance of each classifier in four-class problem 140
Table 6.25 Percentage difference of the BEST performance from BOA in four-class

problem 141
Table 6.26 Comparison of the MEDIAN performance of each classifier in four-class problem 142
Table 6.27 Percentage difference of the MEDIAN performance from BOA in four-class problem 143
Table 6.28 Classification costs in a small versus large vehicle problem 144
Table 6.29 Comparison of the BEST performance of each classifier in a small versus large vehicle problem 145
Table 6.30 Percentage difference of the BEST performance from BOA in a small versus large vehicle problem 145
Table 6.31 Comparison of the MEDIAN performance of each classifier in a small versus large vehicle problem 147
Table 6.32 Percentage difference of the MEDIAN performance from BOA in a small versus large vehicle problem 147
Table 6.33 Comparison of the BEST performance of each classifier in five-class problem of dimension reduction study 149
Table 6.34 Percentage difference of the BEST performance from BOA (30 features) in five-class problem of dimension reduction study 150
Table 6.35 Comparison of the MEDIAN performance of each classifier in five-class problem of dimension reduction study 150
Table 6.36 Percentage difference of the MEDIAN performance from BOA (30 features) in five-class problem of dimension reduction study 151
Table 6.37 Comparison of the BEST performance of each classifier in four-class problem of dimension reduction study 151
Table 6.38 Percentage difference of the BEST performance from BOA (30 features) in four-class problem of dimension reduction study 152
Table 6.39 Comparison of the MEDIAN performance of each classifier in four-class problem of dimension reduction study 152

Table 6.40 Percentage difference of the MEDIAN performance from BOA (30 features) in four-class problem of dimension reduction study 153

Chapter 1

Introduction

Classification problems involve assigning unknown patterns into one of a pre-specified number of classes based on the extraction of significant features or attributes. Such a simple problem to a human is not so simple when we want to make a machine perform this task. In order to be able to classify its input, the machine has to be able to process the input information, measure its similarity and decide which class that input belongs to based upon its recognition of the similarities and differences among classes, i.e., the patterns that belong to each class. We may say that a pattern classification problem is a pattern recognition problem and that recognition is the ability to classify.

1.1 Background and Motivation

The Federal Highway Administration (FHWA) recommends thirteen major vehicle classes as shown in Table 1.1 [101]. This vehicle classification serves as a guideline throughout the country. Some vehicle classes are clearly distinct from other classes, but some are not. The most fuzzy class is class number three in which all two-axle vehicles beside vehicles in class number two are grouped into one class. This class may include pickups, vans, mini-vans, etc. There are obviously at least two subclasses in this class: pickups and vans. The most important information used in this vehicle classification scheme is axle information. Both number of axles and axle spacing are used thoroughly. It is because the most important use of vehicle classification is in the predictions of damage to highways caused by heavy vehicles [102]. A vehicle having more axles and, thus likely more weight, tends to damage the road surface more than a vehicle having fewer axles. This information can be used by Department of Transportation (DOT) Pavement Management System to forecast pavement maintenance needs. Besides road surface maintenance, this vehicle classification information is also used to estimate in cost responsibility studies to

allocate the costs of building and maintaining the highway systems across classes of vehicles in proportion to their responsibility for these costs, e.g. developing tax rates for motor vehicles. Other uses of vehicle classification data are to compare the accident rates of different classes of vehicles and to analyze road capacity, determining the width and number of lanes required and the need for additional climbing lanes, and designing intersections and interchanges.

Table 1.1 FHWA recommended vehicle classification scheme

Vehicle Class	Vehicle Description
1	Motorcycles
2	Passenger cars
3	Other two-axle, four-tire single unit vehicles (Pickups/Vans)
4	Buses
5	Two-Axle, six-tire single unit trucks
6	Three-Axle single unit trucks
7	Four-Axle single unit trucks
8	Three or Four -Axle single trailer trucks
9	Five-Axle single trailer trucks
10	Six or Seven-Axle single trailer trucks
11	Five-Axle multi-trailer trucks
12	Six-Axle multi-trailer trucks
13	Seven -Axle multi-trailer trucks

Pattern recognition is an inexact science involving many areas of disciplines, admitting many approaches, sometimes complementary, sometimes competing, to the approximate solution of a given problem [12]. A typical pattern recognition system may consist of various parts as shown in Figure 1.1.

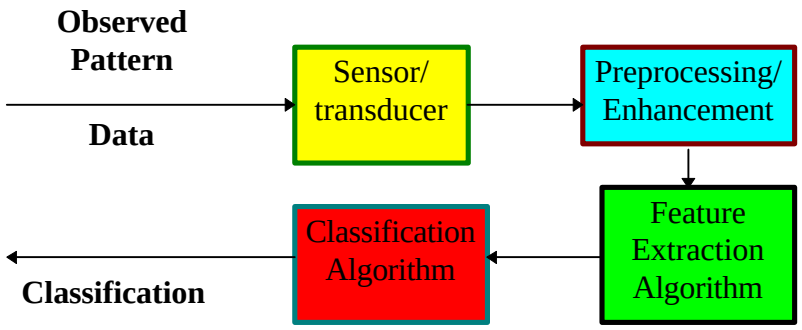

Figure 1.1 Typical pattern recognition system

The observed pattern data are generated from each class. In this case, the observed pattern data are vehicle acoustic signals and the sensor is a microphone or a group of microphones. The sensor will convert the observed world pattern data into electric signals. These electric signals, then, are filtered, sampled as part of preprocessing/enhancement. The dimension of preprocessed data may be too big and redundant to be used in detection and/or classification. For example, there are 44100 points of 1 second of acoustic signals sampled at a 44.1 KHz sampling rate. It is certainly impractical to feed all this data to the classifier. Information reduction is necessary in most cases. This process is called feature extraction. Only significant features are extracted from the processed data. The information about the observed pattern data that the classifier will have are the features extracted. This process is the first and important step to the success or failure of the classification system. If the extracted features are excellent, e.g., all classes are linearly separated, a simple classifier will suffice. If the features extracted are poor, a difficult classification task will be imposed on the classifier design. Thus, all parts of a classification system are important; all parts can affect the performance of the system. Therefore, in the design process, the final performance will be fed back to adjust each part of the system if necessary.

Situations in this world generally cannot be assessed in terms of isolated facts or even in terms of a body of isolated facts [81]. Rather, we find that we need to describe situations in terms of patterns of interrelated facts. Humans are seemingly well adapted to such pattern processing

tasks. For example, we can recognize speech utterances and images such as handwriting in a robust manner even though there are major variations, distortions, or omissions. We can recognize a friend even if we only partly see him in a crowd of people. It is quite amazing to think about human capabilities in pattern recognition problems. Motivation for the study of pattern recognition is our desire to understand the basis for these powers in humans and to build computer-based machines having comparable capabilities. Developing machine classifiers is part of a larger concern with the ideas of designing and making intelligent machines that can carry tasks with skills comparable to human performance. In other words, we would like to have computer based machines with the same pattern information processing capabilities, perception and cognition, that we ourselves possess, ultimately, e.g. for people to communicate with computing machines in natural languages. The goal of this study is to build adaptive classifiers, which use a training algorithm to adapt its parameters in order to solve the vehicle classification problem using acoustic signals. This research will utilize adaptive fuzzy logic systems and neural networks as classifiers. All these matters constitute the motivation for this research.

1.2 Survey of Previous Works

1.2.1 Acoustic Pattern Classification

So far, we have discussed the general problem of classification, and the classification methods presented in this study may be applicable to fairly general classification problems. However, in this study, the concentration will be on classification of acoustic signals, especially vehicle acoustic signals. The overall acoustic signal of a vehicle arises from several sources which include the engine, gears, fan, cooling system, road-tire interaction, exhaust, brakes, and aerodynamic effects [22]. What we want to classify in some cases is not only acoustic signals, but also the source that generates the acoustic signals, such as vehicle classification and speaker verification. In that sense, we try to identify the source of the acoustic signal by acoustic classification. In some cases, we want to classify the acoustic signal itself such as in phoneme

classification. If we want to identify the source of acoustic signals, we might ask why acoustic signals instead of other signals. For example, in a vehicle classification problem, there are many other ways to classify a vehicle such as classification by length, wheel spacing or image. Many transportation agencies use inductive loops which are the most popular vehicle detection devices. This device detects the presence of a vehicle from the magnetic field change caused by that vehicle. By using two of these inductive loops, they can estimate speed, axle spacing and then classify the type of vehicle based upon this information. This device works with acceptable accuracy, thus, most agencies use this device in practice. One drawback is that this device has to be put under the pavement, making it hard to install this device [94]. Care is required to cut a precise depth to achieve a good performance. If there is a problem with this device, the road must be torn up and a lane would have to be closed for the operation. Thus, it will affect traffic flow, especially on an interstate highway. With the same idea, one might use a piezo electric device to detect the presence of vehicle. When a vehicle runs over this device, it will generate a signal that can be used to detect the presence of that vehicle. Once again, by using two devices, they can estimate axle spacing and classify a vehicle based upon this axle spacing and the strength of the generated signals. This device still has the problem of being embedded in the road the same as inductive loops. Another drawback of these two devices is their sensitivity to weather conditions. The performance of both devices will degrade with changes of temperature. An expensive solution is to use a video image device for classification, but during poor weather and at night time, even this device cannot be used effectively. With these problems and corresponding costs, most transportation agencies are trying to find an alternate way of monitoring traffic.

One alternative is an acoustic sensor such as the IRD SmartSonic sensor to detect acoustic signals emitted from a passing vehicle [97]. This kind of sensor operates in all-weather conditions, i.e., is relatively unaffected by rain, snow, fog, and temperature changes. It is a passive acoustic sensor, that is, it emits no radiated energy, and requires low maintenance and no road surface cutting.

A similar approach to , i.e. vehicle classification using acoustic signals, was done at the Center for Transportation Research, Virginia Polytechnic Institute and State University, for AT&T SmartSonic development [45][46]. The AT&T SmartSonic sensor installed on highway I-460 was used to collected data. The processed signals were cross-correlated signals between signals detected across the lane and signals detected along the lane. The signals were sampled and processed and reduced to a frequency of 100 Hz. Features extracted from the processed signals were time domain features only. The method of block-average, an average of processed signals in a small window, was used to compress the data from the raw data as shown in Figure 1.2.

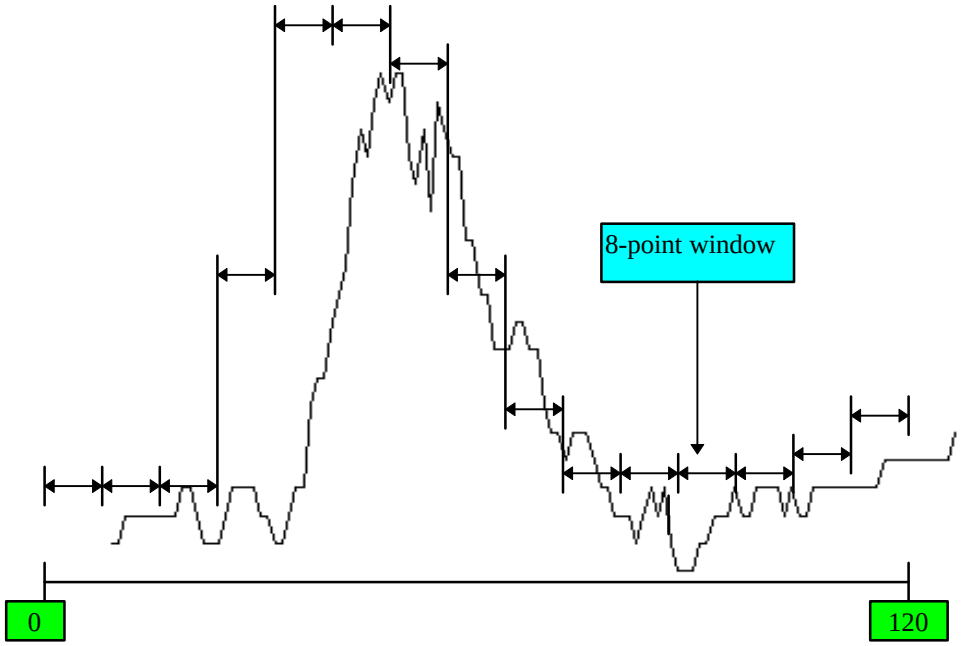

Figure 1.2 Block average method used in feature extraction process

There are four classes of studied vehicles: passenger cars, small trucks, heavy-duty trucks and tractor trailers. The studied data consisted of 800 data with 200 data for each class. For a 2-class problem, cars and trucks, the data, except the passenger car, are combined to be a "truck" class. There were several classifiers designed in the project such as a classical K-nearest neighbors, multilayer perceptrons, radial basis function networks and two-stage multilayer perceptron networks. The comparison of the performance of all classifiers was made by using a

cross validation method so that the training and testing data were different. The training data were 80% of each class and testing data, the rest of data. The overall performance was a cumulative performance of each training and testing data pair. Classification performance between cars and trucks was very good by using MLP. The best performance was about 96% correct classification. For the four-class problem, the best performance was about 90% correct classification. The two-stage MLP network was reported as the best classifier among these methods. Two-stage MLP network consisted of two sets of MLP networks. In the first set, a MLP network was designed for each pair of classes. The numbers of networks were equal to the numbers of different possible pairs. For example, suppose there are four classes, there are six different pairs. Each network was trained by only data in its corresponding classes. The training of the two-stage MLP network was done separately. The first stage networks were selected from the best network according to the cross validation method. Then, all weights of the first stage networks were fixed during training phase of the second stage network. The second stage network that was the overall network was trained by using the outputs of the first stage networks and the corresponding desired outputs. In this way, the training and testing data of the second stage network may be different from the first stage network data set. The testing data of the second stage network, may be the training data of some first stage networks. Thus, the results are biased in some degrees in the two-stage MLP network. All these results were done based upon only the studied data, not all available data. These works should be expanded to include all available data. The cross validation method used predetermined orders of training and testing data sets. The total 800 data were divided into 5 equal smaller sets. The first training data was selected from the first four sets and the testing data was the fifth set. The next training data was selected from the first, second, third, and fifth sets and the testing data was the fourth set, etc. In this way, the results represented only on particular sets of data. The training and testing data should be selected randomly from all available data. That would introduce unequal numbers of available data from different classes, that leaded to a study of this dissertation research. The data of each class was very restricted. For example, in small truck class, only small trucks with box were in that class and other small trucks having two axle six tire were excluded. This work

should be expanded to include all types of vehicles having the same FHWA recommended descriptions in one class.

Another similar work on pattern recognition which classifies aerospace acoustic sources is Cabell's work in his Master's thesis [14]. He designed a system to identify five different acoustic sources, jet planes, propeller planes, a helicopter, wind turbines and trains. The acoustic signal is detected by a single microphone and recorded on a cassette tape. The training data for helicopter, wind turbine and train is very small compared to the remaining classes, i.e. only one helicopter in the entire data set. He used a linear perceptron as a classifier. The performance of the classifier fluctuated due to insufficient training or testing data in his study. Scott extended Cabell's work by introducing associative memory and multilayer perceptrons as classifiers [93]. The performance is better than Cabell's work in most cases. The problem of the number of training and testing data was not solved. Although the helicopter data was processed at different times, it was still the same data set for both training and testing data. In both works, a large set of features (108 features), both in time domain and frequency domain, was selected. Then, they used a feature selector algorithm to eliminate unimportant features. The final features are on the order of 7 to 10.

The most active research on classification of acoustic signals using artificial neural networks and fuzzy logic systems is in the speech recognition field [4][6][10][13][37][65][72][120]. Most neural networks and fuzzy logic systems applications in the speech recognition field are subtasks such as phoneme classification, voice-unvoice-silence discrimination, etc. Some works combine a neural network with a conventional approach to improve speech recognition performance [24][29][43][72][96]. In other words, a neural network is used as another tool in speech recognition field. As in other fields, the neural network research and development for speech recognition is relatively new and lags far behind that of conventional methods. Recurrent neural networks combining with system identification techniques are promising methods in speech recognition applications.

1.2.2 Conventional and Artificial Neural Networks Classifiers

A Bayesian classifier is the most famous fundamental statistical approach to the problem of pattern classification [25]. This approach is based on the assumption that the decision problem is in probabilistic terms and all of the relevant probability values are known. A pattern classifier can be viewed as a machine that computes discriminant functions and selects the category corresponding to the largest discriminant as shown in Figure 1.3.

In Bayesian classifier, all relevant probabilities are defined as :

$$g_i(\mathbf{x}) = P(\omega_i|\mathbf{x}) \tag{1.1}$$

$$P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})} \tag{1.2}$$

and

$$p(\mathbf{x}) = \sum_{i=1}^{c} p(\mathbf{x}|\omega_i)P(\omega_i) \tag{1.3}$$

where $g_i(\mathbf{x})$ is a discriminant function calculated by the classifier, $P(\omega_i|\mathbf{x})$ is a posterior probability defined in (1.2), $p(\mathbf{x}|\omega_i)$ is the conditional probability density function depending on the state of nature, $P(\omega_i)$ is a priori probability of the state of nature ω_i, and $\mathbf{x}$ is a feature vector.

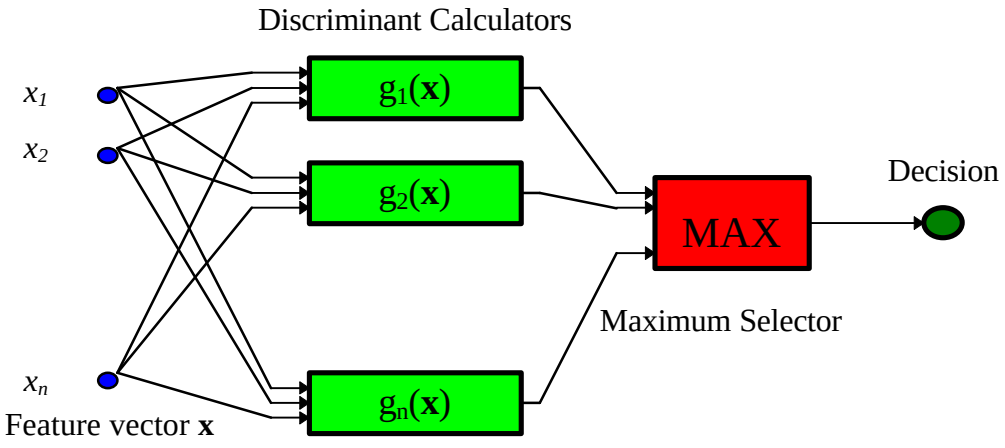

Figure 1.3 A pattern classification using discrimant functions

The Bayesian classifier is considered to be the optimal classifier in a sense that if all relevant probabilities are known, the maximum correct classification can be obtained by the Bayesian classifier. No other classifiers can achieve higher correct classification than the true Bayesian classifier. Unfortunately, in real world pattern classification applications we rarely have complete knowledge about the probabilistic structures of the problem. In other words, there is no true Bayesian classifier in practice. All relevant probabilities have to be approximated. There are two approaches: parametric and nonparametric estimation. The parametric estimation approach is to use the samples to estimate the unknown parameters of the assumed forms for the unknown probabilities and probability densities, and to use the resulting estimates as the true values in a Bayesian classifier. Nonparametric techniques, such as the k-nearest-neighbor rule, bypasses probability estimation and goes directly to decision functions. The k-nearest-neighbor rule classifies **x** by assigning it the class most frequently represented among the *k* nearest samples. In other words, a decision is made by examining the classes on the *k* nearest neighbors and taking a vote. It was shown in [25] that as *k* goes to infinity, the k-nearest-neighbor rule becomes equivalent to the Bayesian classifier.

Genetic Algorithms (GAs) are inspired by Darwinian theories and are considered to be effective optimization tools [9][21][19][91]. John Holland invented GAs in the early 1970's to mimic some of the processes observed in natural evolution. Its main features are, an encoding mechanism, evaluation process, parent selection and reproduction mechanisms. GAs display complicated behavior, and can solve some extremely difficult problems. To solve real world applications encoded in real numbers, there must be at least two conversions: from binary to integer and from integer to real number or vice versa. To work directly with real numbers, some researchers exclude the crossover operation from the algorithm. Only mutation with addition or subtraction by a small random number is carried out. This method is called an evolution algorithm, or sometimes a genetic hill climber algorithm. The best chromosome is kept as a prototype parent, and the mutation process is applied to the parent to get a new population.

Then, all chromosomes in the new generation are evaluated and the best overall chromosome is kept for the next generation. This evolution algorithm may not be as global search as the traditional genetic algorithm. Some researchers have kept the crossover operation that is modified such that it can work with real numbers. For example, Davis used the average crossover operator by taking two parents and producing one child that is the result of averaging the corresponding parents [21]. The evolution algorithms do not posses good features of standard GAs such as the crossover operation.

Neural networks are biologically inspired. Their basic units, connections and overall performances work in ways similar to how we think neurons in the human brain work [36][38][39][40][48][49]. The most popular network is the multilayer perceptron (MLP) trained by the error-backpropagation algorithm. MLPs have been proved to be universal function approximators [38]. Because of its generalization, approximation and learning capability, MLPs can be trained to do pattern classification from given examples. The examples are data pairs of the pattern vectors and their corresponding class. The pattern vectors are usually preprocessed and attributes extracted from the original patterns. The corresponding class is normally represented in a binary form, e.g. $[1\ 0\ 0\ 0]^T$ for class 1, $[0\ 0\ 1\ 0]^T$ for class 3 in a 4 class problem. In this way, the number of outputs of the network is equal to the number of classes. In the training process, weights connecting neurons are updated such that the sum-squared-error between the desired and the actual outputs of the network is minimized. The information about the classification problem is distributed through its weights and their connections. Once it is successfully trained, it can give an estimated class to a previously unseen pattern vector. These approximated outputs are used in decision process to classify that pattern. Normally, the pattern will be assigned to a class having the maximum output. In this way, the MLP can be viewed as a discriminant calculator.

A fuzzy logic system (FLS) is a combination of linguistic variables and a set of IF-THEN rules using fuzzy logic principles [51][58][67][105][106][107]. It is a system that utilizes fuzzy

set theory and its operations. The most commonly used fuzzy logic system consists of a fuzzifier, inference engine, fuzzy rule base and defuzzifier. The fuzzifier converts real world data into fuzzy numbers, membership values. An inference engine uses fuzzy logic rules in a fuzzy rule base in mapping from fuzzy sets to other fuzzy sets. The inputs of the inference engine are the membership values that are the outputs of the fuzzifier. The outputs of the inference engine are fuzzy sets. The defuzzifier converts fuzzy sets back into real numbers. In this way, the fuzzy logic system can be viewed as function mapping from real numbers to real numbers for use in real world problems. FLSs are widely used in control applications, but rarely used in pattern classification [67][68].

An adaptive fuzzy logic system (AFLS) can be defined as the fuzzy logic system whose rules are extracted from numerical data through training, i.e. a FLS equipped with training algorithms so that all its parameters, e.g. centers, spreads, can be adapted in the same manner as with neural networks [105][106][107]. AFLSs have been proved to be universal function approximators [59][106]. The training algorithm adjusts the parameters of the fuzzy logic system based on numerical information. Unlike MLP weights, its parameters have clear meanings in the system. Thus, all its parameters can be effectively initialized and trained from numerical data [105]. Most fuzzy logic systems are normally analyzed and designed as multi-input single-output systems. In the multi-output problems, there is a FLS for each output. In other words, a whole system is decomposed into small systems.

Fuzzy logic techniques can be used in several stages in pattern classification problems. There are excellent papers involving cluster analysis, feature analysis and classifier design in [12]. Fuzzy techniques serve as another supporting tool in solving pattern classification problems.

The fuzzy logic systems have problems with defuzzification part [67][105]. There are several methods: centroid of area (COA), center average (CA), modified center average (MCA), etc. The most popular method is the COA method. The COA uses entire shapes of the

consequent parts of the IF-THEN rules to compute the output. The problem of COA is that it requires many computations. The CA method requires less computations but it suffers from not using the entire shape of the consequent membership function. Regardless of whether the max-min or max-product inference is used, the result of the center average defuzzifier is still the same, regardless of whether the shape is narrow or wide. The MCA method includes the spread information to compute the output but its output is further away from the COA method.

1.3 Objectives and Scope of Research

The objective of this research is to develop new techniques of fuzzy neural classifiers, to investigate and extract significant features from acoustic signals and ultimately to design a robust, effective, and reliable vehicle classification system. The system performance goal is to detect and classify major vehicle classes from acoustic signals. The following main issues considered in this research are:

- Design a traffic acoustic sensor system.
- How to design and build acoustic sensors.
- How to detect and extract significant features from vehicle acoustic signals.
- How to design and train neural fuzzy classifiers.
- How to train neural fuzzy classifiers by using unequal available training data from different classes.

1.3.1 Scope of the research

The scope of this research is described by the following tasks:

- To design and build an acoustic sensor.
- To design a detection algorithm used in an acoustic sensor system.
- To analyze and extract good features from detected acoustic signals.
- To design neural fuzzy classifiers.

- To develop a new defuzzification technique for fuzzy logic systems.
- To develop new techniques to train proposed neural fuzzy classifiers to solve unequal numbers of training data of each class.

1.4 Contributions of dissertation research

The following contributions are resulted from the research work:

- A prototype acoustic sensor system with high correct classification rate is proposed with 97.95% correct classification rate between small and large vehicles on 1327 vehicles, 92.24% correct classification rate in four-class problem on 1327 vehicles, and 78.67% correct classification rate in five-class problem on 1327 vehicles.
- A new modified genetic algorithm is developed for use in acoustic arrays design. This modified genetic algorithm has all conventional genetic algorithm properties, but offers a more efficient and truly global search. It can work directly with real numbers.
- An acoustic circular array of microphones is designed and built to be used as a traffic sensor.
- A detection and feature extraction algorithm is developed to be used with acoustic sensor system.
- MIMO adaptive fuzzy logic systems are developed for use as classifiers in this study. They can be used, not only in classification problems, but also in other applications such as system identification applications. These systems can be trained by error backpropagation algorithms.
- A new defuzzification method is developed. This method is named balance of area (BOA) defuzzification. This defuzzification method can be used with standard fuzzy logic system with fuzzifier and defuzzifier or can be used with MIMO AFLS.
- An unequal learning method is developed to be used in training a classifier for classification problems with unequal numbers of training data from different classes.

Chapter 2

Acoustic Sensor Design and Data Acquisition

An acoustic classification system design composes of acoustic sensor design and data acquisition, feature analysis and extraction, classifier design, and implementation. The design of acoustic sensor and data acquisition procedures are presented in this chapter. Feature analysis and extraction and classifier design will be presented in the next chapters. The implementation part will not be included in this research. This research will only attempt to demonstrate the feasibility of such a system. An acoustic circular sensor is designed and built as a prototype and is used to collect data in this study.

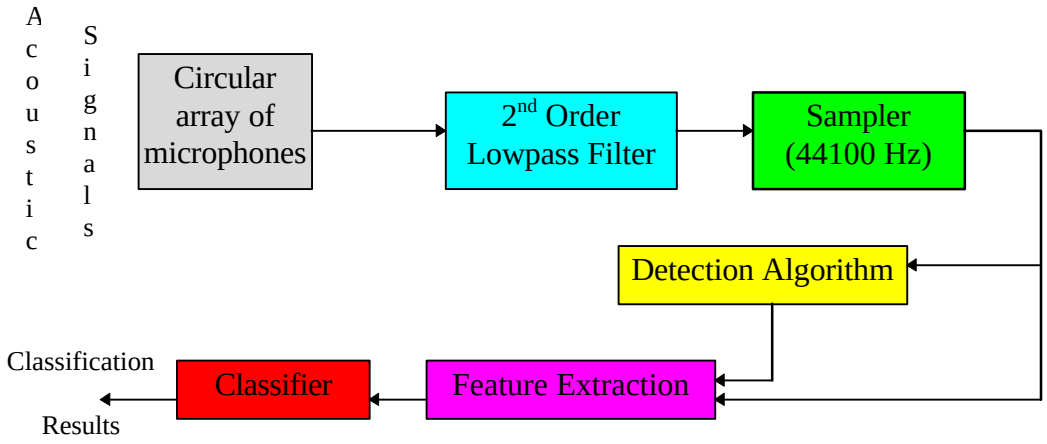

Figure 2.1 Vehicle acoustic classification systems

The vehicle acoustic classification systems may consist of several sub-systems as shown in Figure 2.1. In this dissertation research there are several sub-tasks as shown in Figure 2.2. In task #1 the acoustic sensor is designed, built and used to collect data. In task #2 the recorded data is processed and sampled by a sound card with a sampling rate of 44100 Hz and put into a file to be analyzed or processed at a later stage. These two tasks are presented in this Chapter. Task #3 will be presented in Chapter 3. Task #4 will be presented in Chapters 4 and 5.

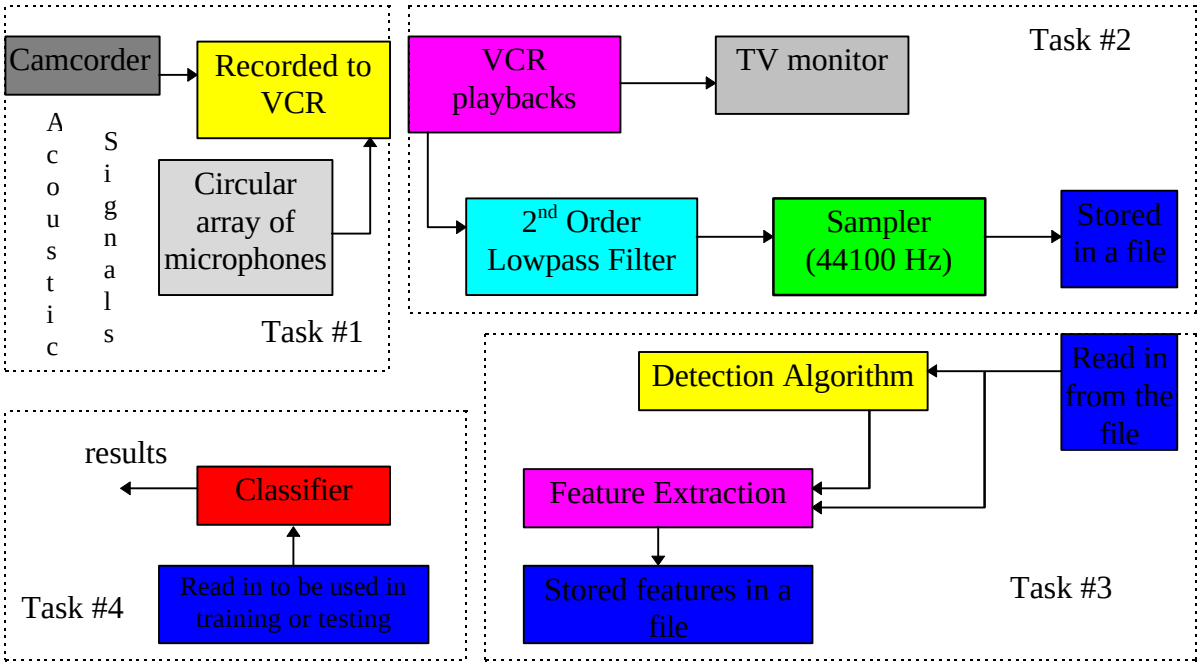

Figure 2.2 Several sub-tasks in vehicle acoustic classification study

2.1 Acoustic Sensor Design

Acoustic signals emitted by a vehicle are to be detected and recorded. Ideally, the detected acoustic signal from a single vehicle is desired one at a time. However, there is often interference from other vehicles in adjacent lanes and/or in the same lane under a normal traffic situation. This poses the interesting problem of how to exclude undesired signals without interrupting traffic flows. All of these signals are passive and uncooperative signals. In other words, there is no communication between the sensor and a vehicle in this case. Thus, the primary concern in the acoustic sensor design is noise suppression. Unlike other information such as 2-D images, detected acoustic signals are more than likely to be contaminated by acoustic signals from other vehicles. A difference of loudness between a heavy truck and an average automobile is about 25 to 30 dB [17]. Thus, signals from adjacent lanes should be suppressed at least 30 dB to reduce false alarms and have a high correct classification rate.

Acoustic signals emitted from a vehicle can be received by a microphone or microphones. An omnidirectional microphone can obviously not be used to meet all requirements in this case. A microphone gun can be used to detect acoustic signals directionally. One drawback of using this kind of microphone is that its main beamwidth is too large in this case [17]. A parabolic reflector antenna could also be used. If all specifications, e.g., 30 dB suppression, narrow beamwidth, must be met, the size of the reflector antenna might be so large that it becomes impractical. Arrays of microphones offer an alternative way to accomplish this task.

Most acoustic energies, as well as useful information, are in a low frequency range. To obtain such information, the size of the array is too big to implement. One constraint is the sensor size. Its size should be small. A big sensor will cost more not only to manufacture but also in operation. It is also hard to install or maintain. Using techniques of antenna design as well as an optimization algorithm, a low cost, but effective, acoustic sensor is to be designed. The new technique of genetic algorithms will make the art of this design even more interesting.

2.1.1 Circular Array of Microphones

The vehicle acoustic signals have a wide range in frequencies (see Figure 2.3). Most of the acoustic energy is in low frequencies. To get all information, a wideband antenna is a logical choice in this case. Constant beamwidth beamforming techniques have been developed and proved to be useful in audio applications [34][108].

However, constant beamwidth beamforming techniques require some modifications such as post-filtering or sampling the signals from each element. In the first case, there is too much cost to have a filter for each element. In the latter case, multi-beamforming technique is used. The multi-beamforming process involves many computations. It is impractical to have a traffic sensor with such high computation and cost. Therefore, the constant beamwidth design will not

be pursued in this research. Since we want to keep the cost low, only the simple beamforming principle of arrays will be used.

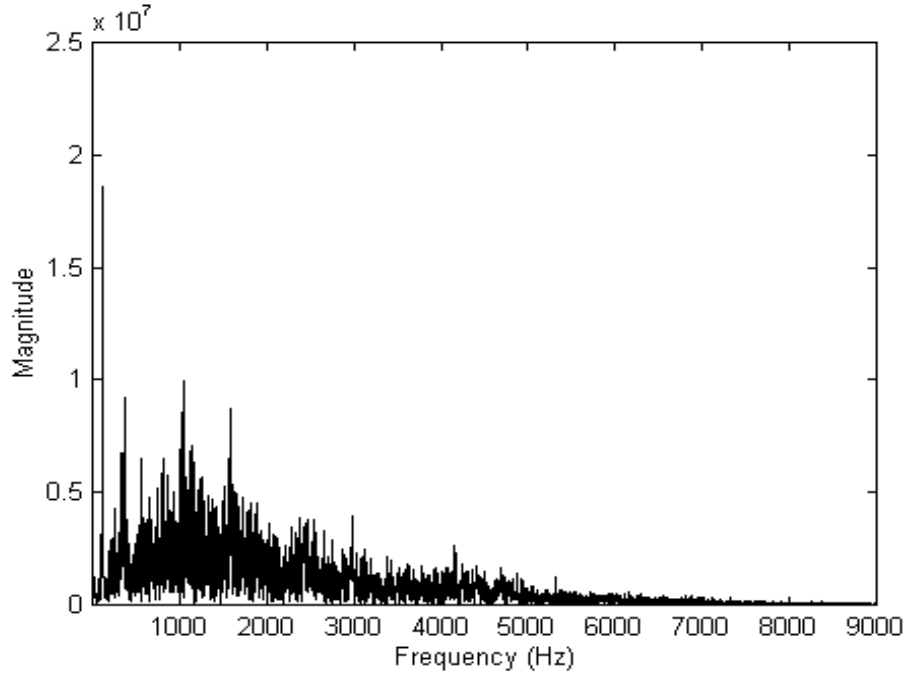

Figure 2.3 Magnitude response of tractor trailer truck's acoustic signals

The principle of uniform spacing planar arrays is well understood [5]. It is one of the most widely used antenna array configurations. This uniformly spaced planar array will be referred to simply as the "planar array." Design procedures of this kind of antenna are also available and well established. Since the planar array is a narrowband frequency array antenna, the design concentrates at a single frequency. Its beamforming pattern will deviate as the signals' frequencies deviate from the designed frequency. The main beamwidth will get narrower as the frequency gets higher; and, the peak sidelobe levels will get higher. As the frequency lowers, the main beamwidth will get wider and the peak sidelobe levels will be lower. Thus, an acceptable performance of this planar array will only be in a narrow band of frequencies. In this way, the planar array may not be the right choice for this application.

The circular array, in which the elements are placed in a circular ring, is an array configuration of very practical interest [5]. It is a non-uniformly spaced planar array. Multiple rings form an even more interesting configuration. Circular arrays with multiple rings are studied and used through out this research. An array factor is defined as a function of the number of elements, their geometrical arrangements, their relative magnitudes, their relative phases, and their spacing [5]. Each array has its own array factor. The array factor of a circular array is given in [5] as

$$AF(\theta,\phi) = \sum_{n=1}^{N} I_n e^{jkr_0\cos(\phi_n - \chi)} \tag{2.1}$$

where

I_n = amplitude excitation of the n^{th} element

$$k = \frac{2\pi}{\lambda} \tag{2.2}$$

λ = a wavelength at frequency of interest.

$$r_0 = r\left[(\sin\theta\cos\phi - \sin\theta_0\cos\phi_0)^2 + (\sin\theta\sin\phi - \sin\theta_0\sin\phi_0)^2\right]^{1/2} \tag{2.3}$$

θ = a vertical angle of the incoming signals (see Figure 2.4).

ϕ = a horizontal angle of the incoming signals.

θ_0 = a vertical angle of the peak of the main beam direction.

ϕ_0 = a horizontal angle of the peak of the main beam direction.

ϕ_n = an angular position of n^{th} element on *x*-*y* plane

$$\chi = \tan^{-1}\left[\frac{\sin\theta\sin\phi - \sin\theta_0\sin\phi_0}{\sin\theta\cos\phi - \sin\theta_0\cos\phi_0}\right] \tag{2.4}$$

The direction of the main beam is designed in direction of $\theta_0 = 0$. Thus, the array factor of a circular array with multiple rings can be simplified as

$$AF(\theta,\phi) = \sum_{r=1}^{R}\sum_{n=1}^{N_r} I_n^r e^{jkr_r\cos(\phi - \phi_n^r)} \qquad (2.5)$$

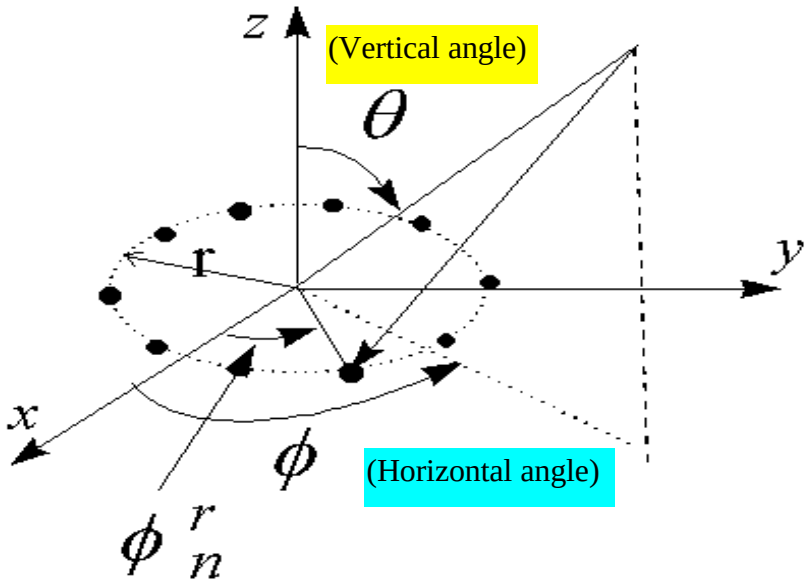

Figure 2.4 Geometry of N element circular of the r^{th} ring

where R = the number of rings

N_r = the number of elements in the r^{th} ring.

I_n^r is the excitation coefficient.

ϕ_n^r is an angular position of n^{th} element in the r^{th} ring on x-y plane.

$$\rho_r = r\sin\theta \qquad (2.6)$$

r = the radius of the r^{th} ring.

The circular array has an interesting and important property: constant minor lobes. Unlike the uniform planar array, the minor lobes of a circular array are constant as a function of frequency. In other words, the circular array has frequency-invariant minor sidelobe levels. The only change with frequency is its main lobe beamwidth. The beamwidth corresponding to the frequency change is in the same manner as the planar array. Its beamwidth corresponding to a

high frequency signal is narrower than a lower frequency one. To avoid grating lobes, the spacing between elements has to be less than or equal to one wavelength of the upper frequency of interest. The minimum spacing between elements is limited by the size of elements. The bigger the antenna is, the lower the frequency range having a satisfactory main beamwidth. The beamwidth of the lower frequency bound can be controlled by the radius of the outmost ring.

The spacing between rings of the circular array was designed heuristically to have relationship as

$$\tau = \frac{d_1}{d_2} = \frac{d_2}{d_3} = \frac{d_3}{d_4} = \ldots = \frac{d_{n-1}}{d_n} \tag{2.7}$$

where τ is the constant, d_1 is the spacing between the first ring that is the most inside ring and the second ring, d_2 is the spacing between the second ring and the third ring, and so on. In this way, the relationship of a radius of the m^{th} ring and a radius of the first ring can be derived by:

$$r_2 = r_1 + d_1 \tag{2.8}$$

$$r_3 = r_2 + d_2 = r_1 + d_1 + \frac{d_1}{\tau} = r_1 + d_1\left[\frac{1+\tau}{\tau}\right] \tag{2.9}$$

$$r_4 = r_3 + d_3 = r_1 + d_1\left[\frac{1+\tau}{\tau}\right] + \frac{d_1}{\tau^2} = r_1 + d_1\left[\frac{1+\tau+\tau^2}{\tau^2}\right] \tag{2.10}$$

In general form,

$$r_m = r_1 + d_1\left[\frac{\sum_{i=0}^{m-2}\tau^i}{\tau^{m-2}}\right] \qquad ; m=2, 3, \ldots \tag{2.11}$$

Since we want to control the size of the sensor, therefore the radius of the last ring is to be adjusted rather than the distance d_1. Within the characteristics of this circular array, the design becomes how to fine tune the parameters such as the radius of the inner most ring, the radius of the outer most ring, the constant ratio τ, the number of elements in each ring and the angle of the first element of each ring to the reference axis. Thus, this design problem becomes how to

optimize all these parameters to meet the requirements. Since genetic algorithms are an excellent optimization algorithm, they will be considered in this design.

2.1.2 Genetic Algorithms (GAs)

Genetic Algorithms (GAs) are inspired by Darwinian theories and are considered to be effective optimization tools [9][21][19][91]. GAs were invented by John Holland in the early 1970's to mimic some of the processes observed in natural evolution. Its main features are, an encoding mechanism, evaluation process, parent selection and reproduction mechanisms. Using simple encoding and reproduction mechanisms, GAs display complicated behavior, and can solve some extremely difficult problems.

The encoding mechanism encodes a real world problem into a bit string as a chromosome. For example if we work on a real number, that real number is encoded into a bit string composed of 0's and 1's. An entire solution is represented by one chromosome. The evolution process takes place at this chromosome level.

The evaluation is done in the real world environment. Before the evaluation process, the bit strings are decoded back to real numbers. The results of this evaluation process will be used in the evolution process. Thus, the evaluation process links between the evolution part and the real world problem.

The evolution part consists of the natural selection and reproduction process. Natural selection uses the results of the evaluation process to select parents for the next generation. The selection process is done in such a way that those chromosomes that encode successful structures are reproduced more often than those that do not. Once the natural selection process is done, the reproduction process takes place. The evolution actually takes place in this reproduction process. There are two main sub-processes in this reproduction process, recombination and mutation. The

recombination process combines the chromosomes of its parents. On the other hand, mutation may cause the children's chromosomes to be different from those of their parents.

The genetic algorithm may be summarized as follows:

a) Initialize a population of chromosomes. Do encoding as necessary.

b) Evaluation. Evaluate each chromosome in the population. Normally this evaluation is done from the decoded part. Do encoding as necessary.

c) Parent selection. Select parents according to their fitness. Those having more fitness have more chance to be selected.

d) Reproduction. Create new chromosomes by mating parents. Apply recombination and mutation of the parent chromosomes to create off spring.

e) Make room and insert the new chromosomes into the population.

f) If time is up, stop and return the best chromosome. If not, go to (b).

Genetic algorithms work well with bit strings, being both simple and effective. To link between real world problems and GAs, encoding and decoding of the real world problem is typically required which can be cumbersome and computationally expensive. For example, if we want to work with the real number space, all parameters must be converted into binary numbers. Normally, they must be decoded back to real numbers to evaluate their fitnesses. This introduces quantization problems. Genetic algorithms will search in a subspace of the operational real space. This means that their solutions are not truly a real optimum. It will be shown that genetic algorithms can be modified to work directly with real numbers.

Most features of GAs should be preserved as much as possible in order to have an effective algorithm. This approach can be viewed as an alternative representation of chromosomes. Chromosomes in GAs are binary numbers, but the chromosomes in this approach are real numbers. The number of bits in GAs is the same as number of digits in this approach. The binary position has the usual interpretation. For example, 1 0 1 1 1 0 0 1 0 1 can be converted to the decimal representation as $1 \times 2^9 + 0 \times 2^8 + 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times$

$2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$ that is equal to 741. Suppose we work with real numbers between 0 and 10, then this bit string represents the real number at the grid point number 741 of 1024 levels, that is, (741/1024)x10. It is obvious that the real numbers between grid point values are disregarded. One might argue that we can increase the number of bits to have a finer resolution; but in doing so, more computations will be involved and the GAs will take longer time to evolve to get the solutions. With the same number of positions, for example 2.553451783, there will be 10^{10} levels to work with an accuracy of $1x10^{-9}$. The same precision requires about 34 bits in binary representation. There are more positions to work with in binary representation than the real number representation with the same accuracy. The effects of mutation and crossover combination in the real number representation will be much greater than in the binary one. In this way, the modified genetic algorithm will be more efficient, more effective and easier to use in solving a real world problem encoded in real numbers. There will be no encoding and decoding requirements.

There have been several attempts to do this approach [21][19][91]. Some researchers exclude the crossover operation from the algorithm. Only mutation with addition or subtraction by a small random number is carried out. This kind of approach is often called an evolution algorithm, or sometimes a genetic hill climber algorithm. The best chromosome is kept, and the mutation process is carried out to get a new population. Then, all chromosomes in the new generation are evaluated and the best overall chromosome is kept for the next generation. This evolution algorithm may not be as global search as the traditional genetic algorithm. It uses mutation to search in the neighborhood of a number of points without the capability of combining the best features of these points. Some researchers have kept both the crossover and mutation operations. The crossover operation is modified such that it can work with real numbers. For example, Davis used the average crossover [21]. His average crossover operator takes two parents and produces one child that is the result of averaging the corresponding parents.

Other approaches tended to move away from the traditional genetic algorithm. For example, the crossover operations are modified only so that the child is a combination of two

parents, e.g., the average crossover. Unlike the traditional genetic algorithm, the representation of that child will be totally different from its parents. The child is reproduced from parents in some senses but it is not in the same sense of the crossover operation of the traditional genetic algorithm, that is, the effect of crossover operation of the existing algorithm is not the same as of the traditional genetic algorithm.

2.1.3 Modified Genetic Algorithm

The proposed genetic algorithm is modified so that it can work in real number space while preserving the traditional genetic algorithm as much as possible. This modified algorithm can be viewed as an extended genetic algorithm working in real number space.

There are four main distinctions between the modified and traditional genetic algorithm. First, the representation of chromosomes in the modified algorithm is in real numbers while it is in the binary number for the traditional genetic algorithm. For example, a chromosome is represented in 16 bits binary number as 1 1 0 0 1 1 1 1 0 1 1 0 0 1 0 0. This binary number is converted to a decimal number as 53092, then, it may scaled and shifted to a real number representation as 24.30395971618219. If there is more than one parameter, the real number representation will be separated for each parameter. Second, the crossover operation is done differently, but it serves the same purpose as a traditional genetic algorithm as shown in Table 2.1. The crossover-point is a one-point crossover. There are two options: one-point crossover for the whole individual or one-point crossover for each parameter.

Table 2.1 Crossover-point operation (the crossover-point is 7 in this case).

	Binary Representation	Real Number Representation
1st Parent	**1 0 0 1 1 1** 0 1 0 1 0 0 0 1 1 0	**1 8 . 4 3 0 7** 6 2 1 9
2nd Parent	*1 1 0 1 0 0* ***1 1 1 0 1 0 0 1 1 1***	***2 4 . 8 0 3 3 8 7 5 0***
Offspring	**1 0 0 1 1 1** ***1 1 1 0 1 0 0 1 1 1***	**1 8 . 4 3 0 7** ***8 7 5 0***

Third, the mutation is also done differently, using the position significance of each digit. For example, 1 8 . 4 3 0 7 8 7 5 0 can be viewed as $1 \times 10^{1} + 8 \times 10^{0} + 4 \times 10^{-1} + 3 \times 10^{-2} + 0 \times 10^{-3} + 7 \times 10^{-4} + 8 \times 10^{-5} + 7 \times 10^{-6} + 5 \times 10^{-7} + 0 \times 10^{-8}$. The mutation is done for each position as same as in the traditional genetic algorithm. Each position is added or subtracted by a random number having significance up to that position if the probability test is passed. For example, suppose the given chromosome is 18.43078750 and the threshold test is 0.2. Then, the procedure is carried out as shown in Table 2.2. The new chromosome after mutation would be the sum of 18.43078750 and 0.342298689, that is 18.773086189. To reduce the computation, the mutation can be further modified such that only the random number of the most significant position is added. For this example, the most significant position in which its probability test is passed is the position 2, then only 0.342298768 will be added to the original chromosome and the result will be 18.773086268. In this way, the mutation is done in a similar fashion as the traditional genetic algorithm. Other researchers use other mutation schemes. For example, Davis replaces the whole chromosome with a new random number if the probability test is passed. He also uses other methods such as moving around the chromosome by a small random amount defined by the designer if the probability test is passed.

The fourth difference is the accuracy issue. In the traditional genetic algorithm, the accuracy is increased as the number of bits is increased. As the number of bits increases, the evolving time tends to be longer since there are more points to consider. One reason that the traditional genetic algorithm is very effective in finding solutions is that its working space is finite. Its working space is only discrete levels of the real number space. If the number of bits in the two-dimensional optimization problem in the next section is 16, the value of x and y can be from 0 and incremented approximately by 0.000457771 to 30. In other words, there are 65536 levels in the working space of this problem. Consequently, a solution may not be optimal. Also, its search cannot be claimed as a global search. The ideal case would be to start with less accuracy and increase it as time evolves to get the final solution. Since the genetic algorithm alone has very complex behavior, such modification cannot be done easily and effectively. As of today, there is

no report of doing so. One reason might be that the change in number of bits affects the change in grid values of its working space. Every time the number of bits is increased, the old binary representation has to be altered such that it maintains its current value. It has to be converted to the real number value, and then converted back to binary representation with a new number of bits. This process will sometimes alter its value. This approach will be shown in the optimization problem later. On another hand, the accuracy in the real number representation can be increased or decreased easily. For example, for an old value of 12.03445, the new value could be 12.034450. Its value is not altered, but the precision is increased. The least significant digit will be introduced in the mutation process later. In this way, increasing accuracy during the evolving time can be done easily and effectively in the modified genetic algorithm. This method will improve accuracy and reduce evolving time of the algorithm.

Table 2.2 Example of mutation process

Position	Probability Test	Random Number to add/subtract
1	0.218959186	0
2	0.047044616	0.342298768
3	0.678864717	0
4	0.679296406	0
5	0.934692896	0
6	0.383502077	0
7	0.519416372	0
8	0.830965346	0
9	0.034572111	-0.000000086
10	0.053461635	0.000000007
Total addition		0.342298689

The rest of the modified algorithm such as the parent selection process is the same as the traditional genetic algorithm. All good characteristics of the traditional genetic algorithm are preserved in the modified genetic algorithm.

The modified genetic algorithm may be summarized as follows:

a) Initialize a population of chromosomes using real number representation.

b) Evaluation. Evaluate each chromosome in the population. Store their fitnesses.

c) Parent selection. Select parents according to their fitnesses. Parents having more fitness have more chance to be selected.

-Set all individuals in the current generation as candidate parents, except the one having the lowest fitness.

-The best individual of all previous runs is always kept as a candidate parent (elitism). Its fitness is also kept.

-A roulette wheel parent selection:

1. Convert fitness to percentages of fitness.
2. Add up all percentages of fitness to unity.
3. Generate two random numbers between 0 and 1 for each offspring.
4. Select two parents having sum of percentages of fitness right above the random numbers in 3.

d) Reproduction. Create new chromosomes by mating parents. Apply recombination and mutation.

-One-Point Crossover Operation for each parameter:

1. The crossover-point is randomly generated between 1 and a preset maximum decimal value.
2. The offspring is a combination of the first parent having a value down to the crossover-point and the second parent having a value up to the crossover-point.

Example:

First parent = **0.250753**826956728

Second parent = 0.701027**073504845**

The crossover-point is 6, then

Offspring = **0.250753**073504845

-One-point crossover operation for each individual consisting of more than one parameter:

1. The crossover-point is randomly generated between 1 and a preset total number of digits for the whole individual. For example, suppose there are 3 parameters in the individual and there are 10 digits for each parameter. The crossover point is then randomly generated between 1 and 30.
2. Find the parameter in which the crossover point occurs. The parameters before this parameter come from the first parent and the parameters after this parameter come from the second parent. The parameter in which the crossover point occurs is a combination of the first parent having a value down to the crossover-point and the second parent having a value up to the crossover-point.

- Mutation operation:

1. Check the probability test. For each position of each chromosome, generate a random index number and compare with the preset index number
2. If the random index number is less than the preset index number, the probability test is passed, then generate a random number having its significant up to its position as shown in Table 2.2. The second position is passes, the the maximum number can be is 0.999999 … Its value can be less.
3. There are two approaches at this point of operation.

   First approach, add all random numbers of passed tests to the original chromosome. That is the total addition in Table 2.2.

   Second approach, add only the most significant one to the original chromosome. That is 0.342298768 in Table 2.2.

   If none is passed, none will be added to the original chromosome.

e) Each offspring is then rounded off to be at most desired accuracy. This desired accuracy is started with less accuracy, and increased over time. In other words, each

offspring is rounded off to have the desired number of decimal digits. This desired number of digits can be increased linearly or nonlinearly in the search. This space will be gradually bigger as the number of digits gets larger. This will help the algorithm search faster.

f) Make room and insert the new chromosomes into the population.

g) If time is up, stop and return the best chromosome. If not, go to (b).

2.1.4 Two-Dimensional Optimization Results

To clarify these differences, a two-dimensional optimization problem is proposed. It can be formulated as finding the maximum of the following function :

$$z = x^{\frac{1}{3}}\left(\sin(x)+1\right)y^{\frac{1}{3}}\left(\sin(y)+1\right) \qquad (2.5)$$

This function is used as the fitness function of both a traditional genetic algorithm and the proposed algorithm. The values of x and y are real numbers in the range of 0 and 30. The purpose is to optimize this function. In other words, the purpose is to find x and y at the maximum point of this function. The solutions are that x and y are approximately equal to 26.7284784480289 and the maximum value of the function is approximately equal to 35.7471209375290. These values are the values of $x_{solution}$, $y_{solution}$, and $\mathrm{Pi}_{\mathrm{solution}}$ in Tables 2.3, 2.4, and 2.5, respectively. This function is shown in Figure 2.5.

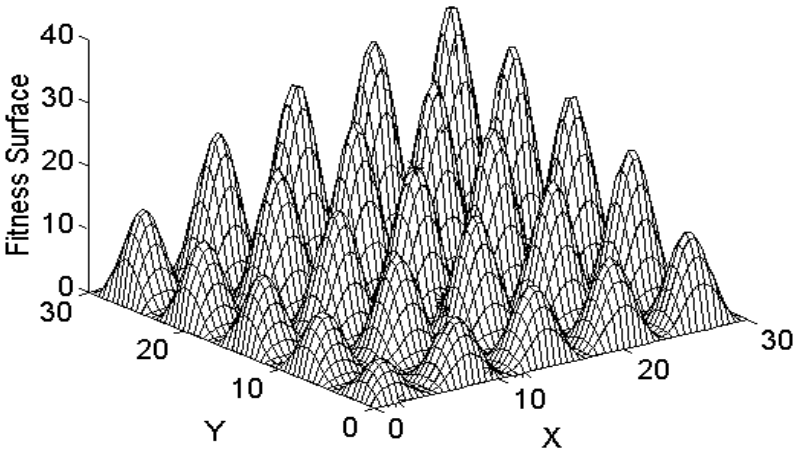

Figure 2.5 Testing fitness function

This two dimensional optimization problem is used as a benchmark problem for various studies of both traditional and modified genetic algorithms. There are twelve individuals in the population, and they are the same for all algorithms. Their control parameters such as preset numbers for probability test are also most likely the same unless different algorithms have different suitable values. For example, the preset number for probability test should be a small number, e.g., 0.005-0.1 in the traditional genetic algorithm. On the other hand, the preset number for the probability test in the modified genetic algorithm is bigger, e.g., 0.15-0.35. These numbers must be tuned in the same manner as the learning rates in ANN. The initial conditions are also the same for all algorithms. All algorithms are allowed to evolve 300 generations. The elitism in which the best chromosome is passed on to be the candidate parents for the next generation is used for all algorithms.

The 1st traditional genetic algorithm has a fixed number of bits at 48 bits for each parameter. Then $30/(2^{48}-1)$ is equal to 1.0658×10^{-13} so that the accuracy is up to about 13

decimal places. The one-point crossover is used for each parameter. The preset number for probability test in mutation process is set at 0.1. A bit is reversed if its probability test is passed.

The 2nd traditional genetic algorithm has an increasing bits mode. Its numbers of bits start at 20 bits for each parameter and they are increased by 1 up to 48 bits every 5 generation. All other control parameters are the same as the first algorithm.

The 3rd traditional genetic algorithm has all control parameters the same as the first one, except that only a one-point crossover for the whole individual is used.

The 1st modified genetic algorithm sets the number of digits at 7 digits. This number of digits indicates that the crossover point is within this number. It is also used in the probability test in mutation process. This number is fixed and can be viewed as the number of bits in the traditional genetic algorithm case. The minimum accuracy is set at 5 decimal places and the maximum accuracy is set at 13 decimal places. This accuracy is increased linearly from the first generation to the last generation. A one-point crossover operation for each parameter is used. The first option of mutation method explained in previous section is used. The preset number for the probability test in the mutation process is set at 0.2.

The 2nd modified genetic algorithm has the same control parameters as the 1st modified genetic algorithm except the mutation method used. This 2nd modified genetic algorithm uses the second mutation option explained in the previous section.

The 3rd modified genetic algorithm also has the same control parameters as the 1st modified genetic algorithm except that one-point crossover for the whole individual is used, and it uses the second mutation option explained in the previous section.

The 4th modified genetic algorithm also has the same control parameters as the 1st modified genetic algorithm except that the average crossover method and different mutation

method are used. The mutation is done by adding/subtracting to the real number chromosome a small random number if the probability test is passed.

Table 2.3 Absolute difference between the *best* solutions in 100 runs and ideal solutions of each algorithm

Algorithm	$\lvert x-x_{solution} \rvert$	$\lvert y-y_{solution} \rvert$	$\lvert PI-PI_{solution} \rvert$
GA#1	0.0000647661634	0.0000713876794	0.0000000831207
GA#2	0.0000877271904	0.0000172525078	0.0000000715158
GA#3	0.0000920840266	0.0000604988634	0.0000001086069
MGA#1	0.0000125816711	0.0000086690511	0.0000000020886
MGA#2	0.0000181169719	0.0000062671289	0.0000000032878
MGA#3	0.0000133133659	0.0000045630289	0.0000000017720
MGA#4	0.0000741121211	0.0002183370389	0.0000004756289
MGA#5	0.0052644480289	0.0087225580289	0.0009285851977

Table 2.4 Absolute difference between the *worst* solutions in 100 runs and ideal solutions of each algorithm

Algorithm	$\lvert x-x_{solution} \rvert$	$\lvert y-y_{solution} \rvert$	$\lvert PI-PI_{solution} \rvert$
GA#1	0.0580120004922	0.0056387231975	0.0303914874889
GA#2	0.0099590031115	0.0243693367409	0.0061993717903
GA#3	0.0122451068150	0.0770725926830	0.0544400531091
MGA#1	0.0024105433610	0.0002023058628	0.0000523523076
MGA#2	0.0031650299711	0.0002013743211	0.0000899847891
MGA#3	0.0020650626501	0.0015739644051	0.0000603164333
MGA#4	0.0248293057389	0.0064351162611	0.0058850567624
MGA#5	0.5002011380289	0.0470996280289	2.2062311068008

The 5th modified genetic algorithm has no crossover point operation; only mutation is used. The preset number for probability test in mutation process is set at 0.45. Other control parameters are the same as in the 1st modified genetic algorithm.

Table 2.5 Absolute difference between the *average* solutions of 100 runs and ideal solutions of each algorithm

Algorithm	$\lvert x-x_{solution} \rvert$	$\lvert y-y_{solution} \rvert$	$\lvert PI-PI_{solution} \rvert$
GA#1	0.0029627888256	0.0031143771737	0.0010027900938
GA#2	0.0043720305494	0.0042777130517	0.0009713043170
GA#3	0.0032124179668	0.0048732782753	0.0013101730470
MGA#1	0.0000589922240	0.0000210758283	0.0000042559966
MGA#2	0.0000435366872	0.0000053548879	0.0000049364962
MGA#3	0.0001141334960	0.0000880902973	0.0000043434549
MGA#4	0.0015257914052	0.0002261840455	0.0007335964491
MGA#5	0.0262410301349	0.0088539360429	0.3090361067770

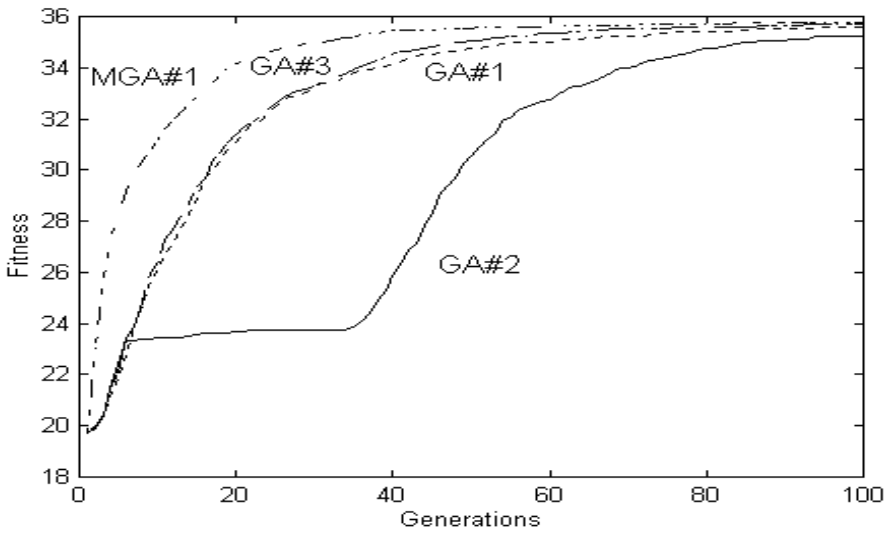

Figure 2.6 Average of the best individual found of 100 runs

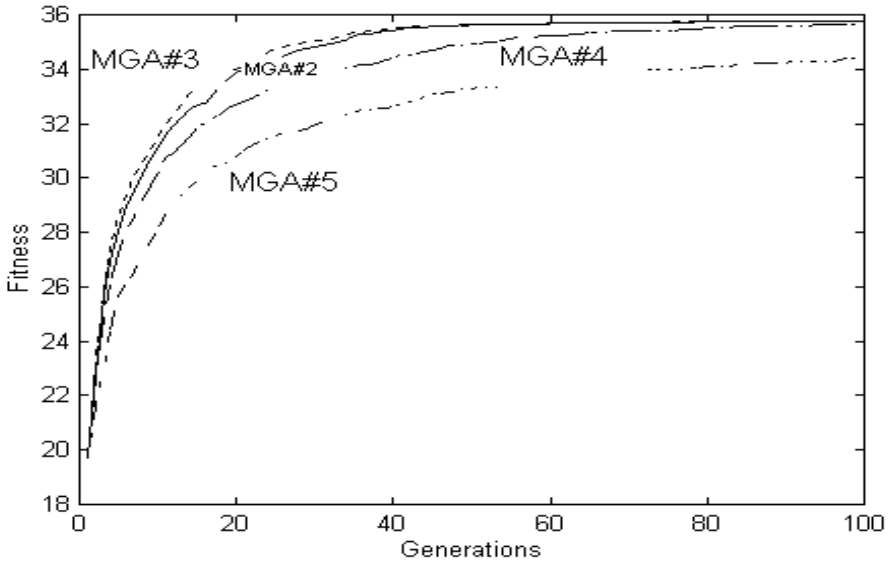

Figure 2.7 Average of the best individual found of 100 runs

The results show that the proposed modified genetic algorithms perform better than both the traditional genetic algorithm and the modified genetic algorithm proposed by other researchers. The average solutions of the modified genetic algorithms are closer to real solutions than other methods in 100 runs. They show efficiency and consistency. Their worst results are still better than the average results of other methods. The modified genetic algorithms have some advantages over the traditional genetic algorithms. Its real number representations are more compact than the binary representations, and it has fewer positions to work for the same accuracy. Its accuracy can be increased easier than those using binary representation; thus, their convergence rates are higher than the traditional genetic algorithms. It looks for the solutions in fewer resolutions in the beginning and more resolutions in the end.

2.1.5 Circular Array Design Using Modified Genetic Algorithms

The modified genetic algorithm is applied to a circular array antenna design. Although the circular array has an interesting property, i.e., constant sidelobe levels, it is not widely used. The first reason is that there is a lack of circular array design methods available, unlike the uniformly

spaced planar array. The second reason is that it is hard to implement a circular array. To overcome these disadvantages, the modified genetic algorithm is used. Any constraint can be put into this design process. This design is slightly different from [103] in the following constraints imposed. The first constraint is the spacing between elements. The upper constraint of the spacing between element is one wavelength of the upper interested frequency that is 9000 Hz. The lower constraint of the spacing between element depends on the physical size of the element. In this design, the lower constraint of the spacing between element is set at a half wavelength of the upper interested frequency. The second constraint is the accuracy in measurement of its radius. The radius of each ring can have accuracy up to 1 millimeter. The third constraint is the accuracy in measurement of the angle of each element. The angle accuracy of each element is set at a half degree. The last three constraints are only set to be able to build a prototype array manually in this study. These constraints may be lowered if better instruments are available.

The necessary parameters are the radius of the first and last ring, the spacing between rings, the number of elements in each ring and the angle of the first element in each ring. The minimum radius of the first ring is set to a half wavelength of an acoustic signal at frequency 9000 Hz that is 19 mm. The maximum radius of the first ring is set to 5 times of the minimum value that is about 95 mm. The number of rings is fixed at 7 rings in this design. The minimum and maximum radii of the last ring are set to 12 and 16 inches, respectively. In the frequency independent antenna design such as log-periodic antenna, the ratio of frequency is kept the same [5],[71], [72]. In this design, the ratio between the spacing of adjacent rings is kept equal. They can be put into equations as

$$\frac{d_1}{d_2} = \frac{d_2}{d_3} = \ldots = C \qquad (2.6)$$

where d_i is the spacing between rings i^{th} and $(i+1)^{th}$. This constant, C, is allowed to be between 0.85 to 1, and will be used to determine the remaining radii. The number of elements in each ring and the angle of the first ring will determine the angle of all elements. The spacing of elements in the same ring is almost the same. In other words, they are uniformly spaced in each ring. The maximum or minimum number of elements in the ring is determined by the maximum and

minimum spacing allowed, respectively. The angles of the first element in each ring are set to be between 0 and 90 degrees. In this way, the parameters to be optimized are the radius of the first ring, radius of the seventh ring, the constant ratio, 7 angles of the first element in each ring and 7 numbers of elements in each ring, i.e. there are 17 parameters to be optimized as shown in Figure 2.8.

The fitness function used in this problem involves the maximum sidelobe level (dB) in the across-the-lane and the along-the-lane direction and the total number of elements in the array. The first two components are measured relatively to the peak value. The last component is positive. The overall fitness will be negative, but it will be made positive by the percentage of fitness process. The fitness function is set as:

Fitness = Maximum Sidelobe Level in Across-the-Lane Direction (dB) +
0.75 x Maximum Sidelobe Level in Along-the-Lane Direction (dB) +
0.1 x Total Number of Elements in the Array.

The preset number in the probability test is 0.1. The maximum number of generations is 500 generations. The accuracy of the constant, *C*, is set to start at 5 decimal points and stop at 15 decimal points. Other accuracy is bounded by the given constraints. The number digits indicating the maximum crossover-point is fixed at 5 digits.

The results are shown in Table 2.6. The traditional genetic algorithm is used as comparison. The average results are obtained by 25 runs. The best design that used the modified genetic algorithm is shown in the final column. The final array gains corresponding to the frequency at 9000 and 5000 Hz are shown in Figure 2.9 and 2.10, respectively. We can see that the sidelobe levels are constant. The design of circular array using the modified genetic algorithm was clearly successful.

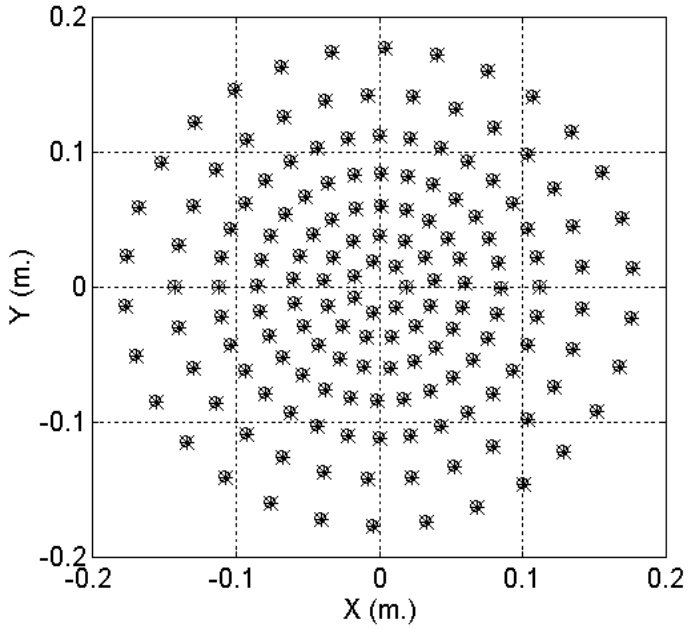

Figure 2.8 Circular arrays of microphones

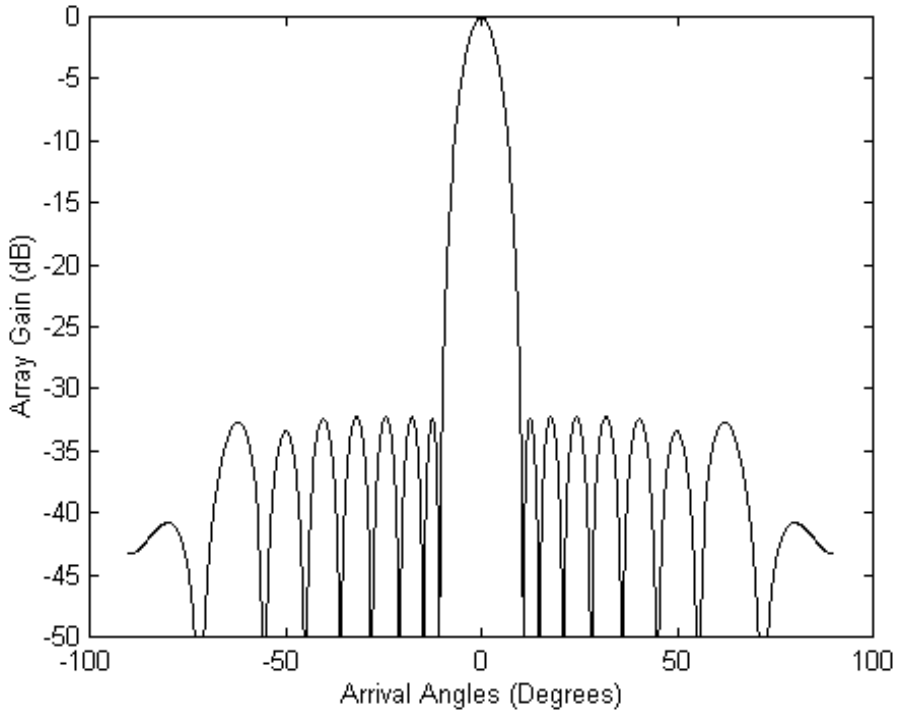

Figure 2.9 Array gains (Across-the-Lane) corresponding to frequency at 9000 Hz.

Table 2.6 Results of GAs (average), MGAs (average) and MGAs (best) for circular arrays design problem

	Traditional GAs (Average)	**Modified GAs (Average)**	**Modified GAs (Selected)**
Radius #1 (cm.)	**1.91**	**1.90**	**1.90**
Radius #7 (cm.)	**15.94**	**15.73**	**15.30**
Constant, *C*	0.875078	0.878313	0.8908
1st Angle of 1st ring (^{o})	41.275	48.525	0
1st Angle of 2nd ring (^{o})	29.225	15.925	90.00
1st Angle of 3rd ring (^{o})	32.85	17.40	88.58
1st Angle of 4th ring (^{o})	26.05	22.725	89.15
1st Angle of 5th ring (^{o})	34.25	25.875	90.00
1st Angle of 6th ring (^{o})	29.85	24.525	43.33
1st Angle of 7th ring (^{o})	27.65	17.60	88.68
Number in 1st ring	8.05	7.9	8
Number in 2nd ring	10.35	10.30	10
Number in 3rd ring	19.40	19.15	19
Number in 4th ring	24.85	24.20	23
Number in 5th ring	30.85	29.65	29
Number in 6th ring	26.85	26.20	26
Number in 7th ring	28.15	27.60	28
Total Number of Elements	**148.5**	**145.0**	**143**
Maximum sidelobes Across-the-Lane (dB)	**-31.401**	**-32.162**	**-32.253**
Maximum sidelobes Along-the-Lane (dB)	**-31.245**	**-32.092**	**-32.170**

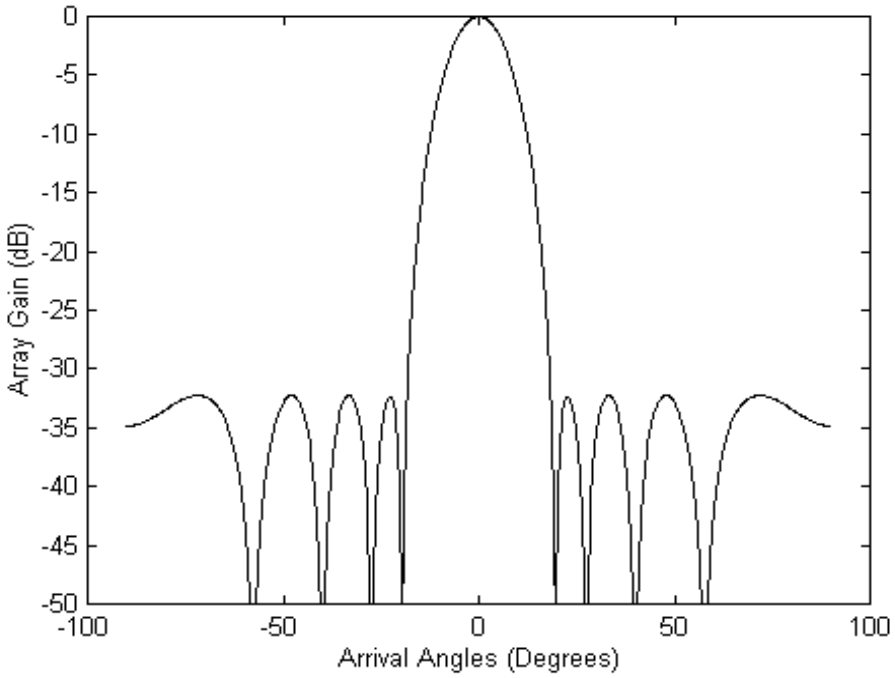

Figure 2.10 Array gains (Across-the-Lane) corresponding to frequency at 5000 Hz.

2.1.6 Sensitivity Study

The best array result of the circular array design was selected to be built as a prototype. In the design all elements are assumed to have unity gains. All variations of microphones, resistors, capacitors, operational amplifiers, and location of the elements will affect the final performance of the array. All electronic components will affect the unity gain assumption. Inexact location of microphones will affect their radii and relative angles to the center of array. Thus, the errors can be categorized as radius, angle and gain errors. The performance index used in this study combined the maximum sidelobe level and the 30 dB beamwidth in both across-the-lane and along-the-lane directions.

The error distributions in this study are assumed to be normal distributions with zero means. The result of each error is the average error over 50 simulation trials.

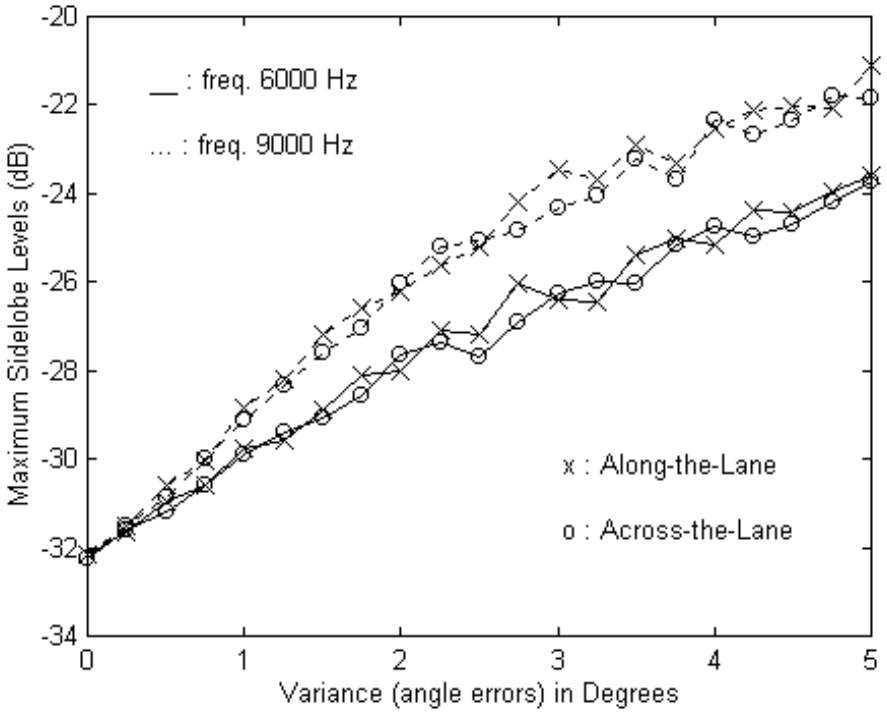

Figure 2.11 Angle errors versus maximum sidelobe levels (dB)

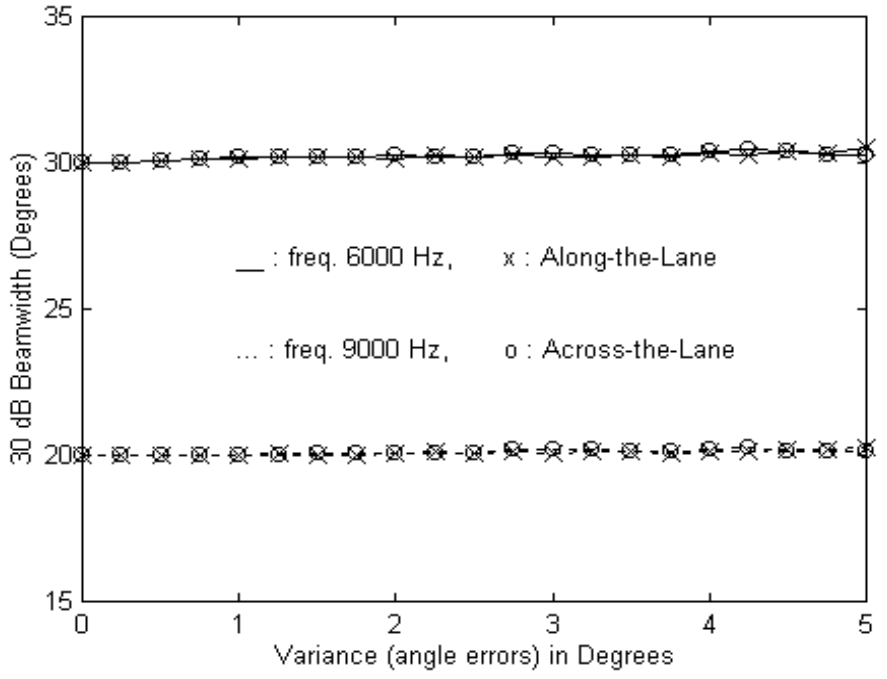

Figure 2.12 Angle errors versus 30 dB beamwidth

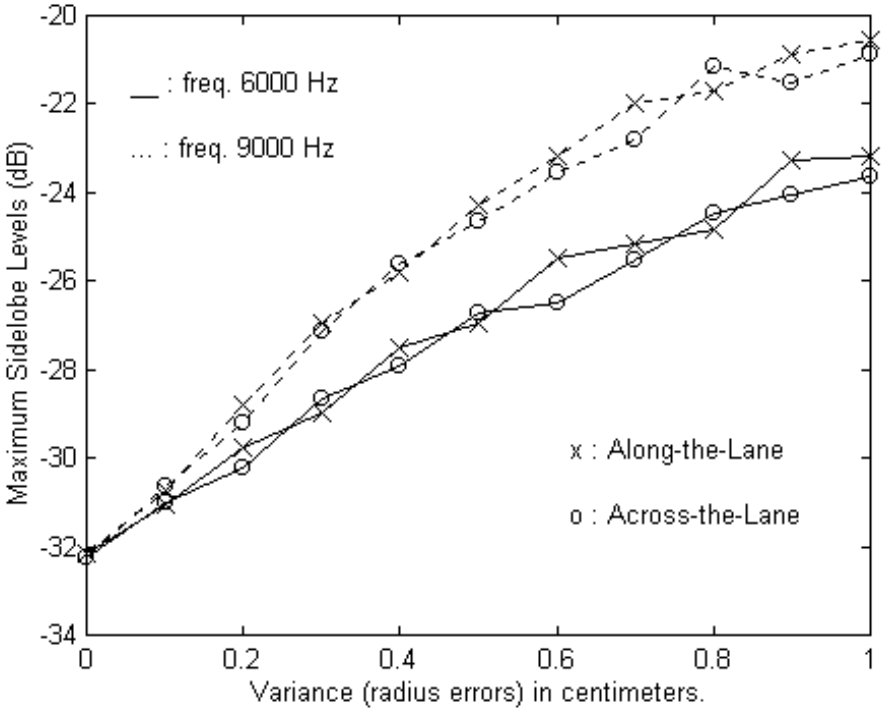

Figure 2.13 Radius errors versus maximum sidelobe levels (dB)

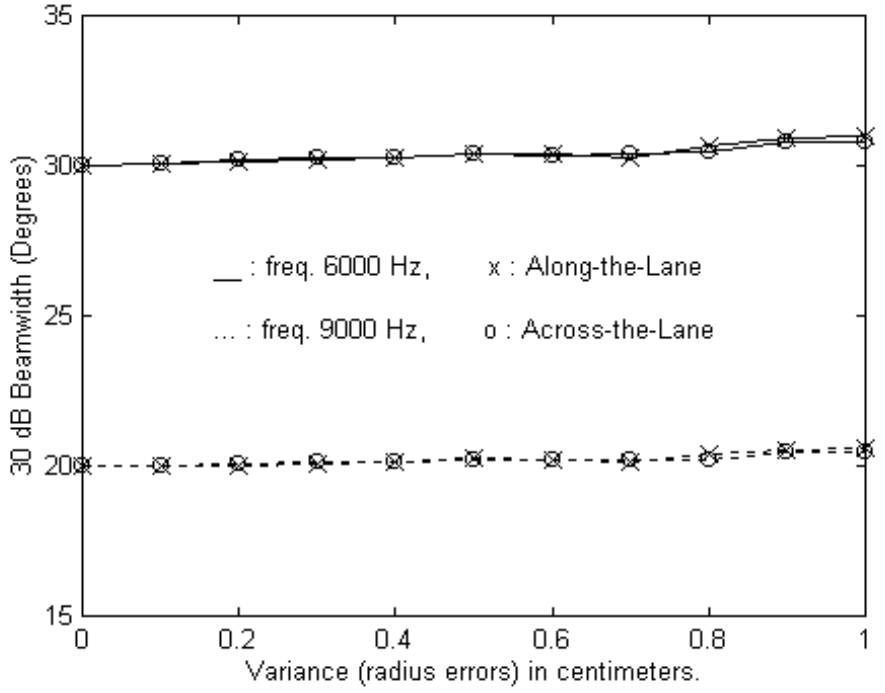

Figure 2.14 Radius errors versus 30 dB beamwidth

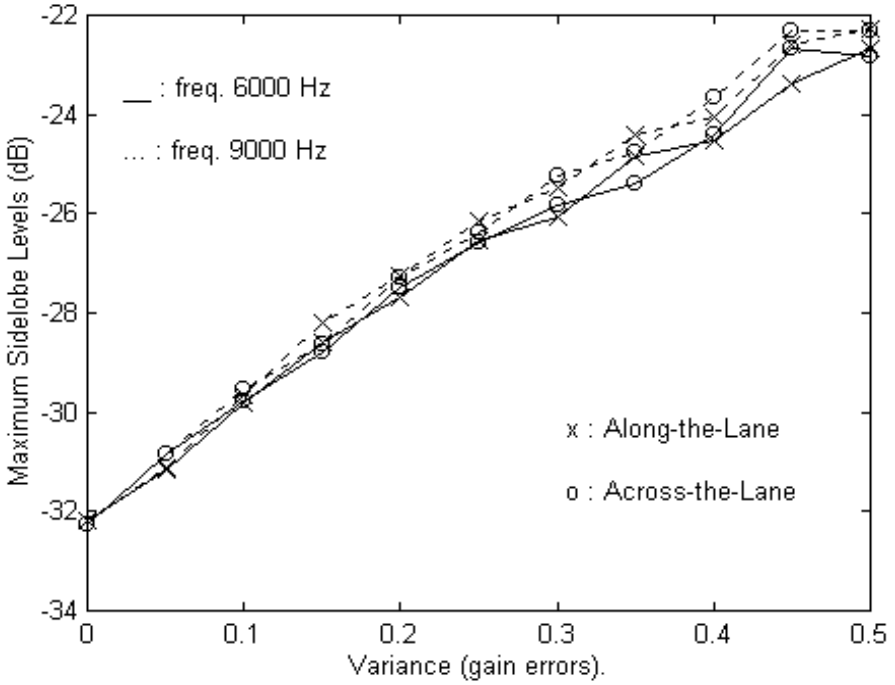

Figure 2.15 Gain errors versus maximum sidelobe levels (dB)

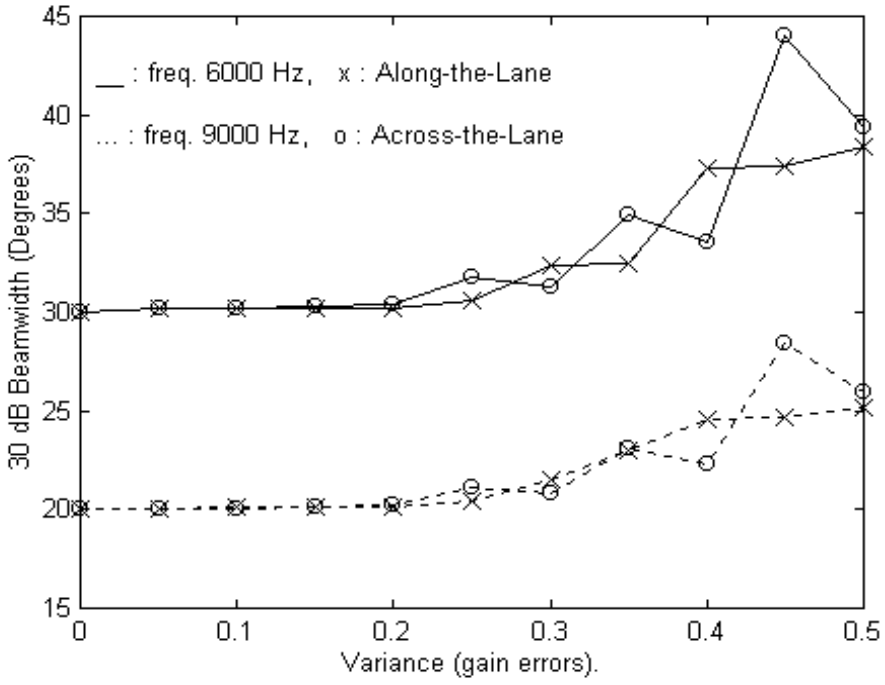

Figure 2.16 Gain errors versus 30 dB beamwidth (dB)

All errors affect the performance of the array. Angle errors and radius errors have small effects on the beamwidth but the gain errors have significant effects on the beamwidth. All errors have almost proportional effects on the maximum sidelobes levels as the variances increase. The gain error is the most significant error among all errors because it affects both sidelobes levels and beamwidth. This error will play an important role of the success or failure of this circular array implementation.

2.1.7 Prototype Circular Array Implementation

The prototype circular array is built to be used for data collection. There are several stages in this array implementation. First, the circuits are designed for the overall array. They comprise microphone parts, first-order highpass filters, summer for each ring, and a summer for the overall array as shown in Figure 2.17.

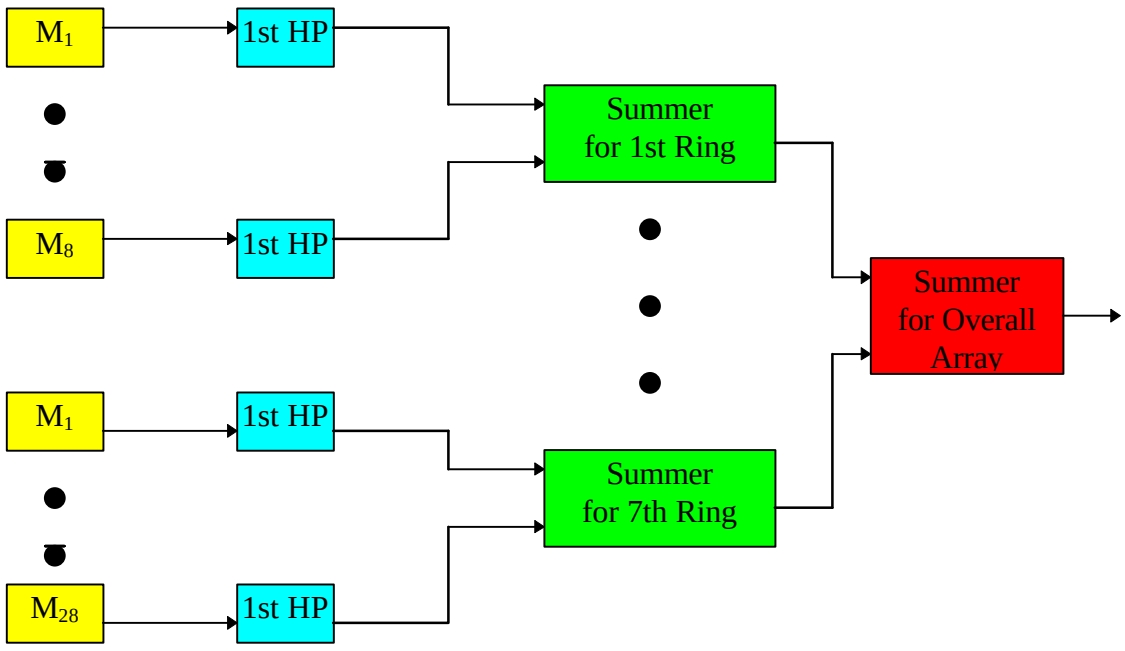

Figure 2.17 Overall array system

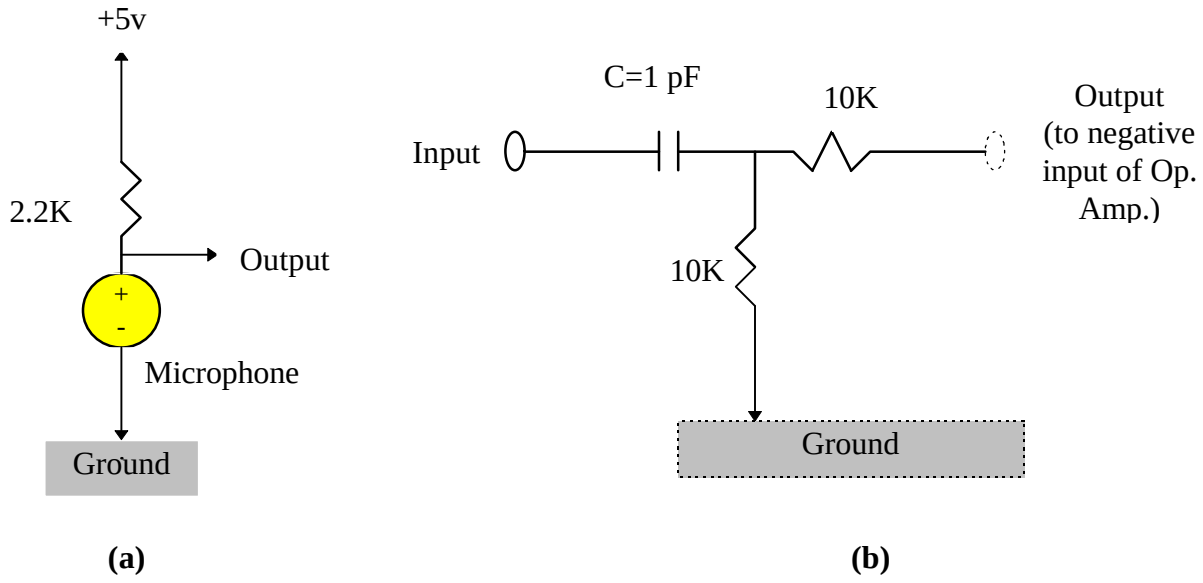

Figure 2.18 (a) Microphone part, (b) 1st order highpass filter

A 1st order highpass filter is used for each microphone to filter low frequency energies of the vehicle acoustic signals. Each highpass filter has its cut-off frequency at about 3000 Hz. The summer circuits are simple summing amplifiers using op amps. There are total of 8 summer circuits, one for each ring and the last one for the overall array.

The next step is to construct the locations of all microphones. In this stage, it is quite hard to do manually, a definite drawback of this circular array. Unlike a uniformly spaced planar array, the locations are far from the ideal case. The locations of all microphones were measured and drilled with tools available at a hardware store. The protractor has 0.5 degree precision. The measurements and constructions of the locations were done by hand. If this circular array is constructed by a better machine, e.g., by machine precision in a production line, this drawback will be diminished. Since this prototype circular array must be done manually, there are errors in measurements and constructions of both radius and angle for each element. These errors will play important roles in the final performance of the array.

The next step is to determine a gain for each microphone. All microphones are the same type and should have very similar characteristics. To verify this assumption, testing must be done. The testing experiment is set up such that we can capture the microphone characteristics corresponding to wide range of frequencies. A microphone circuit as shown in Figure 2.18(a) is used. A source of sound is a computer speaker. The sound has a sinusoidal waveform with frequency varying linearly from 5000 to 9000 Hz in 5 second duration. This sound is generated by a commercial wave editor, so there is at least consistency in this part. This set up is used for all other following tests. Each microphone is identified and tested with the same set up, the resistor in the circuit and power supply. The output of the circuit is recorded into audio input channels of a VCR while the video image of each microphone name tag comes from the camcorder. In this way, it is easy and consistent for all microphones in the playback part. In the playback, the same VCR is used. This VCR was tested to see what noise would be introduced into the audio signals. It turned out that this VCR is one of the best available. The playback audio signals from the output channel of VCR are preprocessed and amplified by a second order low-pass filter having cutoff frequency of about 10,500 Hz before being sampled. The signals are sampled by a sound card with a sampling rate of 44100 Hz. The sampled signals are stored in a file for each microphone. With the capability of the wave editor software, the starting and stopping points of the audio signals can be identified. Then the sampled audio signals for each microphone are read in from a file and filtered by a 19th order low-pass and a 16th order high-pass filter having cutoff frequencies at 10500 and 3500 Hz, respectively. The audio signals have frequencies varying from 5000 to 9000 Hz. These signals are divided into 50 windows with no overlapping. Each window will have 80 Hz variation in frequency and 0.1 second duration. Then, the filtered signals have their energy calculated in each window. This process is done for all 152 microphones. The results of 152 microphones are shown in Figure 2.19. We can see that the characteristics of each microphone are varying with frequency quite widely.

A relative gain of each microphone is determined in the next step. The average of each window is calculated from all microphones. Then, a gain of each window for each microphone is

a result of its energy in that window divided by the average energy for that window. In this way, the gain will be less than one if its energy is below the average value and more than one otherwise. At this point, there are 50 gains but we need only one gain for each microphone. The final gain is calculated by a weighted average. We set a nominal frequency at 7500 Hz. At this nominal frequency, the weight is one. Then, the weights are linearly decreased when the frequency deviates from this nominal frequency. The weight is set at 0.6 at frequency 5000 Hz and at 0.75 at 9000 Hz. In this way, we emphasize the gain at the nominal frequency. The histogram of the results is shown in Figure 2.20. We can see that the results are not very promising. This frequency dependent variation of microphones will play the most significant role in the final performance of the array.

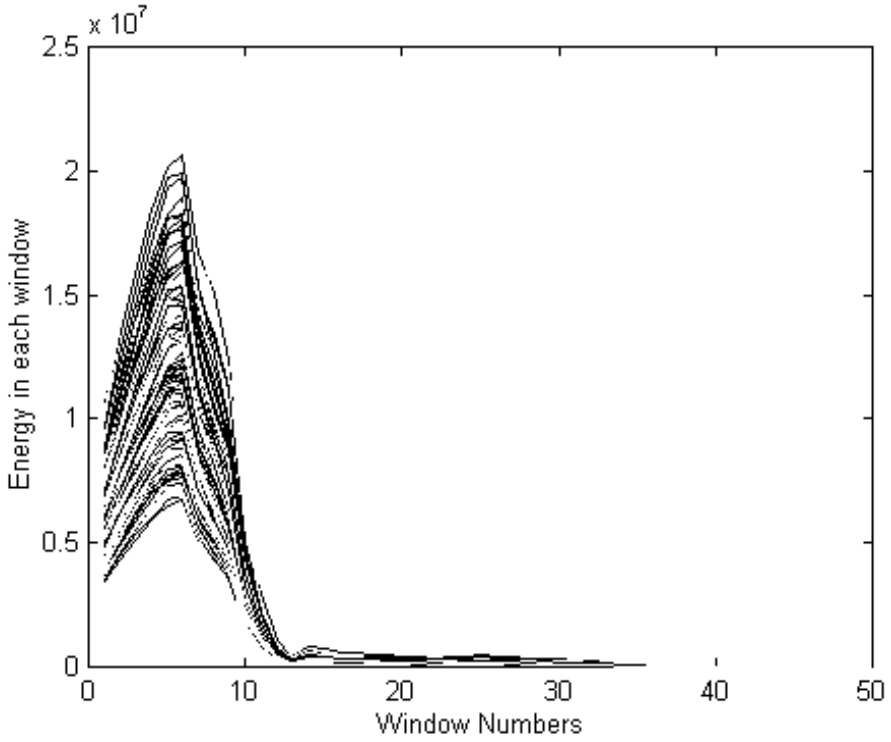

Figure 2.19 Energy for each window for all 152 microphones

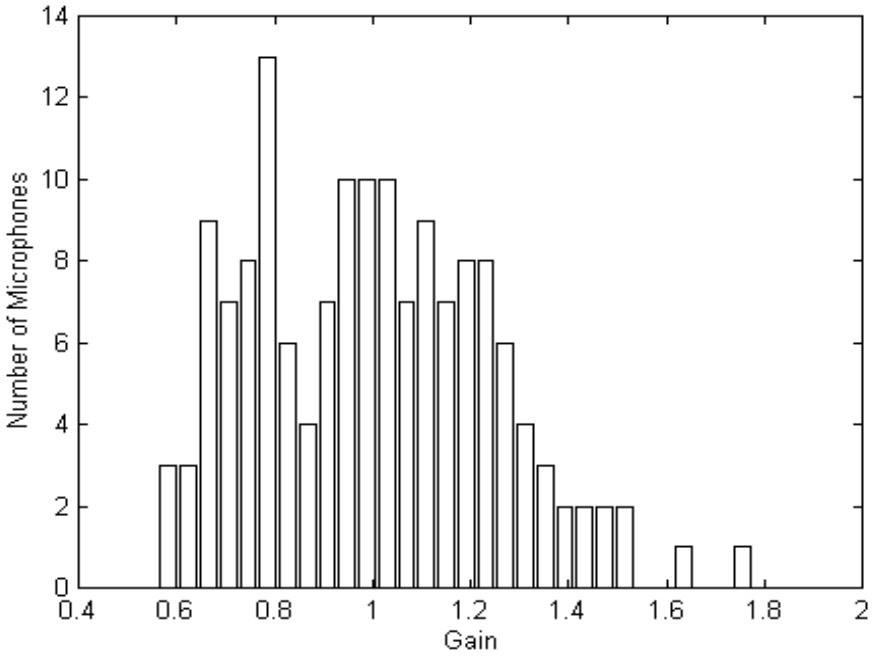

Figure 2.20 Histogram of gains for all 152 microphones

The next step is to determine the gain for each path in the summer circuit. The feedback resistor is adjustable for all summer circuits while others are fixed values. At this testing stage, the microphone circuit, 1st order high pass filter and summer circuit for each ring are combined as shown in Figure 2.21. All elements have about a 5% variation from the nominal values. The microphone having the gain closest to 1 from the previous test is used for all paths. All feedback resistors are set at 8.5K ohms. The testing set up is the same as previous test. At this time, the output of each summer passing through one 10K ohms resistor is recorded into VCR. All procedures are also the same.

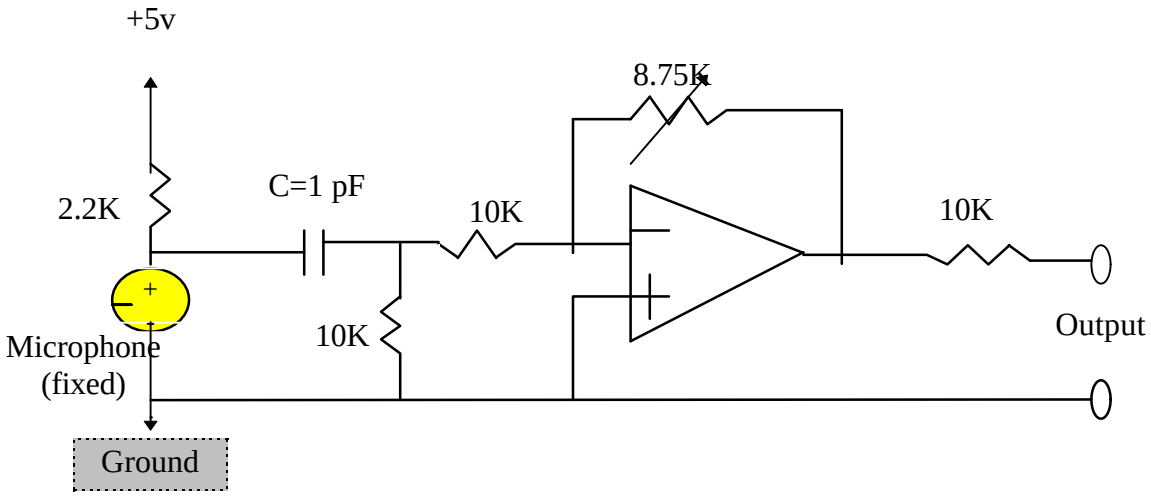

Figure 2.21 Example of each branch circuits testing

The results are shown in Figure 2.22 and 2.23 respectively.

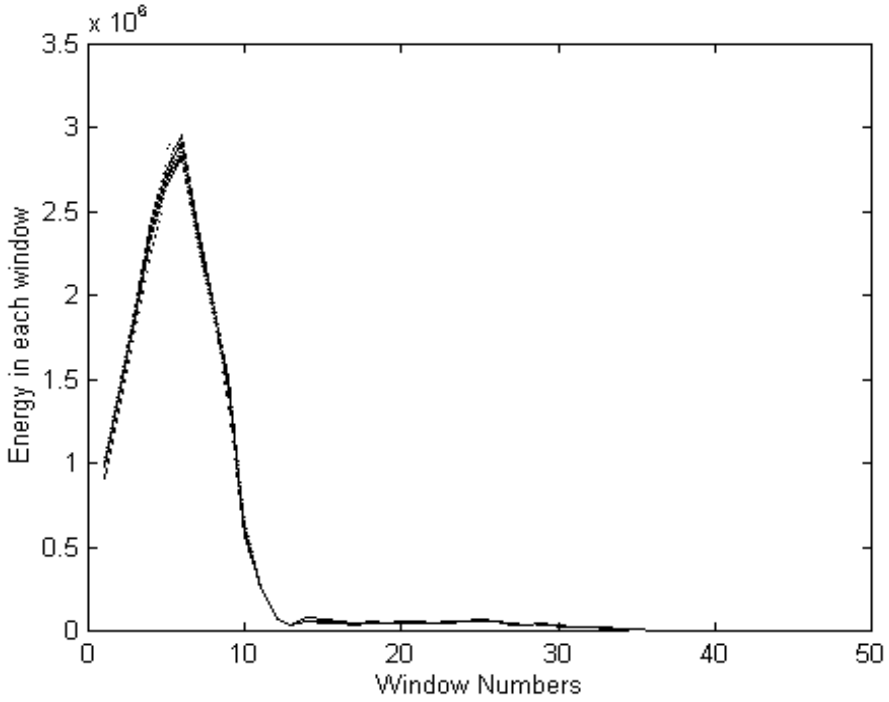

Figure 2.22 Energy for each window for all branches of all summers

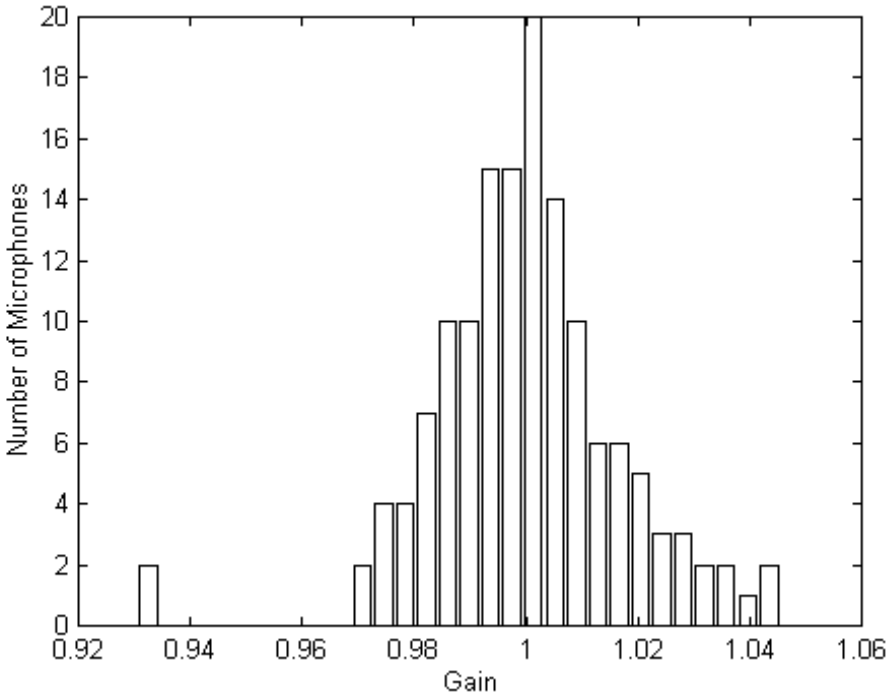

Figure 2.23 Histogram of gains for all branches of all summers

The next process is to allocate each microphone to each branch and determine the feedback adjustable resistance for each summer. This process is done by using the modified genetic algorithm to select and allocate 143 out of 152 microphones and adjust the feedback resistor so that all branches have their gains close to 1. The accuracy constraint of the resistance value is set at 0.01 since that is the highest accuracy of the available multimeter. The results of the feedback resistance are 8.97K, 7.09K, 9.34K, 9.90K, 7.56K, 8.06K and 8.52K ohms for ring 1 to 7, respectively. Then each microphone is mounted and connected to each branch according to its assignment.

The next stage is to test each branch with the assigned microphones. The testing procedures are the same as previous test, except, the testing audio signals are varied from 6000 Hz to 9000 Hz with 7.5 second duration. The window size is still the same. In this way the

frequencies are varied 40 Hz in 0.1 seconds for each window. The testing results are shown in Figure 2.24 and 2.25.

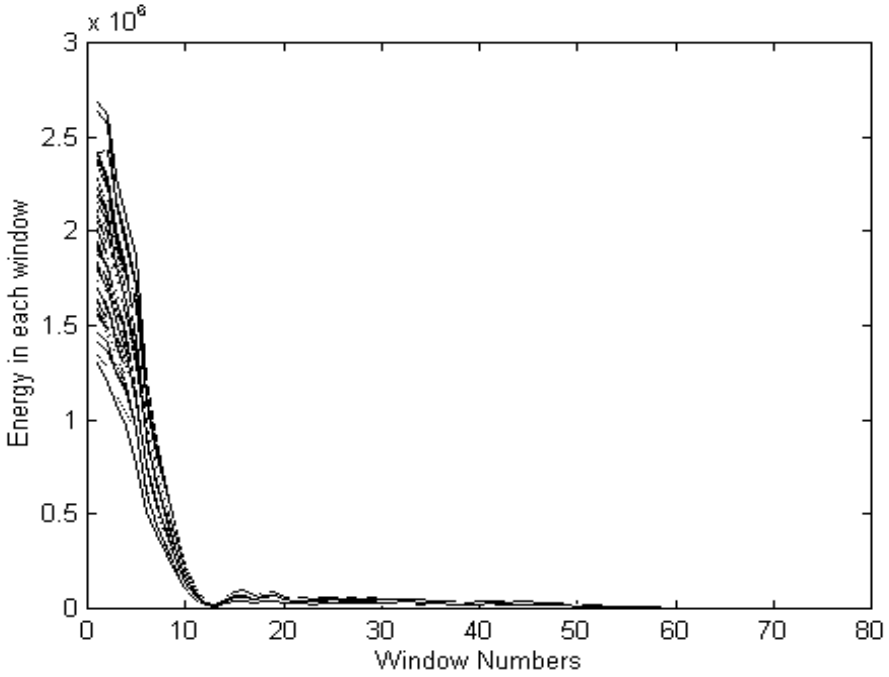

Figure 2.24 Energy for each window for all branches and microphones of all summers

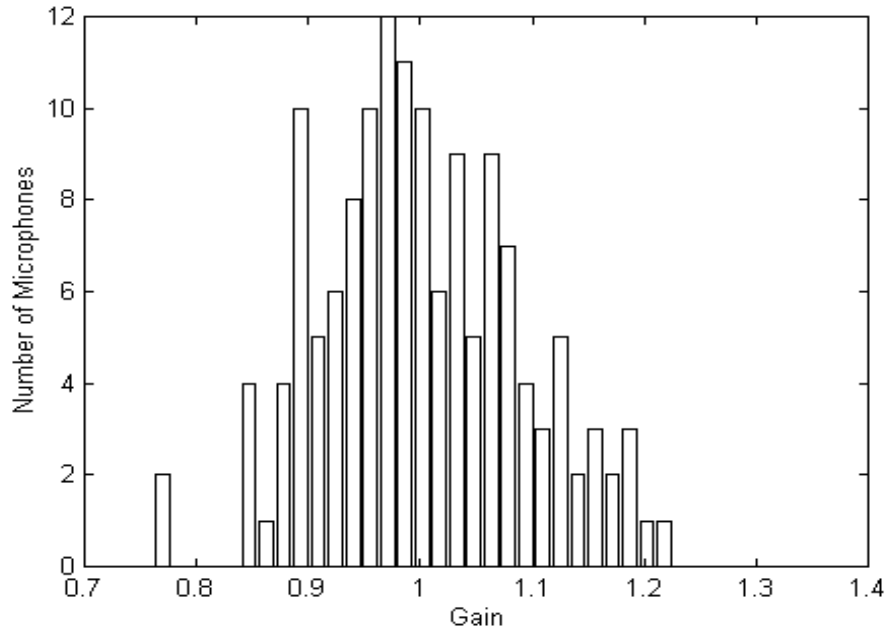

Figure 2.25 Histogram of gains for all branches and microphones of all summers

The gains are so far from unity that the unity gain assumption is strongly violated. It seems to be that we have to re-design the array. Since all construction and circuits are in place, the re-design process must have constraints such that the number of elements and the locations, radii and angles, must be the same. Only the gains are allowed to be altered in the re-design process. Again, the modified genetic algorithm is used in this re-design process. The parameters needed to be adjusted are the feedback resistance of each ring based one the last experiment results. Since the actual locations are not available, we have to make assumption that the radii and angles are the results of the first design. In this re-design process, the fitness function is defined as

$$\begin{aligned} \textit{Fitness} \quad = \quad & \text{Peak sidelobe at 0 degree} + 0.75\,(\text{Peak sidelobe at 15 degrees}) \ldots \\ & + 0.675(\text{Peak sidelobe at 75 degrees}) + 0.9\,(\text{Peak sidelobe at 90 degrees}) \ldots \\ & + 0.675(\text{Peak sidelobe at 105 degrees}) + 0.75(\text{Peak sidelobe at 165 degrees}) \end{aligned} \tag{2.7}$$

where the angle 15 degrees means that the horizontal angle is 15 degrees from the reference line that is the perpendicular line of a highway. The resistance results are 9.86K, 7.04K, 9.90K, 9.90K, 7.49K, 7.80K and 7.00K ohms for ring 1 to 7 respectively. The array gains are shown in Figure 2.26 and 2.27.

2.1.8 Approximate Field Test Beam Pattern

A beam pattern of the circular array is crucial in the array operation. There is no an-echoic chamber that is big enough to test a beam pattern of the array. If the array works properly, its beam pattern should be similar to array gains as shown in Figure 2.26. The beam pattern testing can be done approximately at the sensor location if the starting and finishing points of an acoustic source are approximately fixed and its velocity is approximately constant. The source of acoustic signals is a megaphone having capability to generate sounds. The magnitude response of its signals is shown in Figure 2.28. Human operator held a megaphone and walked with

approximately constant speed passing the sensor in the across-the-lane and in the along-the-lane directions. During the test, the outputs of the sensor were recorded by a VCR and the video images were recorded by a camcorder. The recorded data was processed in the lab in the same procedure presented in the next section. Since all factors are only approximated, the following results are approximate results. The results serve only as a general guide about its beam pattern. The beam pattern results are shown in Figure 2.29 and 2.30. These results are based on the choice of third-order lowpass filter and sixth-order highpass filter having cutoff frequencies at 5400 Hz and 2700 Hz, respectively.

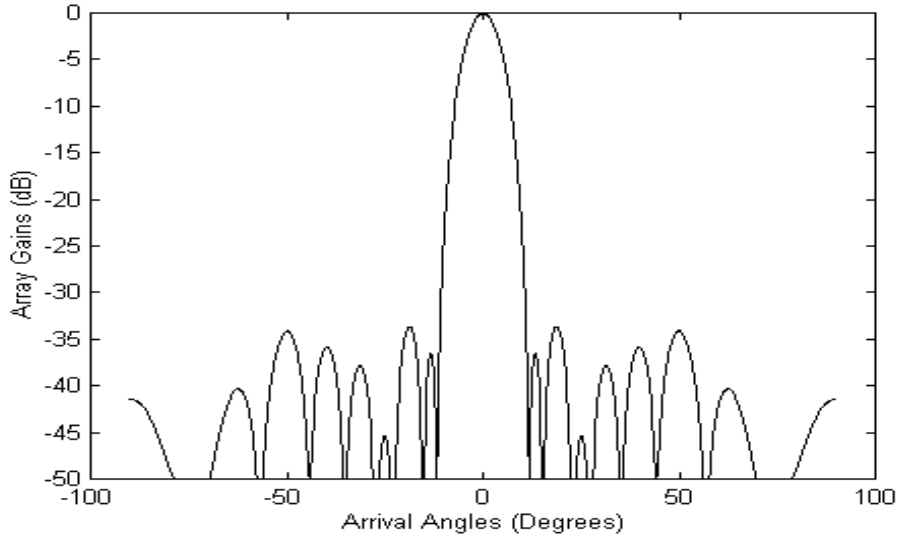

Figure 2.26 Array gains (Across-the-Lane) corresponding to frequency at 9000 Hz.

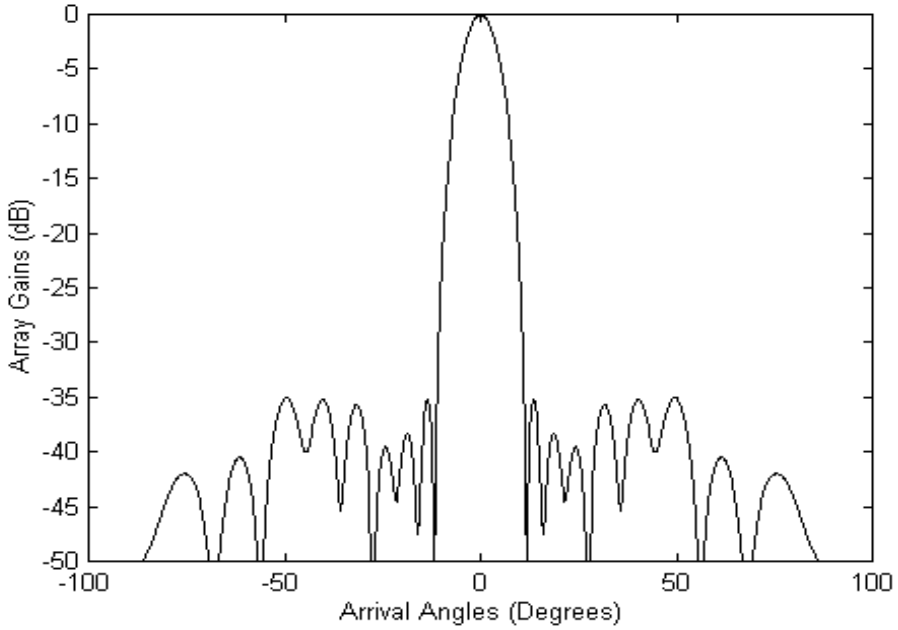

Figure 2.27 Array gains (Along-the-Lane) corresponding to frequency at 9000 Hz.

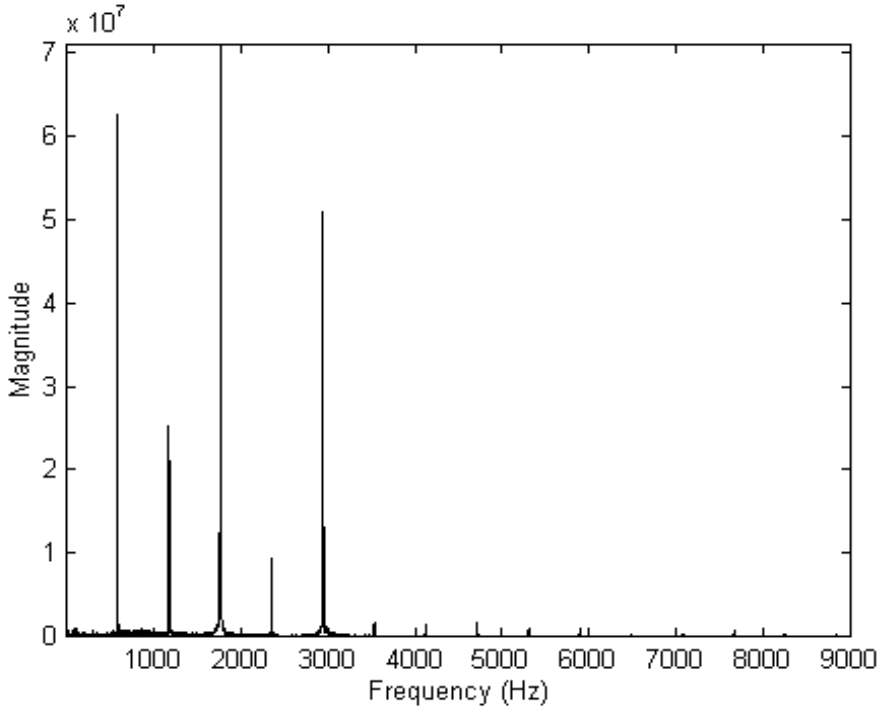

Figure 2.28 Magnitude response of testing signals

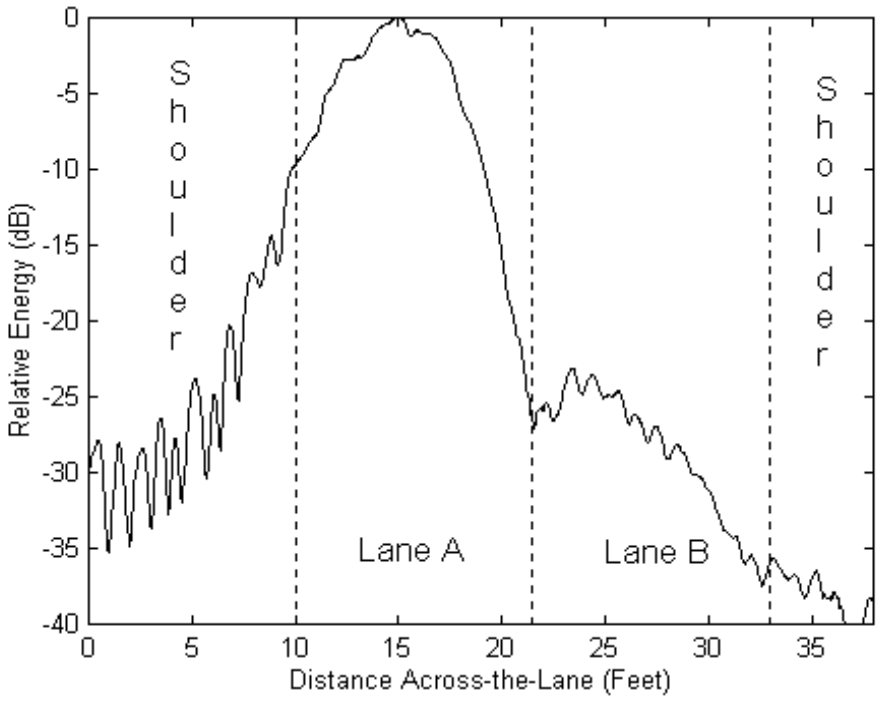

Figure 2.29 Approximated beam pattern in Across-the-Lane direction

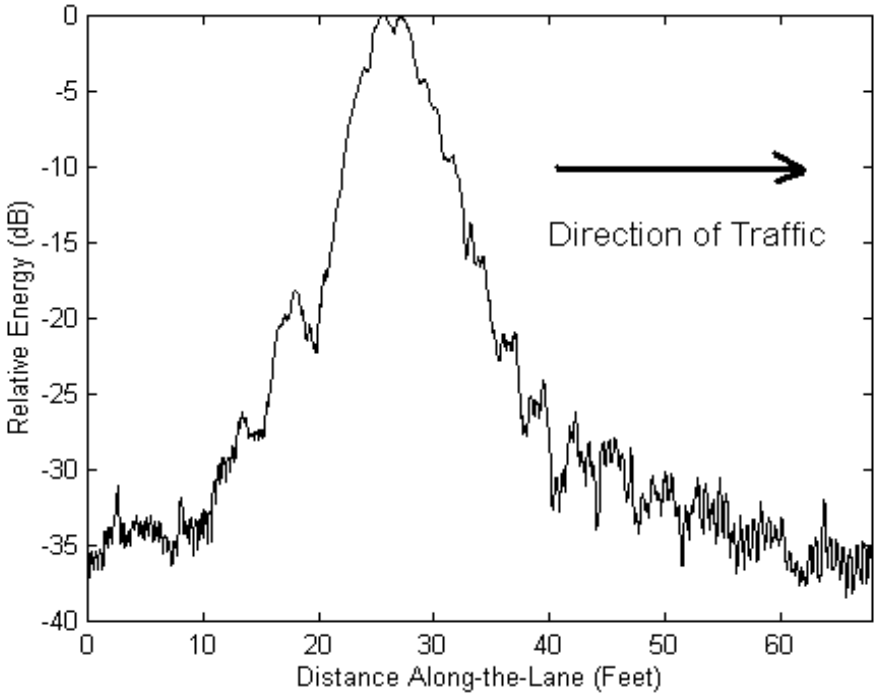

Figure 2.30 Approximated beam pattern in Along-the-Lane direction

Since the sensor was tilted away from adjacent lane in across-the-lane direction and tilted toward traffic in along-the-lane direction, the peaks of the beam patterns were not at the middles. These beam patterns corresponded to the signals from 2700 to 5400 Hz. Even though other frequencies were not completely cut off, they were attenuated or suppressed. In ideal case we want only signals in a desired band of frequencies to pass through the array and signals in other bands of frequencies to be completely excluded. In reality, we can not exclude any signals before hand and we have to select the desired band of frequencies and suppress the undesired frequencies in the later stage. The beam patterns of the array depend heavily on linear operations of all devices. All errors, position errors and gain errors due to the variations in electronic parts, cause the array not to operate up to its design. The choice of operational frequencies must be made to compromise these deficiencies and it is in a frequency band from 2700 Hz to 5400 Hz. It is hard to get all components with identical characteristics. If there is a selection process to select the most similar components from a large number of components based on performance testing, then this error could be minimized. On the other hand, the position errors, angle and radius errors, can also be minimized by using better and more accurate tools to build the array.

2.2 Data Collection

After the prototype sensor was completely built, it was installed at the sensor station on I-460 beside the university campus to collect data. For research purposes, both the acoustic data and the video image of the vehicles are recorded into a video tape simultaneously. The acoustic signals are the outputs of the circular array of microphones while the video image is from a camcorder. In this way, the acoustic patterns are stored effectively and they will be labeled easily and accurately by a human operator later. The roadside data collection process is diagrammed in Figure 2.31.

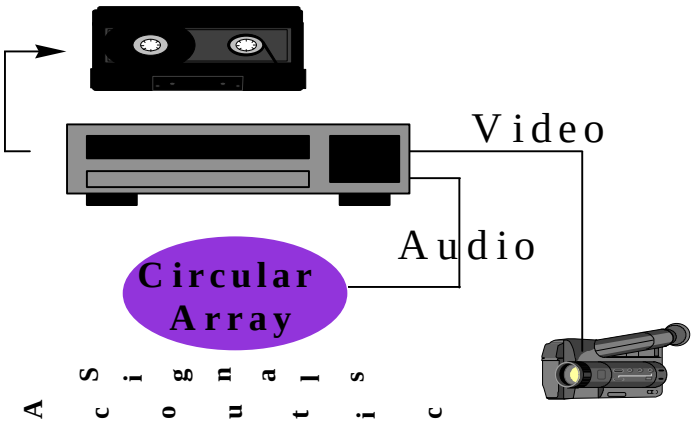

Figure 2.31 Roadside data collection diagram

The recorded data are played back and processed in the lab. The acoustic signals are already transformed to electric signals. These signals are filtered by the second-order lowpass filter with cutoff frequency at 10500 Hz to avoid aliasing before they are sampled. After they are filtered, they are sampled by a sound card with a sampling rate of 44100 Hz and put into a file to be analyzed or processed at a later stage. To avoid unnecessary data storage, data approximately 1.5 seconds before and after the vehicle passes by the sensor are stored. In this way, the data are clearly labeled. Again, this process serves only the research purpose not the implementation part.

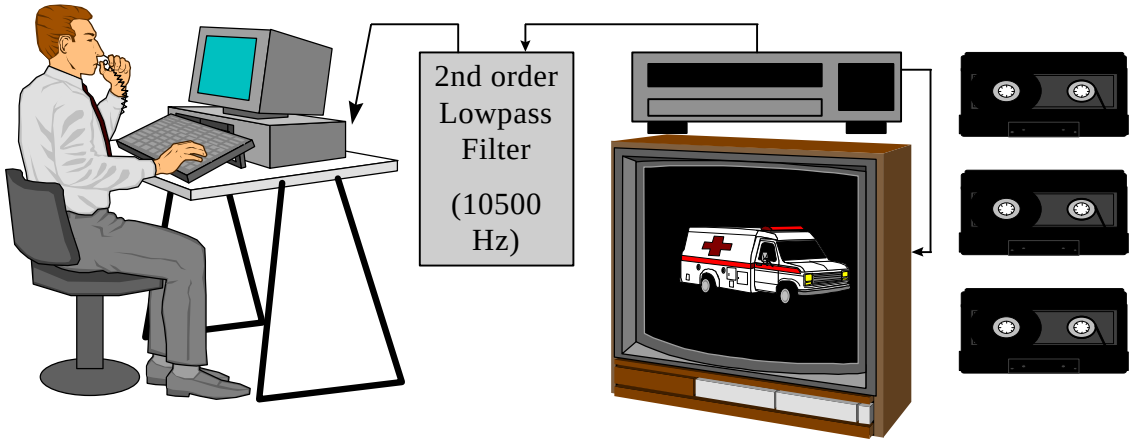

Figure 2.32 Playback process

Chapter 3

Feature Extraction and Analysis

Significant features play an important role in the classification problem. Feature selection is the first step to success or failure of a classification system. A classifier has only feature information about the whole problem. If good features are extracted, then it is easy for a classifier to learn to classify. Although it is a very important stage, most researchers emphasize on the classifier design. Feature extraction depends on the problem at hand; there is no universal rule for feature extraction. Many features can be extracted from data. Humans have an excellent feature analysis and extraction process, but it is hard to duplicate. At this point, feature analysis and extraction still rely on humans to analyze and extract good features from the process.

In the vehicle classification problem, vehicles must be detected first before they can be classified. The detection process is designed to detect the presence of a vehicle and to initialize and end the feature extraction process. The vehicle detection algorithm is developed and presented in section 3.2.

3.1 Preprocessing (Filtering)

Incoming signals are processed one window at a time. A rectangular window is used in this case. The size of the window is 0.02 seconds. Since the sampling rate is 44100 Hz, therefore, there are 882 samples in one window. The signal in this window is filtered by a highpass filter having a cutoff frequency of 2700 Hz: a type I sixth-order Chebyshev filter with 0.1 dB ripple. Then, the filtered signals are filtered again by a lowpass filter having a cutoff frequency of 5400 Hz: a type I third-order Chebyshev filter with 0.1 dB ripple. There are two filters because we want to suppress the signals in lower frequencies more than the higher frequencies.

Using these filters, the circular array ideally forms a beam pattern corresponding to the signals having frequencies from 2700 Hz to 5400 Hz. Even though the sensor is designed for higher frequencies, it still can be used in this lower range of frequencies. Because of the minor lobe invariant property of the circular array, the operation range of frequencies can be varied within the highest designed frequency, 9000 Hz. Since we have frequency dependent characteristic microphones, they introduce minor lobes in the beam pattern. If we operate in a high frequency range, there are more minor lobes than in a lower one (see Figure 2.9 and 2.10). A low frequency bound depends on lane width. If we operate in a very low frequency range, the main beam will cover more than one lane. If we have an ideal circular array, we will have much more freedom to select the operation range.

The energy (E) is calculated for each window. The energy (E) is defined by

$$E = \frac{\sum_{k=1}^{N} s^2(k)}{N} \tag{3.1}$$

where E is the energy in the window, $s(k)$ is the k^{th} sampled signal in that window, and N is the numbers of samples in the window. The current window is designed to overlap 50% with the previous window. The energy envelope of a typical passenger car and a two-axle truck are shown in Figures 3.1 and 3.2, respectively. In Figure 3.1 (a), the original signals sampled with 44.1 kHz are shown. The filtered signals are shown in Figure 3.1 (c). In Figure 3.1 (b), the energy calculated in each window is shown. The average energy (avE) of the energy (E) in the current window and five previous windows is shown in Figure 3.1 (d). The same type of information for a two-axle truck is shown in Figure 3.2. The average energy (avE) is used in the detection task, while the energy (E) in each window is used mostly in the classification task.

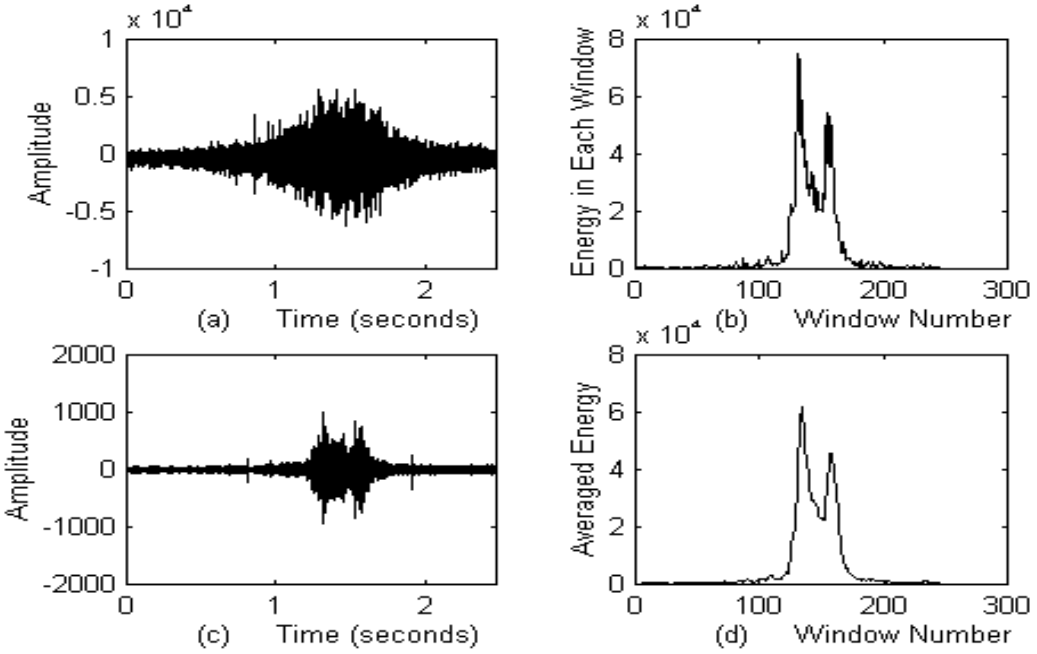

Figure 3.1 Passenger car information (a) the original signals sampled with 44.1 kHz, (b) energy (*E*), (c) filtered signals, (d) average energy (*avE*)

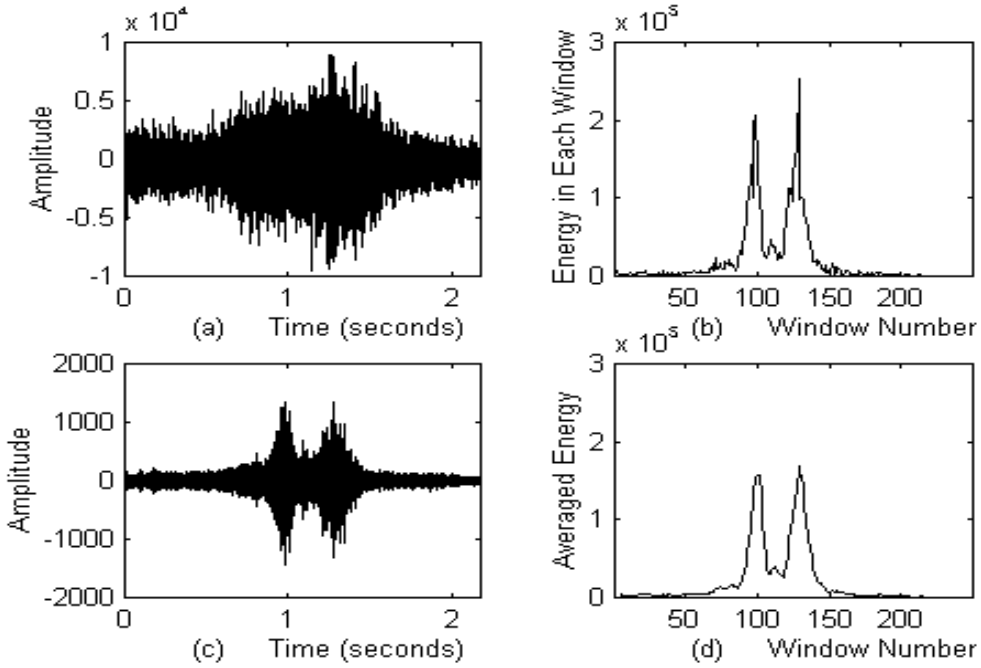

Figure 3.2 Two-axle truck information (a) the original signals sampled with 44.1 kHz, (b) energy (*E*), (c) filtered signals, (d) average energy (*avE*)

Different vehicles have different frequencies; but, even with the same vehicle, there are variations in frequencies in different situations, that is, different speeds, different traffic situations, etc. With different frequencies, the beam patterns are also different. In other words, the energy information even of the same vehicle (see Figure 3.3) is not unique.

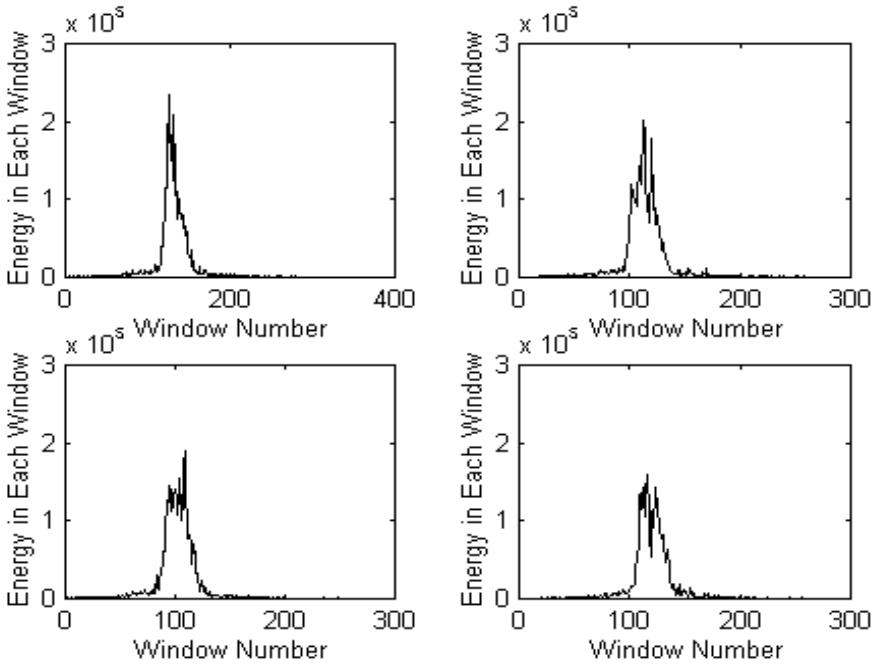

Figure 3.3 Energy patterns of the same vehicle passing the sensor four different times in one hour span (2700 - 5400 Hz)

This test was done within one hour. Even though the driver intentionally drove the vehicle with the same speed, in same manner every time, the energy patterns are different. Because there is no hope to have a unique energy pattern for each class, a prototype matching classifier is not an appropriate choice in this application.

For the classification task, more information is needed. Ideally, the energy resolution method for example, the energies in several frequency bins should be useful information. Since

one concern is real-time operation, energies in several frequency bins may not be appropriate. Besides the energy (E) information in the whole band of frequencies, the upper band of frequencies, 4500 to 5400 Hz, is selected. The filtered signals used in the detection are filtered by a fifth-order highpass filter. The energies (E_2) of this band of frequencies for the same vehicle are shown in Figure 3.4.

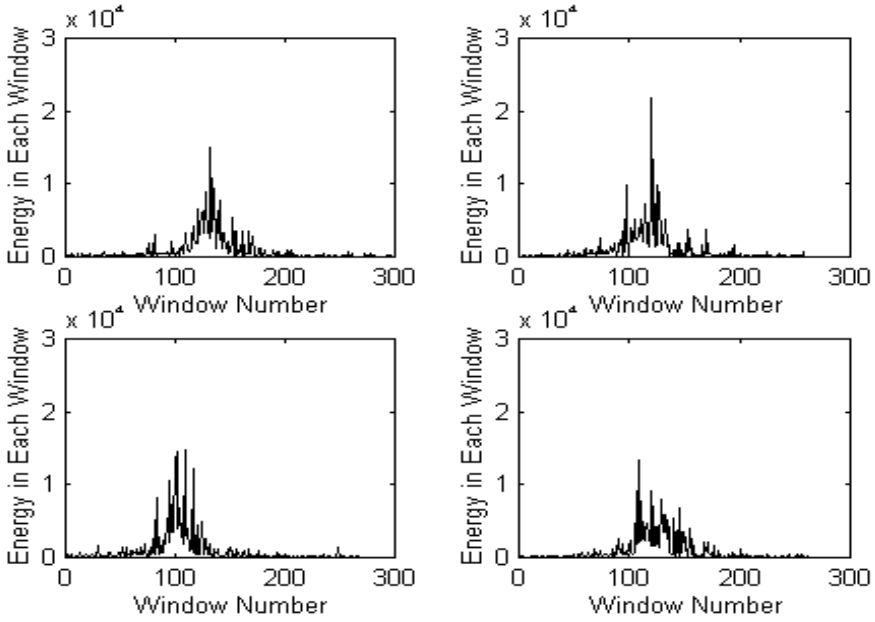

Figure 3.4 Energy patterns of the same vehicle passing the sensor four different times (4500 to 5400 Hz)

Once again, there are variations in the energy patterns. The average energies (avE_2) of this band of frequencies are also calculated and used in the feature extraction process. Features will be extracted from both frequency bands, e.g., from 2700 to 5400 Hz and from 4500 to 5400 Hz. Feature analysis and extraction will be presented in section 3.3.

3.2 Vehicle Detection Algorithm

One requirement of traffic sensors is that they must be able to detect the presence of a passing vehicle. This detection will lead to vehicle count information that is useful in traffic

management and road maintenance. An acoustic sensor such as the IRD SmartSonic sensor is able to count a vehicle's presence [97]. For this research, the detection algorithm is developed not only to detect a vehicle but also to initialize a feature extraction process for vehicle classification.

The detection algorithm must be designed to detect only one vehicle. Although different vehicle classes may have different sizes, the detection algorithm has to overcome this problem and reduce an under count and over count rate. The under count means that the algorithm does not detect the presence of a vehicle. It happens often to a small vehicle such as a passenger car. On the other hand, the over count means that the algorithm detect one vehicle as two vehicles. It happens often to a large vehicle such as a tractor trailer truck. The over count affects not only on the vehicle count information but also the classification performance. In the over count situation two different sets of features are extracted for one vehicle. These two sets of features may not be the same as one set of features and the classifier may not classify them correctly. In the under count situation the vehicle is definitely not classified.

In this study, the detection algorithm is designed to operate under normal highway traffic operations. The nominal speed is approximately 55 mile-per-hour. The detection and feature extraction process may be put into diagram as shown in Figure 3.5.

There are three tests in the designed detection algorithm (see Figure 3.5). All tests are designed to overcome under and over count problems. For the first test three requirements must be met. First, the average acoustic energy (*avE*) must be above a preset threshold. This preset threshold is set such that most of average energy (*avE*) of small vehicles in adjacent lane is below this threshold. Second, the average acoustic energy (*avE*) must be rising. The final requirement is that the operation mode must be in reset mode. The reset mode is the mode that the sensor is ready to detect a new vehicle.

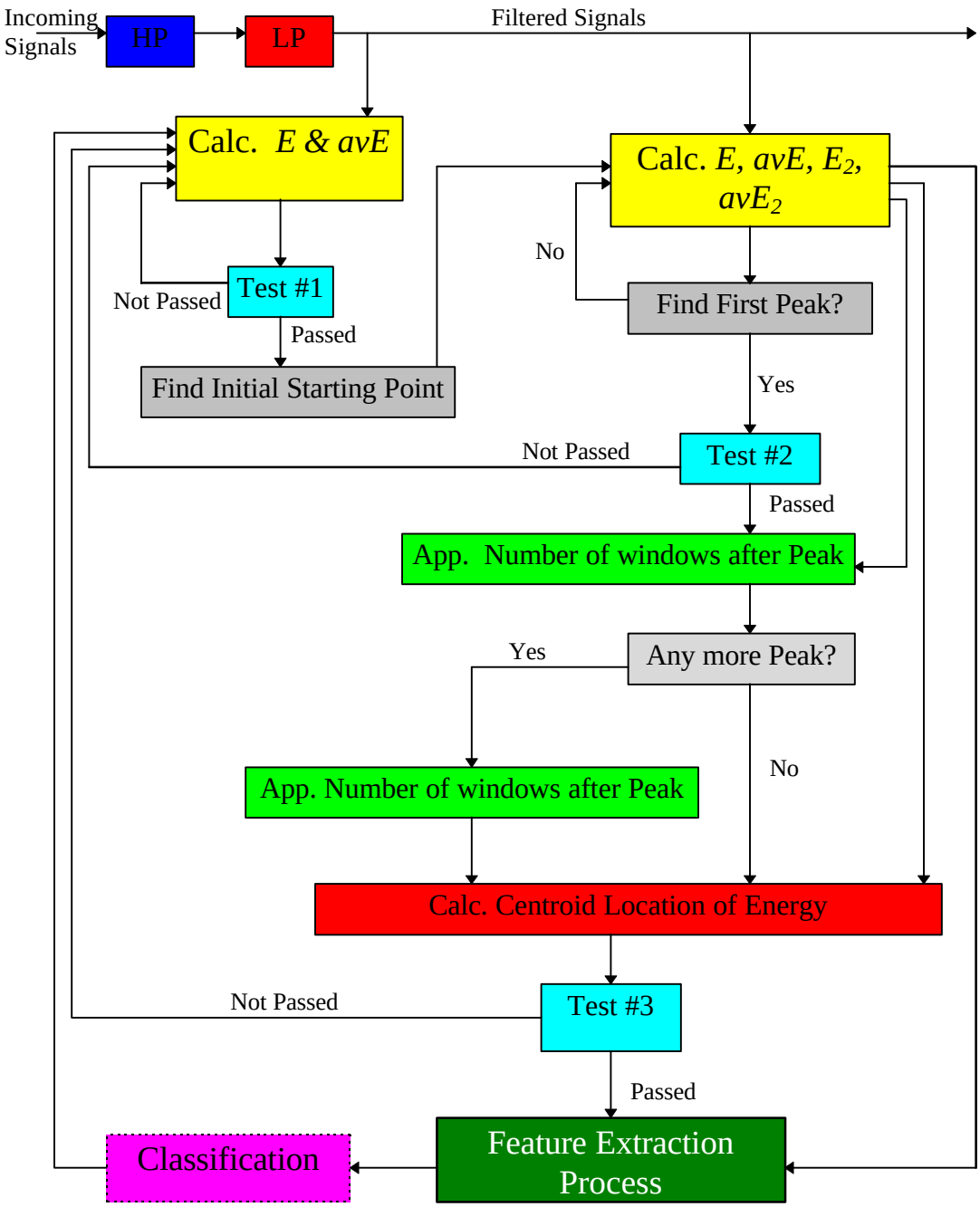

Figure 3.5 Detection and feature extraction diagram

After the first test is passed, the feature extraction process begins. The initial starting extraction point is determined by using the average energy (*avE*) information. The initial starting point should be at the point the average energy (*avE*) is across the threshold. All necessary information used in feature extraction such as the energy (avE_2) in the upper band must be initially gathered at this point. Then, the process gathers information when at least 0.1 seconds after the initial starting point have passed and when at least one of following conditions is met: the first condition is that the average energy (*avE*) is below 125% of the previously preset threshold, the second condition is that the average energy (*avE*) is decreasing, the third condition is that the current window is more than 0.3 seconds away from the found peak. These conditions are set to make sure that the peak is found and there are few numbers of windows from the starting point. The indication of the energy movement indicating the energy rising or decreasing is calculated by the following equation:

$$Index = \frac{1}{N}\sum_{n=1}^{N}\frac{avE(k-N+n)}{avE(k-N+n-1)} \tag{3.1}$$

where avE = the average energy

k = the current window

N = the number of window used, $N = 8$ in this case.

When the average energy goes up, this index will also go up. A few movements of averaged energy will not cause this index to go up much.

Once the first peak is found, the location of the peak, the peak value and the number windows above the preset threshold from the initial point are used in the second test. If all following conditions are met, then the second test is passed. If any of them is not met, then the second test is failed and the operation is placed in the reset mode again. The first condition is that the location of the found peak is not close to the centroid of energies of the last detected vehicle. The number of windows used in this case is calculated by how far the current peak from the centroid of energies of the last detected vehicle. This threshold value might come after the

previous vehicle is classified. If the previously classified vehicle is big and very loud, then this threshold should be big. On the other hand, if the previously classified vehicle is small, then this threshold value should be small. This approach is to avoid counting one big truck as two small vehicles or two small vehicles as one big truck. The second condition is that the peak value is above 125% of the previous preset threshold. The final condition is that the number of windows above the preset threshold from the starting point is greater than N (N = 8 in this case).

After the second test is passed, the approximation for the number of windows following the peak is determined by the fuzzy logic systems. The detected vehicle is classified into 3 broad classes, small, medium and large depending on information of how far is the peak from the initial starting point and the peak value. If the detected vehicle is classified as a small vehicle, the number of windows after the peak is small. If the detected vehicle is classified as a medium vehicle, the number of windows after the peak should be medium. If the detected vehicle is classified as a big vehicle, the number of windows after the peak should be large. At this point, a fuzzy logic system is used as the number of windows approximator. The linguistic variables are the number of windows from the initial starting point to the peak and the peak value. The following fuzzy If-Then rules as shown in Table 3.1 are used.

		Peak Value (P)		
		Small	Medium	Large
Number of Windows From starting Point to the Peak (Nwd)	Small	Small	Small	Medium
	Medium	Small	Medium	Large
	Large	Medium	Large	Large

Table 3.1 Fuzzy rule base for number of windows after the peak approximation

For example, the first rule is : If **P** is **Small** and **Nwd** is **Small** Then **Nwd-After** is **Small**.

The membership function of Number of Windows From starting Point to the Peak (Nwd), the Peak Value (P) and the Number of Windows After the Peak (Nwd-After) are shown in Figures 3.6, 3.7 and 3.8, respectively.

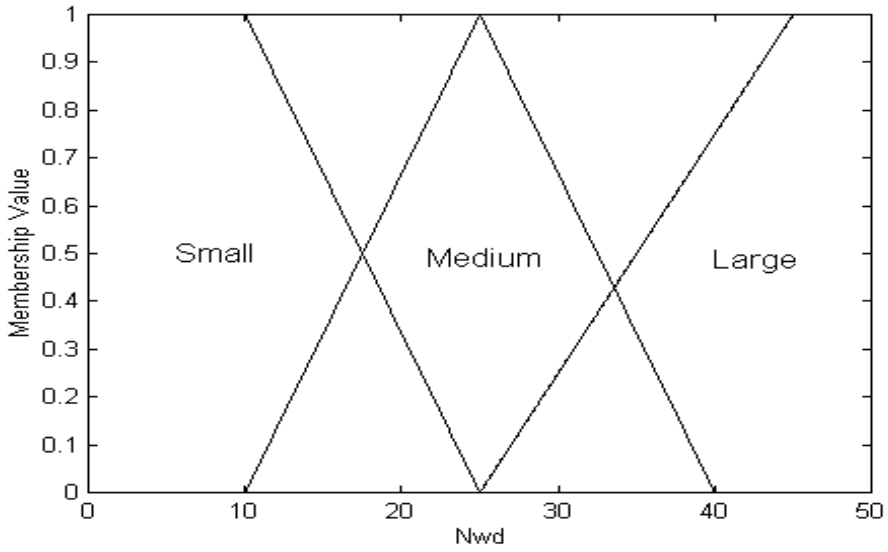

Figure 3.6 Fuzzy membership functions of number of windows from starting point to the peak (Nwd)

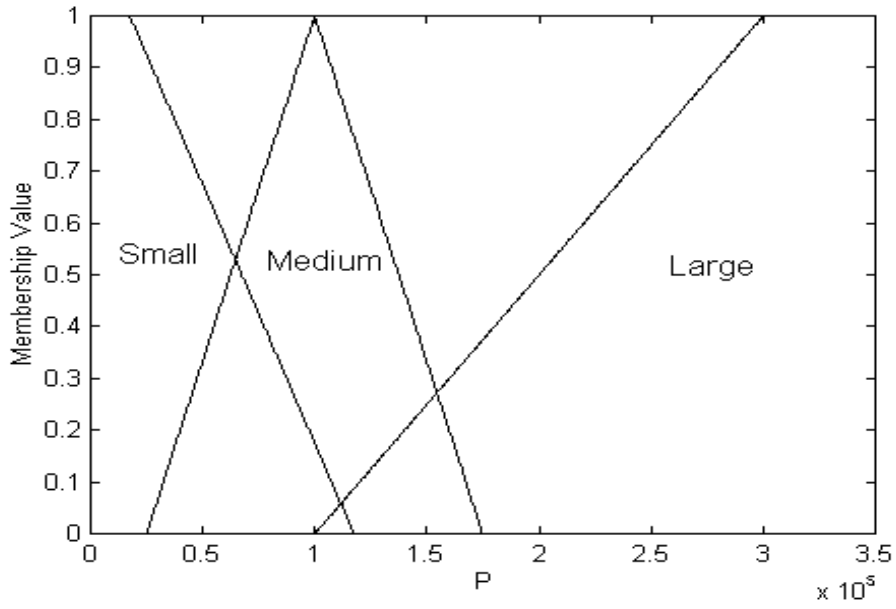

Figure 3.7 Fuzzy membership functions of the peak value (P)

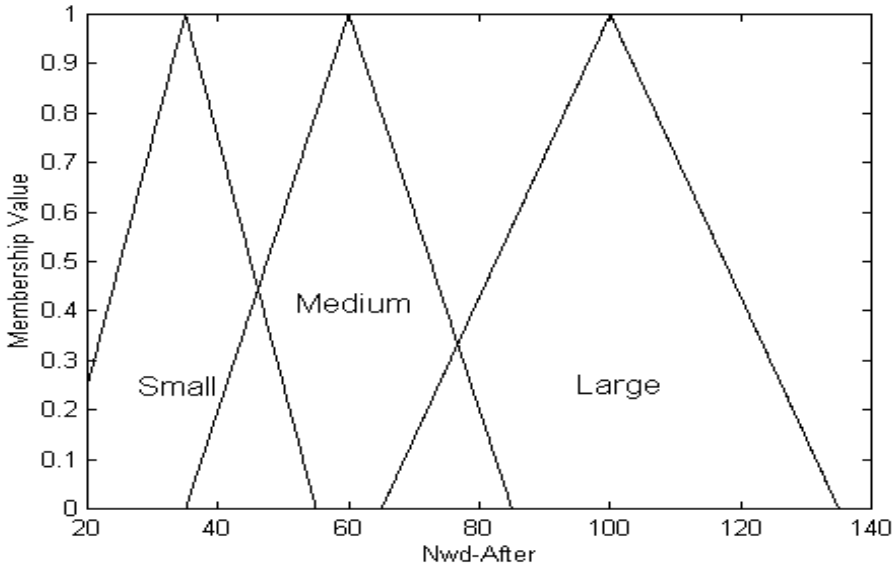

Figure 3.8 Fuzzy membership functions of Nwd-After

This fuzzy logic system uses the singleton-fuzzifier, product inference, and balance of area defuzzifier (BOA). More details about this system are presented in Chapter 4. Once the number of windows following the peak is determined, a new peak and its location are determined again from the initial starting point to the current point. If the new and old peaks are the same, then go to the next process, centroid location of energy calculation. If the new peak is different from the old one, then, the number of windows after the peak (Nwd-After) is approximated again by using the same fuzzy logic method. After this process is done, the centroid location of energy is calculated by the following equation:

$$L = \frac{\sum_{k=1}^{N} k \cdot E(k)}{\sum_{k=1}^{N} E(k)} \tag{3.2}$$

where L = Approximated location of the centroid of energy from starting point to the current position.

N = the number of windows from the starting point to the current one.

$E(k)$ = the energy of the k^{th} window.

This location serves as the focal point of the detected vehicle's energies.

In the third test, the centroid location is used. If this location is far from the centroid of energies of the last detected vehicle by more than 112.5% of the number of windows after the peak (Nwd-After) of the last detected vehicle, then the third test is passed. If this condition is not met, then the third test is failed. If the test is failed, then, the operation mode is in the reset mode and the detected signals are determined as parts of the last detected vehicle. If the third test is passed, then the feature extraction process begins. First, the starting and ending points are determined. The starting point is set at the point that is before the centroid point with the fixed number of windows. In this case, there are 36 windows before this centroid point for every type of vehicle. The ending point is the point that is far from the centroid point by the same number of windows as the number of windows after peak (Nwd-After). In this way, a different vehicle will have a different ending point, depending on its peak value and how far the peak is from the starting point. Feature extraction will be presented in the next section. Once the features are extracted, then the vehicle can be classified.

Once the classifier is added to the system, the overall system will be improved, especially the number of windows away from the current centroid of energy. As of today, this detection algorithm categorizes the types of vehicles as small, medium and large from only partial information. It serves only as an automatic tool in features' preparation and extraction to be used in training and testing the classifier.

3.3 Feature Analysis and Extraction

There is a need to extract only significant features from the sampled acoustic signals. Humans have an excellent feature extraction capabilities. We can extract only good and necessary features from the patterns of interest. It is a straight forward process for us, but it is hard to duplicate. There are no general methods in feature extraction for general problems, since the

process is problem dependent. There also are an unlimited number of features that can be considered, some important, some not. Thus, the feature extraction process requires knowledge and capability of the designer to be involved in the process.

3.2.1 Feature Extraction

There are numerous features that can be extracted. At this time, only features involving the energy envelope in time domain are considered.

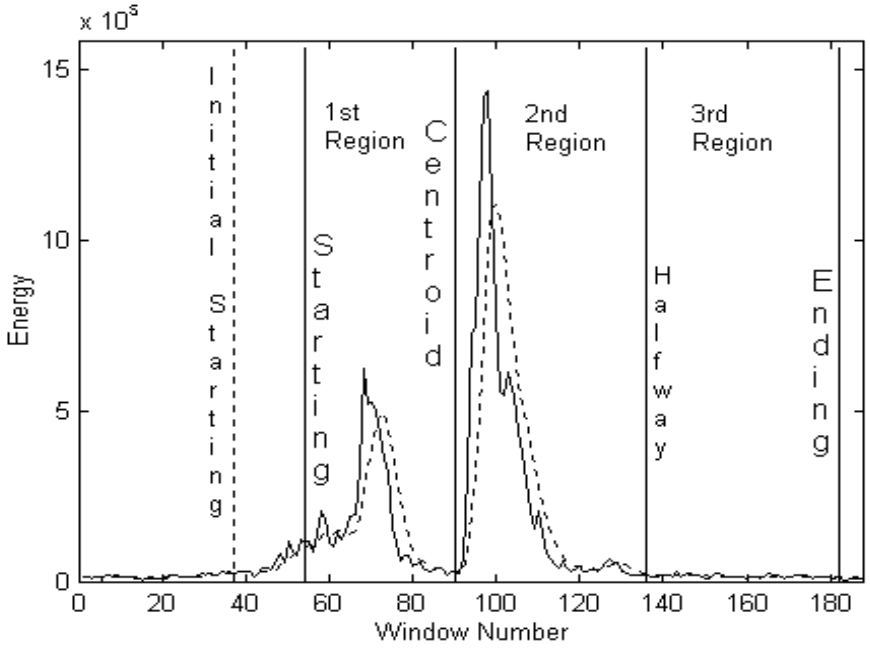

Figure 3.9 Feature extraction regions

There are 30 features extracted as shown in Table 3.2. Figure 3.9 shows an example of feature extraction along with the detection process. The "initial starting" shows the beginning of process after the first test is passed. The "centroid" is the centriod location calculated using Equation (3.2). The "starting" line is the beginning of the region of windows from which the features are extracted. The "ending" line is the end of the whole region. Besides the windows, there are three small regions: 1st, 2nd and 3rd region to extract features. The 1st region is defined as

Table 3.2 Feature descriptions

Feature Number	Feature Description
1	Maximum value of averaged energy (*avE*)
2	Approximated Location of centroid of energy from starting point
3	Location of the peak value of *avE* from the starting point
4	Difference in centroid location and the peak location
5	Maximum value of energy (*E*,) in each window
6	Number of windows from starting point to ending point
7	Approximated Location of maximum value of *E/avE* in 1st region
8	Approximated Location of centroid of energy in 1st region
9	Mean of energy in 1st region
10	Approximated Location of maximum value of *E/avE* in 2nd region
11	Approximated Location of centroid of energy in 2nd region
12	Mean of energy in 2nd region
13	Approximated Location of maximum value of *E/avE* in 3rd region
14	Approximated Location of centroid of energy in 3rd region
15	Mean of energy in 3rd region
16	Number of windows having $E/avE > 1$
17	Number of windows having *avE* > 50% of maximum *avE*
18	Number of windows having *avE* > 25% of maximum *avE*
19	Maximum value of average energy, *avE* in 3rd region
20	Approximated Location of maximum value of E_2/avE_2 in 1st region
21	Approximated Location of maximum value of E_2/avE_2 in 2nd region
22	Approximated Location of maximum value of E_2/avE_2 in 3rd region
23	Number of windows having $E_2/avE_2 > 1$
24	Maximum value of avE_2
25	Sum of number of zeros crossing for each window in 1st region
26	Sum of number of zeros crossing for each window in 2nd region
27	Sum of number of zeros crossing for each window in 3rd region
28	Number of windows having avE_2 > 50% of maximum avE_2
29	Number of windows having avE_2 > 25% of maximum avE_2
30	Number of windows after peak approximated by FLS

the region from the starting line to the centroid line. The 2nd region is defined as the region from the centroid line to halfway between the centroid and the ending line. The 3rd region is defined as the region from the halfway line to the ending line.

All features are derived heuristically. Features number 7, 8, 9, 10, 11, 12, 13, 14, 15, 19, 20, 21, 22, 25, 26 and 27 are features extracted from each region. The others are from the whole region. The ratio between the energy (*E*) in each window and the averaged energy (*avE*) will indicate the fluctuation of energy. The approximate location of the these fluctuations in each region indicates where is the most fluctuation in that region. The maximum value of energy indicates how loud it is. Most trucks are loud, but some passenger cars are also loud. Some vehicles having a box shape have multiple peaks of energy, for example, the vehicle energy shown in Figure 3.10 is a vehicle in FHWA class 5. Energies of FHWA class 9, single trailer truck, also have multiple peaks as shown in Figure 3.11. Unlike a single unit truck (FHWA class 5), trailer trucks have a second peak far from the first peak.

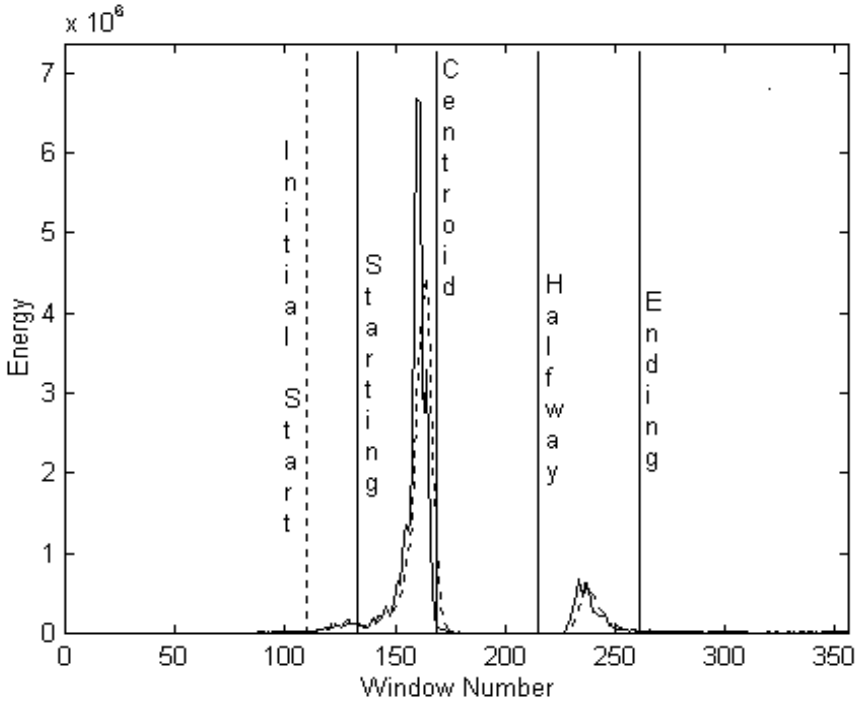

Figure 3.10 Feature extraction regions of tractor trailer truck (Class 9, FHWA)

The features involving the amplitude of energy, e.g., feature number 5, are transformed into log scale. Then, all features are linearly transformed into a range from 0 to 2 by using the following equation :

$$X_n = \frac{(Mx_n - Mn_n)(X_o - Mn_o)}{(Mx_o - Mn_o)} + Mn_n \qquad (3.3)$$

where

Mx_o : maximum value of the old range,

Mn_o : minimum value of the old range,

Mx_n : maximum value of the new range,

Mn_n : minimum value of the new range,

X_o : the old value in the old range,

X_n : the new value in the new range.

In this way, the inputs of the classifiers are in the range of 0 to 2. This normalization is part of the data preparation for neural networks or fuzzy logic system classifiers introduced in the next chapter.

3.2.2 Principal Component Analysis

In classification problems it is important to extract significant features. As mentioned earlier, the success or failure of a classification system depends heavily on these features. The feature extraction process might be viewed as a transformation from pattern space to feature space. In an ideal case there is no loss of information in this transformation. The transformation is designed only to extract significant features and reduce the dimension of the problem. Although the designed feature extraction might already reduce the dimension of the problem, there might be a need to further reduce these features. Principal component analysis (PCA) is an approach to do this dimension reduction effectively. For more detail's analysis of PCA, please consult [38].

After all desired features of the data are extracted and transformed into a 0 to 2 region, PCA is applied to these data. First, the correlation matrix, *R*, is calculated by:

$$R = \frac{1}{n}\left(XX^T\right) \tag{3.4}$$

where R = correlation matrix

X = original feature matrix, m x n

n = total number of data

m = dimension of the old feature vector

Then, find the eigenvalues and eigenvectors of *R* and order its eigenvalues in decreasing order with their corresponding eigenvectors:

$$\lambda_0 > \lambda_1 > \ldots > \lambda_{m-1} \tag{3.5}$$

and

$$\left[\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{m-1}\right] = \mathbf{U} \tag{3.6}$$

The matrix **U** is normalized such that each column has norm 1. The final dimension can be determined from the eigenvalues. Normally, the eigenvectors corresponding to eigenvalues large enough are kept. The transformed feature matrix is calculated by

$$\mathbf{X}_{new} = \mathbf{V}^T\mathbf{X}_{old} \tag{3.7}$$

where $\mathbf{X}_{new}$ = the new feature matrix, p x n

$\mathbf{V}$ = the transform matrix that keeps p columns of $\mathbf{U}$

$\mathbf{X}_{old}$ = the old feature matrix, m x n

In this way, the only significant features are kept and the redundant features are discarded without loss of valuable information. In this research the features are reduced from 30 to 24 features. After this transformation, normalization might be needed to linearly transform to a suitable range for neural networks and fuzzy logic system classifiers.

3.3 Training and Testing Data

Training and testing data used in this study are collected from the roadside sensor station and processed using the described detection and feature extraction. The whole data consists of five classes: FHWA class 2, class 3, class 5, class 6 and class 9. The other classes will not be included because the number of data is not sufficient to train or test the classifiers. From this point, the data are classified into 5 classes. Class number 1 is the FHWA class 2, passenger cars. Class number 2 is the FHWA class 3, pickups, vans and mini-vans. Class number 3 is the FHWA class 5 that includes all single unit trucks having two axles, six tires. They are small trucks, small trucks with flatbeds, small trucks with flatbeds including loads, trash trucks, heavy duty trucks, trucks with box, etc. Class number 4 is the FHWA class 6 that includes all single unit trucks having three axles, ten tires. They are heavy duty trucks, trash trucks, cement trucks, trucks with box, etc. The last class, class number 5, is the FHWA class 9 that includes all five-axle single trailer trucks that include tractor trailer trucks, gas trucks, flatbeds, and flatbeds with load. Class number 1 has the most dominant number in the whole data and class number 4, the fewest. Because of the weather during the data collecting period, there are few numbers in class 4. These data were collected in various conditions, from cold to warm weather, from cloudy to clear sky and from early morning to late afternoon. The road surfaces are from dry to light wet surface. Data during snow and raining are excluded. All data are collected under normal highway traffic operations where the nominal speed is 55 miles per hour.

Chapter 4

Classifier Designs

The design and analysis of classifier networks are the most emphasized part of this research. Existing methods are examined and used in comparison to new proposed approaches. There are two major paradigms studied: multilayer perceptrons (MLP) and adaptive fuzzy logic systems (AFLS). The pattern classification problem is an approximation problem, exemplified by the vehicle acoustic classification. Even within the same class, there are many variations in shape, size, model, engine, manufacturer, etc. Among different classes some classes are very similar, such as automobile and van classes. For example, a mini-van and automobile are very similar in size, engine, and number of axles. They may differ only in their shapes. There is a need to have a powerful classifier to solve this classification problem besides good feature extraction methods. Both network types have been proved to be universal approximators [38][59][105]. Since there are approximations in nearly every stage of a pattern classification system, a good approximator is certainly a help to solving the problem.

4.1 Pattern Classification with Discriminant Functions

Pattern classification may be viewed as an information mapping or information labeling process. The pattern must be classified as one of the pre-specified classes or as none of them. This is a "hard" decision in which the object is in the class, or not. Even though it is reasonable to assign a fuzzy membership value *between* 0 and 1 to the objects rather than *either* a 1 or 0 membership value, but, ultimately, we still categorize them into certain classes. In other words, the information must be further reduced until the end result, classification. Those fuzzy class membership values may be used to indicate a confidence level in the classification results. For example, an object is classified as class A with "high" confidence if its membership value in class A is much higher than other classes.

The pattern classification system classifies a feature vector **x** that is in $\boldsymbol{R}^n$ according to the decision rule: **x** is in the i^{th} class if and only if $g_i(\mathbf{x})$ is greater than $g_j(\mathbf{x})$ for all j not equal to i, where g_i is a discriminant function mapping $\boldsymbol{R}^n$ to $\boldsymbol{R}$ [25][57].

For example, in a Bayesian classifier, the Bayes decision rule is defined as

$$\mathbf{x} \in w_i \qquad \text{if } p(w_i|\mathbf{x}) > p(w_j|\mathbf{x}) \qquad \text{for all } j \neq i \tag{4.1}$$

Since the posterior probabilities, $p(w_i|\mathbf{x})$ and $p(w_j|\mathbf{x})$, cannot be calculated directly, the decision rule is reduced to

$$\mathbf{x} \in w_i \qquad \text{if } p(\mathbf{x}|w_i)p(w_i) > p(\mathbf{x}|w_j)p(w_j) \qquad \text{for all } j \neq i \tag{4.2}$$

Therefore, the discriminant function is

$$g_i(\mathbf{x}) = p(\mathbf{x}|w_i)p(w_i) \tag{4.3}$$

In the linear classifier, the linear discriminant function is defined as

$$g_i(\mathbf{x}) = w^T\mathbf{x} + w_{i0} \tag{4.4}$$

where w^T is a weight vector and w_{i0} is the threshold weight.

In this way, a classifier may be viewed as a discriminant calculator. All classifiers studied in this research may be viewed as a discriminant calculator or a function mapping from the feature space to the decision space.

4.2 *K*-Nearest Neighbor (*k*-NN)

Statistical pattern recognition has been used in pattern recognition longer than any other approaches. The ideal optimal classifier, Bayesian classifier, is one of these statistical approaches. It is considered to be an optimal classifier since no other classifiers can achieve a higher correct classification rate than this classifier. Bayesian classifiers require all relevant probabilities to be known. In practice, these probabilities are rarely, if ever known; and, they must be approximated from a limited number of available samples. The probability distributions are typically assumed to

be a certain form, usually, a normal distribution. The results are suboptimal if all relevant probabilities are not completely correct which is always the case. We may say that there is no realizable optimal classifier in practice. This is the main reason why there are so many approaches in the pattern recognition field. All existing approaches give good and acceptable results that strongly depend upon the applications. Since a statistical approach will be used only as a comparison method, there is too much risk in approximating the relevant probability distributions. Consequently, a non-parametric statistical approach, *k-nearest neighbor*, will be used to avoid bad assumptions in probability distributions. In this way we let the data speak for itself, making no critical assumptions.

The *k*-nearest neighbor (*k*-NN) method estimates the posterior probabilities [25][68][86][92]. Although the *k*-NN directly estimates the posterior probabilities, it still requires and uses directly a priori information as the training data set. Suppose we have P classes to consider and that there are n training data, n_1, ..., n_p from each class. Suppose we want to classify the input $\mathbf{x}$ by taking k nearest observations and the number observations from class w_1, ...,w_p are $k_1(\mathbf{x})$, ..., $k_p(\mathbf{x})$, respectively. From the hypersphere surrounding $\mathbf{x}$ having the volume $v(\mathbf{x})$, the joint density function of $\mathbf{x}$ and w_i, $p(\mathbf{x}, w_i)$, may be approximated by $(k_i(\mathbf{x})/n)/v(\mathbf{x})$. From Bayes rule,

$$p(\mathbf{x}, w_i) = p(w_i \mid \mathbf{x})p(\mathbf{x}) \tag{4.5}$$

In this way, the posterior probability will be

$$p(w_i|\mathbf{x}) = \frac{p(\mathbf{x},w_i)}{\sum_{j=1}^{P} p(\mathbf{x},w_j)} \tag{4.6}$$

$$p(w_i|\mathbf{x}) = \frac{(k_i(\mathbf{x}) / n)v(\mathbf{x})}{\sum_{j=1}^{P}(k_j(\mathbf{x}) / n)v(\mathbf{x})} \tag{4.7}$$

Thus, the estimate of the posterior probability, $p(w_i \mid \mathbf{x})$, is $k_i(\mathbf{x})/k$. This leads immediately to the decision rule (3.1) : classify $\mathbf{x}$ as belonging to the i^{th} class if $k_i = \max_p(k_p)$. In other words, we

select the class most frequently observed with the largest population within the region centered at **x**. This is well known as the *k*-nearest neighbor (*k*-NN) classification rule.

A suitable metric used to measure the closeness or similarity is crucial in this *k*-NN classifier. If the measurements are measured on different scales, normalization is required. Data are usually normalized by their standard deviations or the ranges of each variable. There are several distance measures. A distance measure must obey three metric axioms:

$$d(\mathbf{x}, \mathbf{y}) = 0 \qquad \text{if and only if } \mathbf{x} = \mathbf{y} \tag{4.8}$$

$$d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x}) \tag{4.9}$$

$$d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{x}, \mathbf{z}) \qquad \text{for all } \mathbf{z} \text{ in } R^n \tag{4.10}$$

The most popular distance measure is Euclidean distance or l^2 norm:

$$d(\mathbf{x},\mathbf{y}) = l^2(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{4.11}$$

The l^p distance measure is another distance measure that can be used. It is defined as:

$$d(\mathbf{x},\mathbf{y}) = l^p(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{1/p} \qquad \text{for } p \geq 1 \tag{4.12}$$

The Euclidean distance measure is a special case of the l^p norm distance measure when $l = 2$. Another commonly used distance measure is the Mahalanobis distance:

$$d(\mathbf{x}, \mathbf{y}) = (\mathbf{x}, \mathbf{y})\mathbf{K}^{-1}(\mathbf{x}, \mathbf{y})^T \tag{4.13}$$

where **K** is the positive-definite covariance matrix of the random vector **x** that is $E[(\mathbf{x} - \mathbf{m}_x)(\mathbf{x} - \mathbf{m}_x)^T]$ and $\mathbf{m}_x$ is the mean vector of **x**, $E[\mathbf{x}]$. This Mahalanobis distance measurement closely relates to the natural logarithm of the Gaussian probability density function, therefore, it is particularly useful for classifying Gaussian-distributed patterns.

Another variable affecting the performance of the *k*-NN classifier is the number of nearest neighbors, *k*. The choice of *k* can be made by cross-validation methods, in which the training data are divided into two sets. The first set is used as a training database and the second set is

classified using a *k*-NN rule. The choice of *k* is the one having the highest classification performance.

A tie-breaking rule must be used if there are more than two classes. In two-class problems, *k* is chosen to be an odd number to avoid 'ties'. In more than two class problems, the tie-breaking rule is normally used. The assigned class of **x** may be the class that is closest to **x** among the tied classes. In this research tie-breaking is done by reducing *k* sequentially until the winner is determined. If *k* is reduced to 1, it becomes *1*-NN rule and the previous tie-breaking rule is automatically applied. This *k*-NN classifier will be used as a "baseline" comparison method in this research.

4.3 Multilayer Perceptron (MLP)

4.3.1 Overview

The multilayer perceptron (MLP) is the most widely used paradigm among neural networks in various applications. It is biologically inspired, that is, it is designed to have a similar structure and behavior to a biological neural system. Even if it is far from the real biological neural system, it is a powerful signal processing algorithm for solving many practical problems. It is known to be a universal approximator in which it can approximate any function to any degree of accuracy given enough nodes in hidden layer. This capability makes the MLP a powerful classifier in pattern classification problem. It can learn from examples through training. Besides its ability to approximate, MLPs have a good generalization. After successful training, it can give good results for unseen input data within the same input space.

The attractive attributes of neural networks, including MLPs, may be summarized as:

- Capability to learn information from examples.
- Ability to generalize to new unseen inputs.

- Robustness to noisy data.
- High fault tolerance. Failures of a few neurons do not greatly affect the overall system performance.
- Inherent parallelism in the network architecture. This leads to fast hardware implementation for real-time applications.

Since the pattern classification problem involves approximations in almost every stage, the MLP is certainly a good candidate for a classifier in a pattern classification system. The MLP classifier can be viewed as a discriminant calculator. Its generalization property will provide discriminant values that are used in the decision process.

4.3.2 Artificial Neuron

The basic unit in the biological brain is the neuron while the basic unit in a MLP is called an artificial neuron. The most commonly used neuron model is shown in Figure 4.1. It was first introduced by McCulloch and Pitts in 1940s.

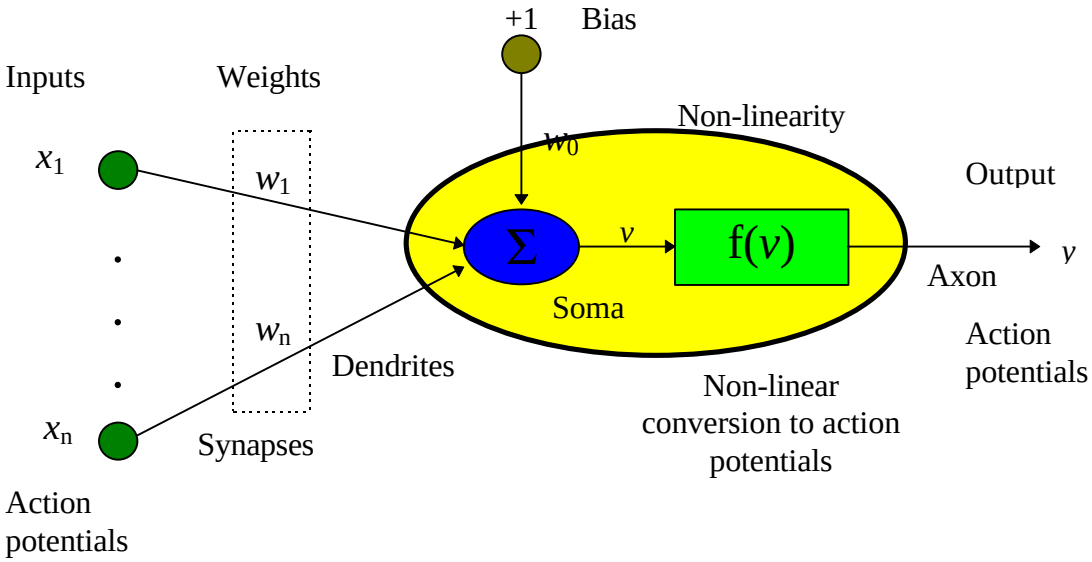

Figure 4.1 Artificial neuron (McCulloch-Pitts neuron model)

In a biological neuron, the numerous synapses receive action potentials from other neurons and convert these signals to voltage potentials. These modified versions of signals are transmitted to the soma through dendrites. The soma performs a spatio-temporal weighted aggregation (summation) of all these inputs and converts this weighted aggregation signal into an action potential as an output of the neuron if it greater than its threshold value. These action potentials are transmitted through the axon to other neurons for further processing. Additional information about biological neuron can be obtained from [36][48][49].

On the other hand, in an artificial neuron, the input signals are real or binary numbers instead of pulse signals, so there is no need for demodulation. The inputs are weighted by "synaptic" weights. There is also followed by a summation of these weighted inputs. This summed signal is added to a bias term. The resulted signal is then transformed by a non-linear "activation" function yielding the output signal. This model is a crude representation of biological neuron. The whole operation may be put into a mathematical representation as follows. The output, y, of the neuron is defined as

$$y = f(v) \tag{4.14}$$

and

$$v = \sum_{i=0}^{n} w_i x_i \tag{4.15}$$

where x_i = the i^{th} input ($x_0 = 1$).

w_i = the weight connecting to the i^{th} input. It is a bias if i is equal to 0.

$f(\ )$ is the non-linear activation function.

This operation can be viewed as a non-linear transformation from R^n to a scalar, R, output. This operation may be divided into two parts, similarity measurement and non-linear activation. The similarity measurement can be done in two methods, inner or scalar product and distance measurement. Only the inner or scalar product will be used in this research. The sum of the weighted inputs is actually a scalar product between the input vector and weight vector, a linear transformation from R^n to R. This operation gives a maximum value if both of them are colinear.

The past experiences are stored in the weights through training. This operation measures the similarity of the current inputs with the past experiences (weights). This similarity measurement along with the activation function will recall the stored knowledge and give a reasonable output to a new input. The non-linear activation function transforms the similarity measure into a real, bound output. There are many different types of mathematical functions used to model non-linear activation functions. Some of the possible shapes are shown in Figure 4.2.

As in biological neural systems, the power of neural computing comes from the collective action of a large number of neurons interconnected in a multilayer structure. The multilayer of neurons consists of more than one layer of simple neurons. The repeated use of simple neurons is one of the attractive features of neural systems. This structure is known as a multilayer perceptron (MLP) network.

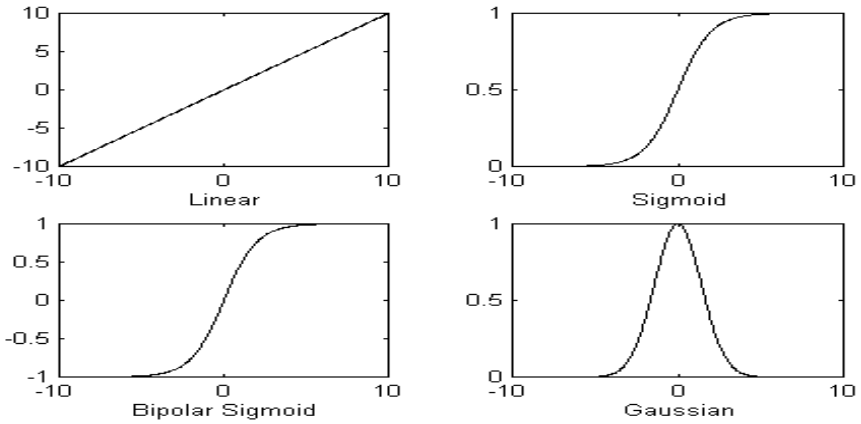

Figure 4.2 Examples of activation functions

A MLP network has demonstrated ability as a function approximator and is used in many applications. For this classification problem, the MLP will be included as a reference paradigm. This MLP consists of three layers: input layer, hidden layer and output layer as shown in Figure 4.3. The inputs of the network are assumed to be normalized to real values roughly between 0 and 2 as part of the preprocessing. This normalization is usually a linear transformation on the feature data. The normalized inputs will be fed into the network along with a bias (+1) in the input layer.

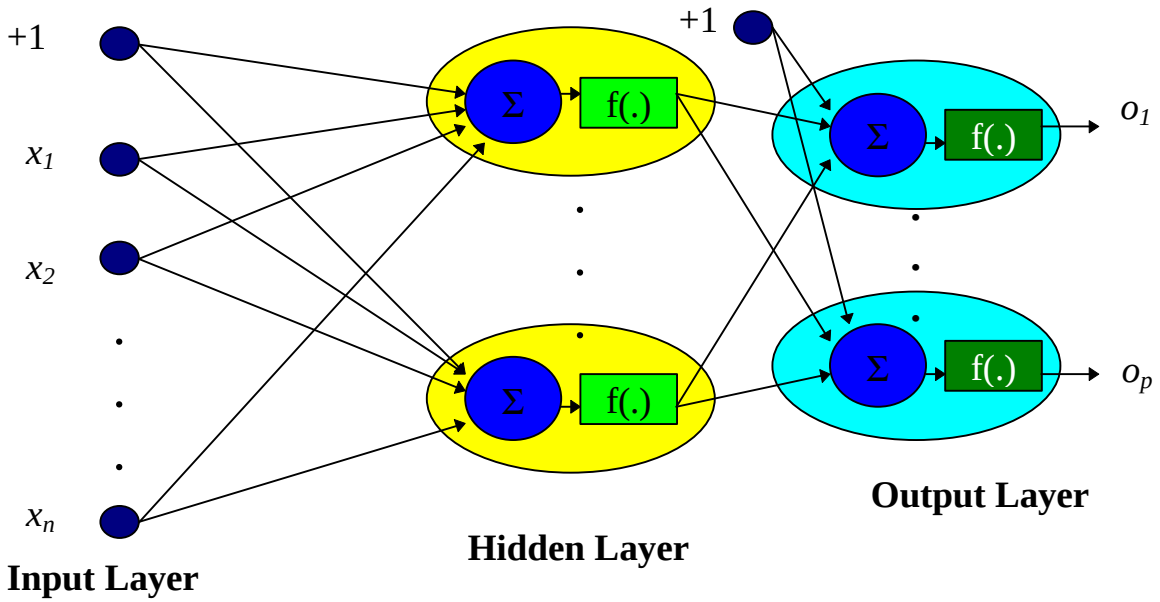

Figure 4.3 Multilayer perceptron (MLP) network

The nonlinear functions used in this network are normally taken from the set of sigmoids, bi-sigmoids, hyperbolic tangents and Gaussian functions. The function does not have to be the same throughout the layer, but in general they are kept to be the same function to simplify the network. A MLP can have more than one hidden layer. The more hidden layers, the more power it will have, but with a trade off of training difficulty. It has been shown that a MLP with one hidden layer has the capability to approximate any function with acceptable degree of accuracy if there are enough hidden nodes. For this classification problem, the MLP will consist of only one hidden layer. The output layer typically consists of as many neurons as the number of classes. If we use a sigmoid function as the output activation function, the output training data should be well within the linear region of this function. In this classification problem, the output training data will be a binary type, e.g., $[1\ 0\ 0\ 0]^T$ for data in class number 1. If we use a sigmoid function, the output training data might be $[0.9\ 0.1\ 0.1\ 0.1]^T$ instead of $[1\ 0\ 0\ 0]^T$ to act in the linear range of the sigmoid. Otherwise, the networks will tend to saturate to achieve 1 or 0 as an output. Another approach to this sigmoid function problem is to scale the value of the function to cover 1 and 0. For example, the maximum value of the function might be 1.3 and the minimum value might be -0.3.

4.3.3 Training Algorithm

MLP networks can be trained by several methods but the most popular method is the error backpropagation algorithm. This algorithm uses a gradient-descent to update the weights of the network. Although there are several methods to improve the rate of convergence of the error backpropagation, the standard error backpropagation is used through out this research. The main concern in this study is not the rate of convergence but the possibility of using such a network as a classifier. The error backpropagation algorithm may be summarized as follows:

a) Initialization

Specify network size, e.g., number of neurons in hidden layer and initialize all weights with small uniformly distributed random numbers. The selection of the network size is problem dependent and is up to the designer's experience.

b) Forward pass

Present training examples to the network and calculate the outputs of all neurons, layer by layer.

$$v_h = \sum_{i=0}^{n} w_{hi} x_i \tag{4.16}$$

$$O_h = f_h(v_h) \tag{4.17}$$

$$v_p = \sum_{h=0}^{H} w_{ph} O_h \tag{4.18}$$

$$O_p = f_p(v_p) \tag{4.19}$$

where

x_i : the i^{th} input of the network, i=0,1, ..., n and x_0 is a bias term (+1)

w_{hi} : the weight of the hidden layer that connects between the i^{th} input and the h^{th} neuron in the hidden layer

$f_h(.)$: the function used in the hidden layer and h=1,2, ..., H

O_h : the output of the h^{th} hidden neuron and O_0 is the bias term (+1)

w_{ph} : the weight of the output layer that connects between the h^{th} neuron in the hidden layer and the p^{th} neuron in the output layer

$f_p(.)$: the function used in the output layer and p=1,2, ..., P

O_p : the p^{th} output of the output layer.

c) Backward pass

Calculate the error and propagate that error backward to calculate a necessary change in weights, layer by layer.

$$e_p = O_p - d_p \tag{4.20}$$

$$\delta_p = e_p f_p'(v_p) \tag{4.21}$$

$$\delta_h = f_h'(v_h) \sum_{p=1}^{P} \delta_p w_{ph} \tag{4.22}$$

where

d_p : the desired p^{th} output

$f_p'(v_p)$: the derivative of the function used in the output layer, e.g.,

$f_p'(v_p) = O_p(1 - O_p)$ for sigmoid function

δ_p : the local gradient of the p^{th} output neuron

$f_h'(v_h)$: the derivative of the function used in the hidden layer, e.g.,

$f_h'(v_h) = O_h(1 - O_h)$ for sigmoid function

δ_h : the local gradient of the h^{th} hidden neuron

d) Update parameters

Update all weights of the network by equation

$$w_{hi}(k+1) = w_{hi}(k) + \alpha_h\left[w_{hi}(k) - w_{hi}(k-1)\right] - \eta_h \delta_h(k) x_i(k) \qquad (4.23)$$

$$w_{ph}(k+1) = w_{ph}(k) + \alpha_p\left[w_{ph}(k) - w_{ph}(k-1)\right] - \eta_p \delta_p(k) O_h(k) \qquad (4.24)$$

e) Iteration

Iterate the computation by presenting new training examples to the network until the weights stabilize their values or the sum squared error of the whole training set is at acceptably small value.

For complete information about MLP and error backpropagation algorithm, please consult [38][39][40][66][112].

4.4 Fuzzy Logic Systems (FLS)

A Fuzzy Logic System (FLS) is a system utilizing fuzzy set theory and its operations to solve a given problem. The FLS can be classified into three broad types: pure fuzzy logic systems, Takagi and Sugeno's fuzzy system and fuzzy logic systems with fuzzifier and defuzzifier [105]. The FLS with fuzzifier and defuzzifier is used throughout this research. It composes of fuzzifier, fuzzy rule base, fuzzy inference engine and defuzzifier as shown in Figure 4.4. This type of FLS is the most widely used FLS.

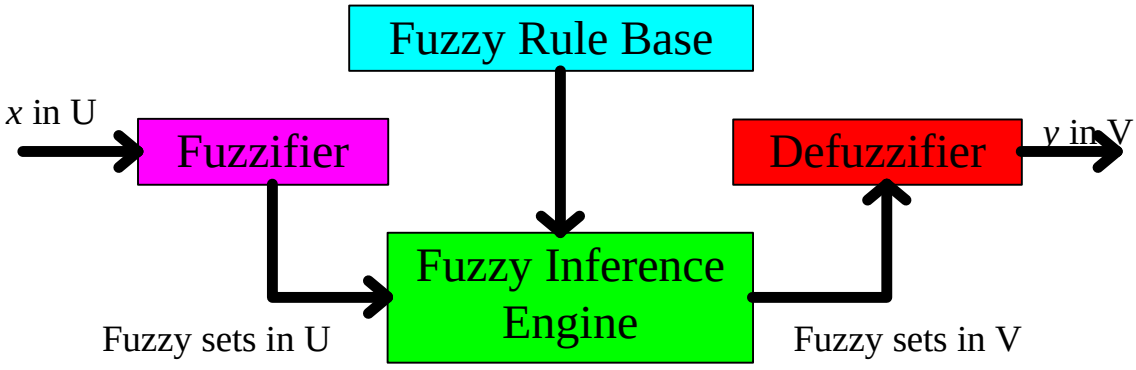

Figure 4.4 Fuzzy logic systems with fuzzifier and defuzzifier

The fuzzy rule base composes of fuzzy IF-THEN rules. A fuzzy rule is in the following form:

IF x_1 is F_1 and ... and q is F_n THEN y is G.

where F_i is fuzzy set in U_i and G is fuzzy set in V and $U_i \subset R$, $V \subset R$. Fuzzy sets are also known as linguistic variables; and, x_i and y are measured variables. Most researchers use and analyze only multi-input-single-output FLS, since a multi-input multi-output FLS can be decomposed into a multi-input single output FLS for each output. In classification problems with multi-classes, the FLS used as the classifier will be a MIMO FLS. Although, the decomposed system is easier to analyze than a nondecomposed one, the overall system may be too large and not as compact as for neural networks. In this research, a MIMO FLS will be used as a composed FLS. In this way the fewer rules are required. The fuzzy IF-THEN rules are then in the following form:

IF x_1 is F_1, x_2 is F_2, ..., and x_n is F_n THEN y_1 is G_1, y_2 is G_2 , ..., and y_p is G_p .

The fuzzy rule base is the most important part in FLS. The other three parts of a FLS use these rules in solving problems.

The fuzzifier in the FLS maps a crisp point $\mathbf{x}$ in $\boldsymbol{U}$ into a fuzzy set in $\boldsymbol{U}$. There are two types of fuzzification: singleton and non-singleton. A singleton fuzzifier maps a crisp point $\mathbf{x}$ into a fuzzy singleton set with support $\mathbf{x}'$ and $\mu(\mathbf{x}) = 1$ for $\mathbf{x} = \mathbf{x}'$ and $\mu(\mathbf{x}) = 0$ for all other $\mathbf{x}$ in $\boldsymbol{U}$. The non-singleton fuzzifier, on another hand, maps a crisp point $\mathbf{x}$ into a fuzzy set in which $\mu(\mathbf{x}) = 1$ for $\mathbf{x} = \mathbf{x}'$ and $\mu(\mathbf{x})$ decreases from unity as $\mathbf{x}$ moves away from $\mathbf{x}'$. The most widely used fuzzifier is the singleton fuzzifier. Singleton fuzzifiers will be used throughout this research.

The fuzzy inference engine is a system that maps fuzzy sets into fuzzy sets by means of $\mu_{A \to B}(\mathbf{x}, \mathbf{y})$ as shown in Figure 4.5.

Each rule R^l determines a fuzzy set $B^l = A_x \circ R^l$ for each output y such that

$$\mu_{B^l}(y) = \mu_{A_x \circ R^l}(y)$$

$$= \sup_{x \in A_x} \left[\mu_{A_x}(\mathbf{x}) * \mu_{A \to B}(\mathbf{x}, y) \right] \tag{4.25}$$

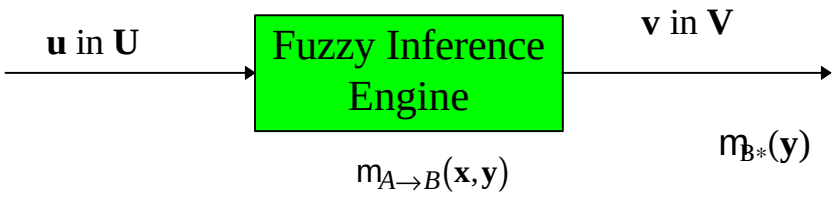

Figure 4.5 Fuzzy inference engine

Since we use a singleton fuzzifier, all $\mu_{A_x}(\mathbf{x}) = 1$, at $\mathbf{x} = \mathbf{x}'$ then

$$\begin{aligned} \mu_{B'}(y) &= \sup_{x \in A_x} \left[\mu_{A_x}(\mathbf{x}) * \mu_{A \to B}(\mathbf{x}, y) \right] \\ &= \mu_{A_x}(\mathbf{x}') * \mu_{A \to B}(\mathbf{x}', y) \\ &= 1 * \mu_{A \to B}(\mathbf{x}', y) = \mu_{A \to B}(\mathbf{x}', y) \end{aligned} \tag{4.26}$$

Mendel shows in [67] that the minimum or product implication preserves cause and effect, i.e., $\mu_{p \to q}(x, y)$ is fired only when both antecedent and consequent of the IF-THEN rule are true. These two inferences are, indeed, the most widely used inferences in the engineering applications of fuzzy logic. Thus,

$$\mu_{A \to B}(\mathbf{x}', y) = \min\left[\mu_A(\mathbf{x}'), \mu_{B'}(y) \right] \tag{4.27}$$

or

$$\mu_{A \to B}(\mathbf{x}', y) = \mu_A(\mathbf{x}') \mu_{B'}(y) \tag{4.28}$$

Since we normally have more than one rule in FLS, all rules must be combined such that only one result is given for each output. Most people connect rules by using fuzzy union. Kosko, on another hand, combines rules by addition [58]. Both methods are used in this research. In the previous chapter, the FLS was used to approximated the number of windows after the peak. Three rules will be used as examples. These rules are:

R^1: IF **P** is **Small** and **Nwd** is **Small** THEN **Nwd-After** is **Small**.

R^2: IF **P** is **Medium** and **Nwd** is **Medium** THEN **Nwd-After** is **Medium**.

R^3: IF **P** is **Medium** and **Nwd** is **Big** THEN **Nwd-After** is **Big**.

Suppose that Nwd = 22 and P = 75000, the results of fuzzy inference are shown graphically in Figures 4.6-4.8.

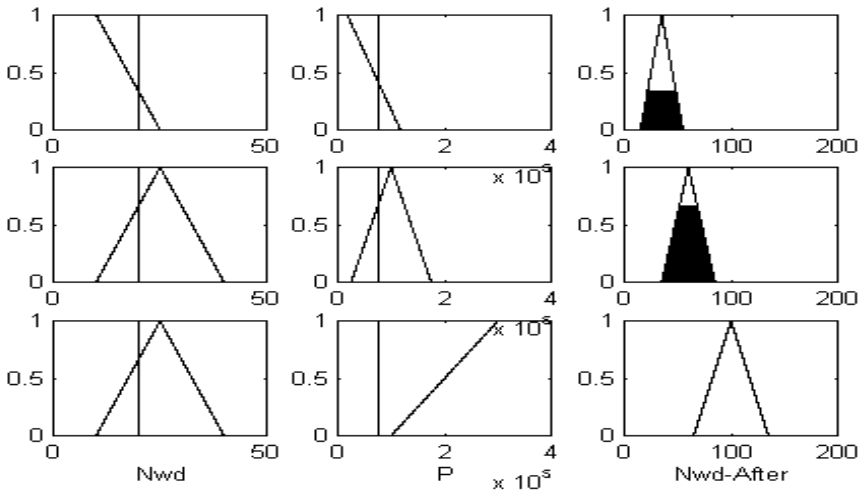

Figure 4.6 Minimum inference

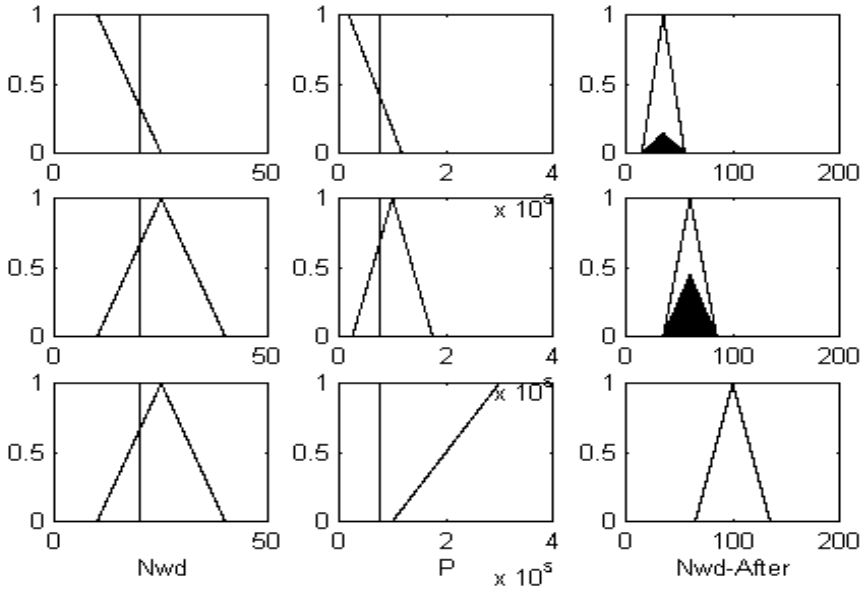

Figure 4.7 Product inference

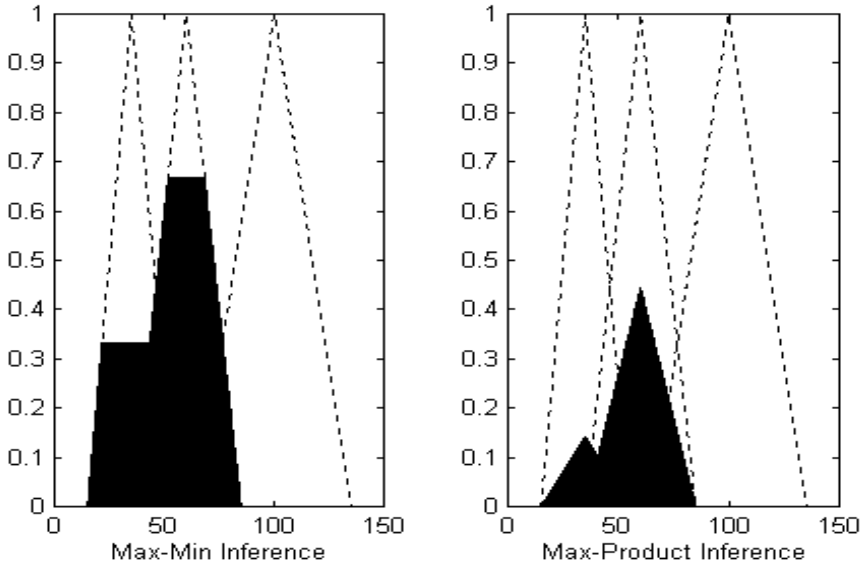

Figure 4.8 The overall output fuzzy set obtained by using Max-Min and Max-Product inference

As can be seen from Figure 4.8, the output of the inference engine is a fuzzy set characterized by the membership values shown. To use this result in a real world problem, it must be converted into a real number. This process is called defuzzification. There are many proposed defuzzifiers in the literature. This is one problem of FLS. There is no clear transformation from a fuzzy set to a real world value. The output of an inference engine is an area with an irregular shape depending on the membership functions used in the consequent part, the membership value of each rule and the inference type used. Defuzzification converts this area to a real value. There are several methods used in defuzzification such as center average, centroid of area, etc. The centroid calculation approach is a logical answer to defuzzification because it uses all information available to compute the output. Centroid defuzzification can be put into equation form as

$$\bar{y} = \frac{\left[\int_S y \mu_B(y) dy\right]}{\left[\int_S \mu_B(y) dy\right]} \tag{4.29}$$

where S is the support of $\mu_B(y)$. If S is discrete, the output can be approximated by

$$\bar{y} = \frac{\sum_{i=1}^{I} y_i \mu_B(y)}{\sum_{i=1}^{I} \mu_B(y)} \tag{4.30}$$

One problem of the centroid defuzzifier is intensive computation. The center average defuzzifier, on the other hand, is easy to compute and use. The center average defuzzifier has the formula

$$y_c = \frac{\sum_{m=1}^{M} \bar{y}^m \mu_{B^m}\left(\bar{y}^m\right)}{\sum_{m=1}^{M} \mu_{B^m}\left(\bar{y}^m\right)} \tag{4.31}$$

where $\bar{y}^m$ is the center of gravity of the fuzzy set B^m and M is the number of rules. The problem with the center average defuzzifier is that it suffers from not using the entire shape of the consequent membership function. Regardless of whether the max-min or max-product inference is used, the result of the center average defuzzifier is still the same, regardless of whether the shape is narrow or wide. The modified version of the center average defuzzifier is called the modified center defuzzification. The modified center average defuzzifier can be formulated as in [67] :

$$y_{mc} = \frac{\sum_{m=1}^{M} \bar{y}^m \mu_{B^m}\left(\bar{y}^m\right) \Big/ \delta^{m^2}}{\sum_{m=1}^{M} \mu_{B^m}\left(\bar{y}^m\right) \Big/ \delta^{m^2}} \tag{4.32}$$

where δ^m is a measure of the spread of the consequent part for rule R^m. Wang also proposes the modified center average defuzzifier as [105]

$$y_{mc} = \frac{\sum_{m=1}^{M} \bar{y}^m \mu_{B^m}\left(\bar{y}^m\right) \Big/ \delta^{m}}{\sum_{m=1}^{M} \mu_{B^m}\left(\bar{y}^m\right) \Big/ \delta^{m}} \tag{4.33}$$

The result of each defuzzification method is shown in Figure 4.9. All results are close. Surprisingly, the outputs of both modified center averages are further from the centroid calculation method than the center average method.

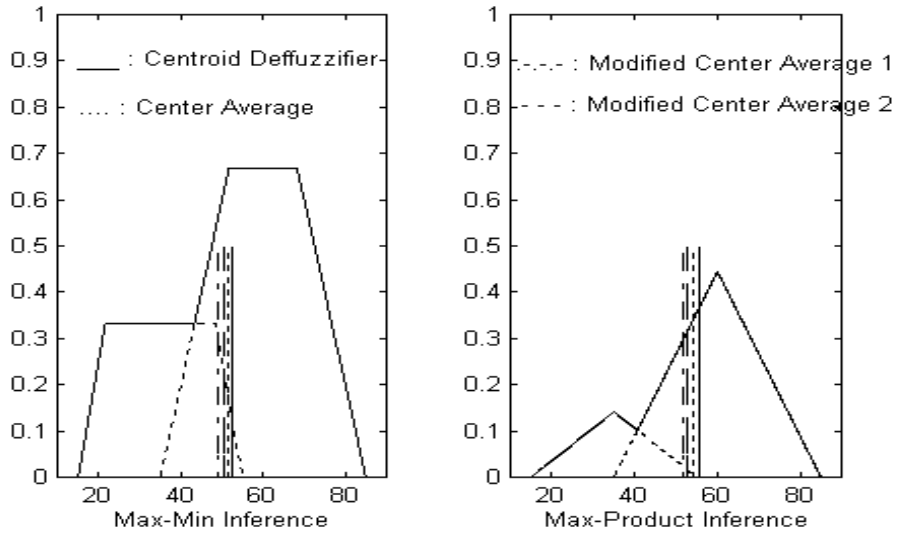

Figure 4.9 The approximated output of different defuzzifiers

4.5 Adaptive Fuzzy Logic System (AFLS)

An adaptive fuzzy logic system (AFLS) is a fuzzy logic system having adaptive rules. Its structure is the same as a normal FLS but its rules are derived and extracted from given training data. In other words, its parameters can be trained like a neural network approach, but with its structure in a fuzzy logic system structure. Since we have general ideas about the structure and effect of each rule, it is straight forward to effectively initialize each rule. This is a tremendous advantage of AFLS over its ANN counterpart. A disadvantage of AFLS is its computation expense. With the same number of nodes or rules, the AFLS always has more computations than the MLP. The AFLS is one type of FLS with a singleton fuzzifier and center average defuzzifier. The centroid defuzzifier cannot be used because of its computation expense and that it prohibits using the error backpropagation training algorithm. The proposed AFLS consists of two

defuzzification approaches, center average and a new defuzzification approach, balance of area (BOA). Each approach will be derived, compared and tested in the classification problem.

4.5.1 MIMO Adaptive Fuzzy Logic System with Center Average Defuzzification

4.5.1.1 Overview

This AFLS has the same approach as the system presented by Wang [105]. The only difference in the structure is that the proposed AFLS is multi-input and multi-output (MIMO). The MIMO AFLS has the following rule form:

IF x_1 is c_1, x_2 is c_2, ..., and x_n is c_n THEN y_1 is o_1, y_2 is o_2 , ..., and y_p is o_p .

This multi-output rule can be separated into multiple single output rules as:

IF x_1 is c_1, x_2 is c_2, ..., and x_n is c_n THEN y_1 is o_1.

IF x_1 is c_1, x_2 is c_2, ..., and x_n is c_n THEN y_2 is o_2.

...

and IF x_1 is c_1, x_2 is c_2, ..., and x_n is c_n THEN y_p is o_p.

Thus, the analysis of the MIMO-AFLS will be as same as MISO-AFLS. In other words, all techniques available in MISO-AFLS will be applicable to MIMO-AFLS.

The proposed MIMO-AFLS has a singleton fuzzifier, product-inference engine and center average defuzzifier. It has a feedforward structure as shown in Figure 4.10 with an extra "fuzzy basis" layer. This fuzzy basis layer consists of fuzzy basis nodes for each rule. A fuzzy basis node has the following form:

$$\phi_m(\bar{x}) = \frac{\mu_m(\bar{x})}{\sum_{l=1}^{L} \mu_l(\bar{x})} \tag{4.34}$$

where $\phi_m(\bar{x})$ is a fuzzy basis node for rule m and $\mu_m(\bar{x})$ is a membership value of rule m.

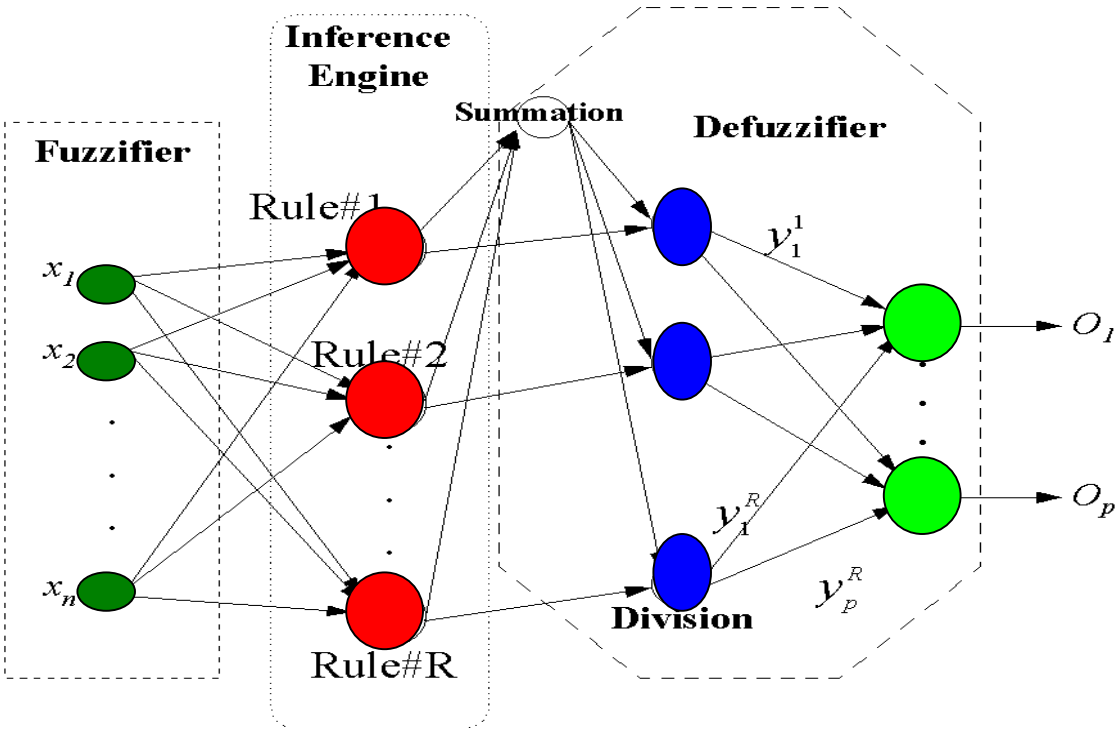

Figure 4.10 MIMO-AFLS with center average defuzzification

Since we use a product-inference, $\mu_m(\bar{x})$ is in the following form:

$$\mu_m(\bar{x}) = \prod_{i=1}^{n} \mu_{F_i^m}(x_i) \tag{4.35}$$

where $\mu_{F_i^m}(x_i)$ is a membership value of the i^{th} input of rule m. If we use a Gaussian shape as a membership function of each input of each rule, then, $\mu_{F_i^m}(x_i)$ will be in the following form:

$$\mu_{F_i^m}(x_i) = \exp\left(-\frac{\left(x_i - c_i^m\right)^2}{2\left(b_i^m\right)^2}\right) \tag{4.36}$$

where c_i^m and b_i^m are the center and spread parameters, respectively, of the membership function of the i^{th} input of the m^{th} rule. Thus, the output of AFLS is a linear combination of the weighted outputs of the fuzzy basis, is in the following form:

$$O_p(\bar{x}) = \sum_{m=1}^{M} y_{pm} \mathsf{f}_m(\bar{x}) \tag{4.37}$$

where y_{pm} is interpreted as the center of the membership function of the p^{th} output of the m^{th} rule. This is why its name is center average defuzzification. In this form the AFLS is very similar to radial basis function networks (RBFN). The differences between the two networks are the basis function and the bias term in the output layer. The RBFN uses a radial basis function where as the AFLS uses a fuzzy basis function. If we use a Gaussian kernel function as a basis function of RBFN and a Gaussian shape function as a membership function in the AFLS, then, the output of the hidden node in RBFN and the membership value of that rule will be the same with the same center and spread parameters. With AFLS this membership value will be normalized by the sum of membership values of all rules according to the fuzzy basis function. Thus, the effect of one hidden node in RBFN and one rule in AFLS are not the same. Any change in one rule will affect all rules. This effect has both advantages and disadvantages. This effect will make the system learn faster but it will make AFLS less capable of structural adaptation. This fuzzy basis function will make a rule of AFLS localized to a specific region in the input space but, all rules together still cover all the input space. Another difference is that the RBFN has a bias term in the output node but the AFLS does not.

4.5.1.2 Training Procedure MIMO-AFLS with Center Average Defuzzification

Let us define the objective function as:

$$J = \sum_{k=1}^{K} J_k \tag{4.37}$$

where K is the number of training patterns, J_k is the sum of squared error for the k^{th} pattern. Then, J_k is defined as:

$$J_k = \frac{1}{2}\sum_{p=1}^{P}\left(O_p(\bar{x}_k) - d_p(\bar{x}_k)\right)^2 \tag{4.38}$$

where P is the number of outputs and d_p is the desired response of the p^{th} output. $O_p(\bar{x}_k)$ is defined as in equation (4.37). The update equation of y_{pm} is as in the form:

$$y_{pm}(n+1) = y_{pm}(n) + m_y\left(y_{pm}(n) - y_{pm}(n-1)\right) - h_y \frac{\partial J}{\partial y_{pm}}\bigg|_n \tag{4.39}$$

and

$$\frac{\partial J}{\partial y_{pm}} = \frac{\partial J}{\partial J_k}\frac{\partial J_k}{\partial e_p}\frac{\partial e_p}{\partial O_p}\frac{\partial O_p}{\partial y_{pm}}$$

$$= \sum_{k=1}^{K} e_p^k f_p^{k\prime}\left(\bar{x}_k\right) \tag{4.40}$$

The update equation of the center is in the following form:

$$c_i^m(n+1) = c_i^m(n) + m_c\left(c_i^m(n) - c_i^m(n-1)\right) - h_c \frac{\partial J}{\partial c_i^m}\bigg|_n \tag{4.41}$$

and if we use a Gaussian shape membership function, then

$$\frac{\partial J}{\partial c_i^m} = \frac{\partial J}{\partial J_k}\frac{\partial J_k}{\partial e_p}\frac{\partial e_p}{\partial O_p}\frac{\partial O_p}{\partial m_m}\frac{\partial m_m}{\partial m_{F_i^k}}\frac{\partial m_{F_i^m}}{\partial c_i^m} + \ldots \tag{4.42}$$

$$= \sum_{k=1}^{K}\left\{\sum_{p=1}^{P} e_p\left(y_{pm} - O_p^k\right)\right\} f_m^{k\prime}\frac{\left(x_i^k - c_i^m\right)}{{b_i^m}^2} \tag{4.43}$$

The update equation of the spread parameter is in the following form:

$$b_i^m(n+1) = b_i^m(n) + m_b\left(b_i^m(n) - b_i^m(n-1)\right) - h_b \frac{\partial J}{\partial b_i^m}\bigg|_n \tag{4.44}$$

and if we use a Gaussian shape membership function, then

$$\frac{\partial J}{\partial b_i^m} = \frac{\partial J}{\partial J_k}\frac{\partial J_k}{\partial e_p}\frac{\partial e_p}{\partial O_p}\frac{\partial O_p}{\partial \mu_m}\frac{\partial \mu_m}{\partial \mu_{F_i^k}}\frac{\partial \mu_{F_i^m}}{\partial b_i^m} + \ldots$$

$$= \sum_{k=1}^{K}\left\{\sum_{p=1}^{P} e_p\left(y_{pm} - O_p^k\right)\right\}\mu_m^k \frac{\left(x_i^k - c_i^m\right)^2}{{b_i^m}^3} \qquad (4.45)$$

All parameters of the network will be updated according to the above equations. The initial center, c_i^m and y_{pm} are randomly selected from the k^{th} training data, x_i^k and d_p^k respectively. The initial spread parameter, b_i^m, is determined by

$$b_i = \frac{\max(x_i) - \min(x_i)}{N} \qquad (4.46)$$

where b_i is a spread parameter of the i^{th} input of all rules and N is the number of rules. Any available technique in feedforward ANNs will be applicable to AFLS. Indeed, the learning of the AFLS is much faster than MLP.

This AFLS will be tested with the classification problems and its performance compared with the other networks.

4.5.2 MIMO Adaptive Fuzzy Logic System with Balance of Area Defuzzification

4.5.2.1 Overview

As mentioned earlier, one problem of AFLS is its defuzzification. The most popular defuzzification method is the centroid calculation that returns the centroid of the area formed by the consequent membership function, the membership value of its rules and the max-min or max-product inference. In the case of a discrete universe, the centroid calculation yields

$$y = \frac{\sum_{q=1}^{Q} \mu_y\left(y_b^q\right) y_b^q}{\sum_{q=1}^{Q} \mu_y\left(y_b^q\right)} \tag{4.47}$$

where Q is a number of quantization levels of the output. The higher Q is, the finer y will be. The computation will increase as Q increases as a trade off. Thus, this method is very computationally expensive, so that it is not appropriate to use this method in AFLS. In other words, it will be very computationally expensive to make this method adaptive. Since this method provides good performance, its main characteristics, center of gravity and use of the shape of membership function, will be preserved in the design of a new defuzzification approach.

As mentioned earlier, the overall output of the system may be the result of fuzzy union or the addition of rule outputs as in Kosko's method. The following AFLS will use Kosko's method with product inference.

The density (D) is defined as mass (M) per unit volume (V).

$$D = \frac{M}{V}$$

or
$$M = VD \tag{4.48}$$

Assume that we use the same material and all shapes have the same thickness, T, then

$$M = ATD \tag{4.49}$$

where A is an area and T is a thickness. Suppose the shape of the membership function used in the consequent part is symmetric, e.g., triangular, Gaussian or bell shape as shown in Figure 4.11. The center of gravity will pass through the halfway point of the base of that shape. For example, if we use a triangular shape and product-inference as a t-norm, then the shape of the consequent part of rule m will be shown as in Figure 4.12.

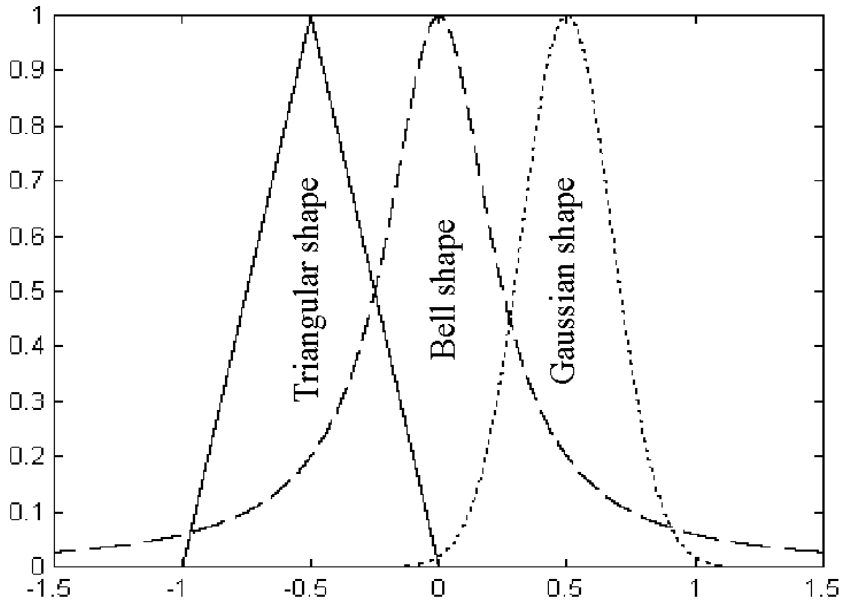

Figure 4.11 Symmetric membership function

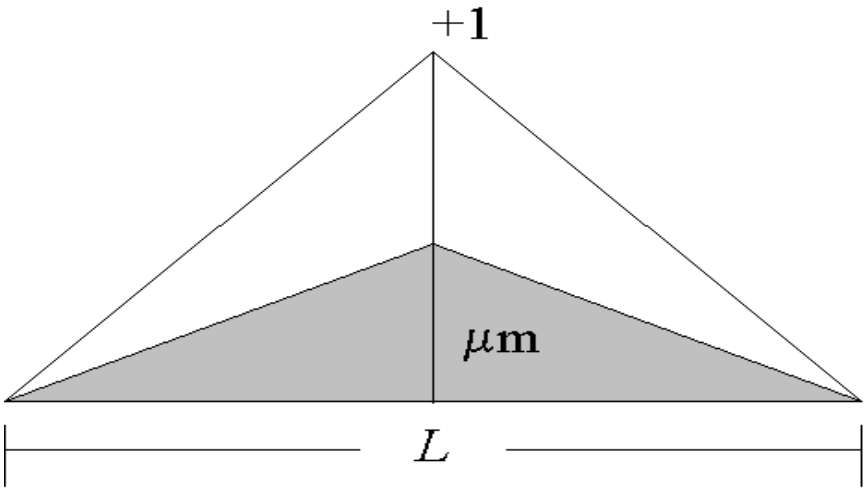

Figure 4.12 Triangular shape membership function

The shaded area (*A*) is $\frac{1}{2}\mu_m L_m$. Imagine that we have the consequent part of each rule placed on the massless beam having the pivot point at origin as shown in Figure 4.13.

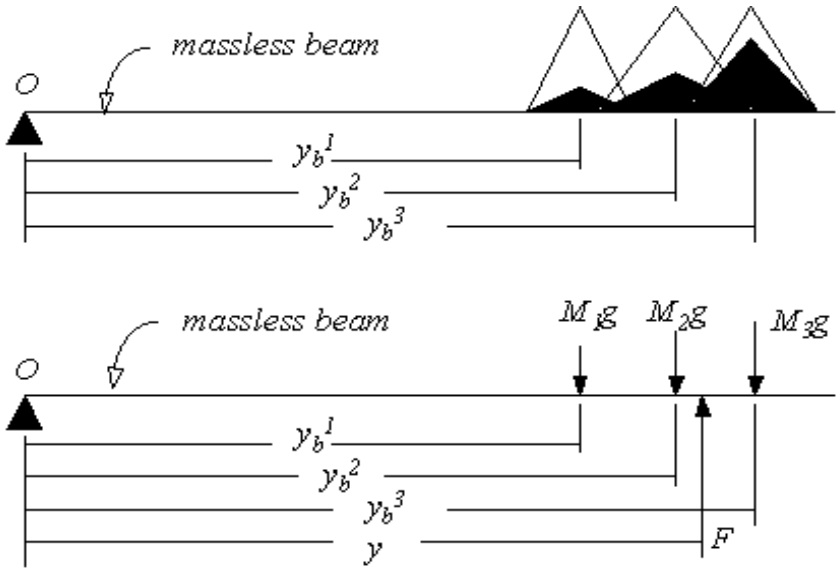

Figure 4.13 Consequent fuzzy set placed on massless beam

Then,

$$F = M_1 g + M_2 g + M_3 g = (M_1 + M_2 + M_3)g \tag{4.50}$$

For balance

$$Fy = M_1 g y_b^1 + M_2 g y_b^2 + M_3 g y_b^3 = \left(M_1 y_b^1 + M_2 y_b^2 + M_3 y_b^3\right)g \tag{4.51}$$

$$y = \frac{\left(M_1 y_b^1 + M_2 y_b^2 + M_3 y_b^3\right)g}{F} = \frac{\left(M_1 y_b^1 + M_2 y_b^2 + M_3 y_b^3\right)g}{\left(M_1 + M_2 + M_3\right)g}$$

$$y = \frac{\left(M_1 y_b^1 + M_2 y_b^2 + M_3 y_b^3\right)}{\left(M_1 + M_2 + M_3\right)} \tag{4.52}$$

Assume that D in Equation (4.49) is the same, thus,

$$y = \frac{\left(A_1 y_b^1 + A_2 y_b^2 + A_3 y_b^3\right)TD}{\left(A_1 + A_2 + A_3\right)TD} = \frac{\left(A_1 y_b^1 + A_2 y_b^2 + A_3 y_b^3\right)}{\left(A_1 + A_2 + A_3\right)} \tag{4.53}$$

The area, A, will be depended upon the membership function used. Under the assumption

of symmetric shape this method will have comparable capability with the centroid calculation method to approximate the output from the fuzzy set in the consequent part.

If the triangle is used as a membership function and we use the max-product inference, then the shaded area, *A*, will be derived as:

$$A_m = \tfrac{1}{2}\mu_m L_m \tag{4.54}$$

From equation (4.53), the output, *y* will be

$$y = \frac{\frac{1}{2}\mu_1 L_1 y_b^1 + \frac{1}{2}\mu_2 L_2 y_b^2 + \frac{1}{2}\mu_3 L_3 y_b^3}{\frac{1}{2}\mu_1 L_1 + \frac{1}{2}\mu_2 L_2 + \frac{1}{2}\mu_3 L_3} = \frac{\mu_1 L_1 y_b^1 + \mu_2 L_2 y_b^2 + \mu_3 L_3 y_b^3}{\mu_1 L_1 + \mu_2 L_2 + \mu_3 L_3} \tag{4.55}$$

From the integral table [95],

$$\int_0^\infty e^{-a^2 x^2}\, dx = \frac{1}{2a}\sqrt{\pi} \tag{4.56}$$

Let

$$x = \left(x_i - C_i^m\right)$$

and

$$a^2 = \frac{1}{2b_m^2} \tag{4.57}$$

If we use a Gaussian shape as a membership function, then the area, *A*, will be

$$A_m = \sqrt{2}\mu_m b_m \sqrt{\pi} \tag{4.58}$$

From Equation (4.53), the output, *y*, will be

$$y = \frac{\sqrt{2}\mu_1 b_1 \sqrt{\pi}\, y_b^1 + \sqrt{2}\mu_2 b_2 \sqrt{\pi}\, y_b^2 + \sqrt{2}\mu_3 b_3 \sqrt{\pi}\, y_b^3}{\sqrt{2}\mu_1 b_1 \sqrt{\pi} + \sqrt{2}\mu_2 b_2 \sqrt{\pi} + \sqrt{2}\mu_3 b_2 \sqrt{\pi}} = \frac{\mu_1 b_1 y_b^1 + \mu_2 b_2 y_b^2 + \mu_3 b_3 y_b^3}{\mu_1 b_1 + \mu_2 b_2 + \mu_3 b_3} \tag{4.59}$$

In general form, the calculation of the output, *y*, will be

$$y_p = \frac{\sum_{m=1}^{M} \mu_m L_p^m y_p^m}{\sum_{m=1}^{M} \mu_m L_p^m} \qquad (4.60)$$

where y_p : the p^{th} output of the network

μ_m : the membership value of the m^{th} rule.

L_p^m : the spread parameter of the membership function in the consequent part of the p^{th} output of the m^{th} rule.

y_p^m : the center of the membership function in the consequent part of the p^{th} output of the m^{th} rule.

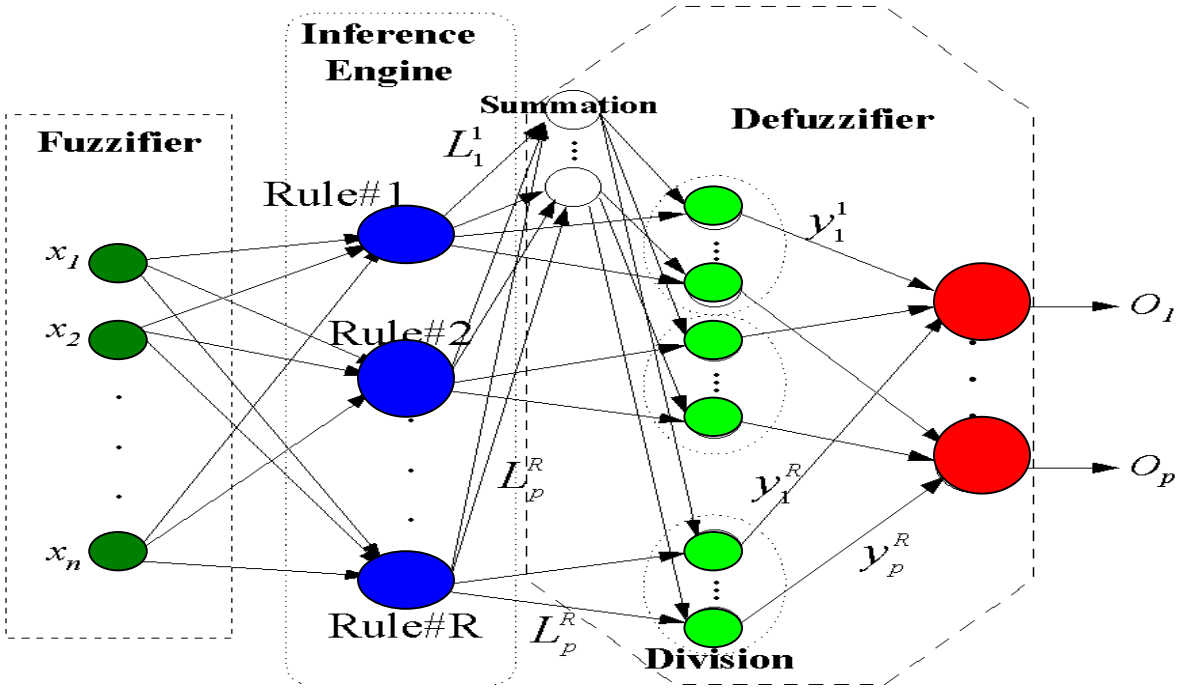

Figure 4.14 AFLS with balance of area (BOA) defuzzification method

The center average defuzzifier was modified such that the spread parameter was included into the equation by using common sense. This new defuzzification method is applied to the

defuzzification of the previous example. The results are shown in Table 4.1 and Figure 4.15. The BOA defuzzification is closer to COA than other methods while the number of calculations required is comparable with other methods as shown in Table 4.1.

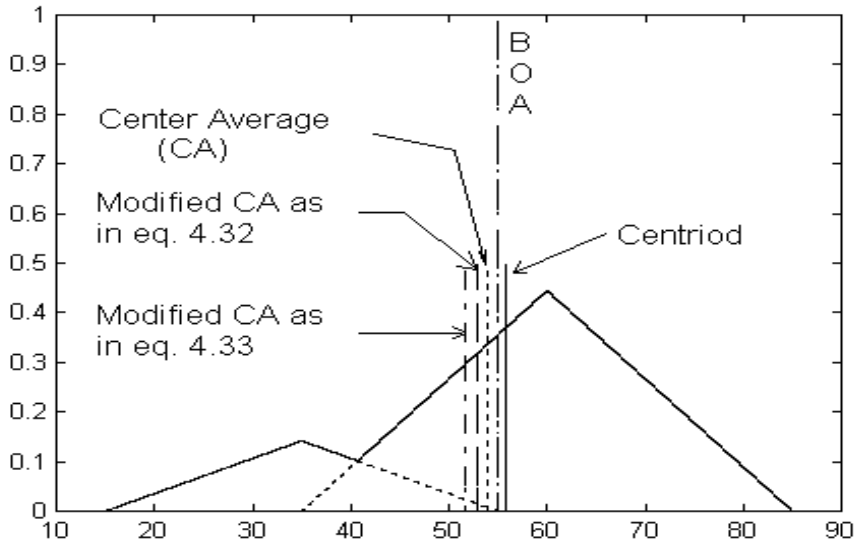

Figure 4.15 Results of different defuzzifiers

Table 4.1 Numerical results of different defuzzifiers

	Centroid of Area (COA) (0.01 increment)	Center Average (CA)	Modified Center Average # 1 (MCA#1)	Modified Center Average # 2 (MCA#2)	Balance of Area (BOA)
Output	55.8098	53.9573	51.6884	52.8771	54.9203
Number of Flops	20997	9	21	15	15

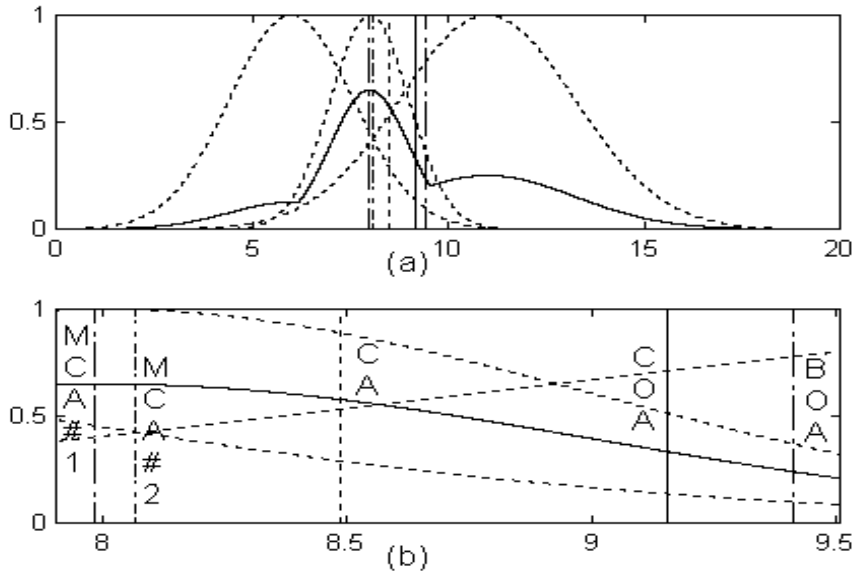

Figure 4.16 Example of Gaussian shape consequent membership functions and different defuzzification methods

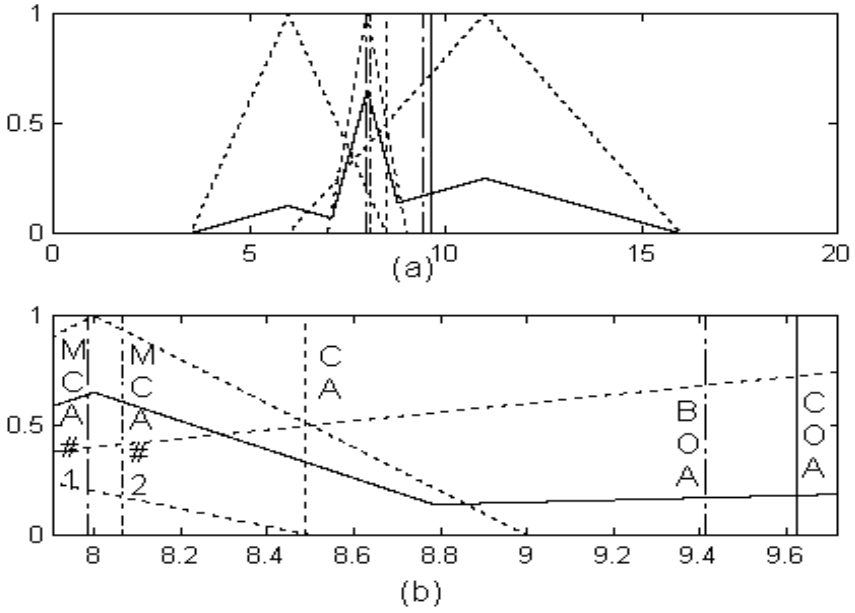

Figure 4.17 Example of triangular shape consequent membership functions and different defuzzification methods

For both Gaussian and triangular shaped membership function, the BOA defuzzifier gives results closer to the COA's than other defuzzification methods as shown in Figures 4.16 and 4.17. More tests are done by randomly generating the membership values of each rule, the shape (spread) of each membership function and the center of each function. The calculated output of each method is compared with the centroid of area defuzzifier result. There are 500 independent runs for each shape. The results are summarized in Tables 4.2 and 4.3.

Table 4.2 Results of Gaussian shape membership functions

Method	Mean of y_{COA}-y_i	Standard deviation of y_{COA}-y_i
CA	0.4287	0.7455
MCA#1	0.9058	2.2349
MCA#2	0.8199	1.6978
BOA	-0.1977	0.4623

Table 4.3 Results of triangular shape membership functions

Method	Mean of y_{COA}-y_i	Standard deviation of y_{COA}-y_i
CA	0.7453	1.0962
MCA#1	1.3196	2.5419
MCA#2	1.1899	2.0274
BOA	0.0860	0.2197

4.5.2.2 Training Procedure for MIMO-AFLS with BOA Defuzzifier

This defuzzification approach can also be adaptive like AFLS with center average defuzzification and neural networks. By using an error-backpropagation technique, the update equations can be derived. We define the objective function as

$$J = \sum_{k=1}^{K} J_k \qquad (4.61)$$

where K is the number of training patterns, J_k is the sum of squared error for the k^{th} pattern. Then, J_k is defined as:

$$J_k = \frac{1}{2}\sum_{p=1}^{P}\left(y_p(\bar{x}_k) - d_p(\bar{x}_k)\right)^2 \tag{4.62}$$

where P is the number of outputs and d_p is the desired response of the p^{th} output. $y_p(\bar{x}_k)$ is defined as in Equation (4.60). The update equation of y_p^m is in the form

$$y_p^m(n+1) = y_p^m(n) + m_y\left(y_p^m(n) - y_p^m(n-1)\right) - h_y \frac{\partial J}{\partial y_p^m}\bigg|_n \tag{4.63}$$

where

$$\frac{\partial J}{\partial y_p^m} = \frac{\partial J}{\partial J_m}\frac{\partial J_m}{\partial e_p}\frac{\partial e_p}{\partial y_p}\frac{\partial y_p}{\partial y_p^m}$$

$$= \sum_{k=1}^{K}\left\{\left(y_p^k - d_p^k\right)\frac{m_m^k L_p^m}{\sum_{m=1}^{M} m_m^k L_p^m}\right\} \tag{4.64}$$

where y_p^k is the p^{th} output of the network corresponding to the k^{th} pattern in the training data, d_p^k is the p^{th} desired output of the k^{th} pattern and m_m^k is the membership value of the m^{th} rule corresponding to the k^{th} pattern in the training data. The update equation of L_p^m is in the following form:

$$L_p^m(n+1) = L_p^m(n) + m_L\left(L_p^m(n) - L_p^m(n-1)\right) - h_L \frac{\partial J}{\partial L_p^m}\bigg|_n \tag{4.65}$$

where

$$\frac{\partial J}{\partial L_p^m} = \frac{\partial J}{\partial J_m}\frac{\partial J_m}{\partial e_p}\frac{\partial e_p}{\partial y_p}\frac{\partial y_p}{\partial L_p^m}$$

$$\frac{\partial J}{\partial L_p^m} = \sum_{k=1}^{K} \left[\left(y_p^k - d_p^k \right) \frac{\left\{ \left[\sum_{m=1}^{M} \mu_m^k L_p^m \right] \mu_m^k y_p^m - \left[\sum_{m=1}^{M} y_p^m \mu_m^k L_p^m \right] \mu_m^k \right\}}{\left[\sum_{m=1}^{M} \mu_m^k L_p^m \right]^2} \right] \tag{4.66}$$

$$\frac{\partial J}{\partial L_p^m} = \sum_{k=1}^{K} \left[\left(y_p^k - d_p^k \right) \frac{\mu_m^k}{\left[\sum_{m=1}^{M} \mu_m^k L_p^m \right]} \left\{ y_p^m - y_p^k \right\} \right] \tag{4.67}$$

where y_p^m is interpreted as a center of the membership function of the p^{th} output of the m^{th} rule in the consequent part of IF-THEN rule. The update equation of the center parameter, c_i^m, is in the form:

$$c_i^m(n+1) = c_i^m(n) + \alpha_c \left(c_i^m(n) - c_i^m(n-1) \right) - \eta_c \frac{\partial J}{\partial c_i^m}\bigg|_n \tag{4.68}$$

where

$$\frac{\partial J}{\partial c_i^m} = \frac{\partial J}{\partial J_m} \frac{\partial J_m}{\partial e_p} \frac{\partial e_p}{\partial y_p} \frac{\partial y_p}{\partial \mu_m} \frac{\partial \mu_m}{\partial \mu_{F_i^m}} \frac{\partial \mu_{F_i^m}}{\partial c_i^m} + \ldots \tag{4.69}$$

If we use a Gaussian shape, then

$$\frac{\partial \mu_{F_i^m}}{\partial c_i^m} = \mu_{F_i^m} \frac{\left(x_i^k - c_i^m \right)}{{b_i^m}^2} \tag{4.70}$$

and

$$\frac{\partial \mu_m}{\partial \mu_{F_i^m}} = \prod_{\substack{j=1 \\ j \neq i}}^{n} \mu_{F_i^j} \tag{4.71}$$

Thus,

$$\frac{\partial J}{\partial c_i^m} = \sum_{k=1}^{K}\left[\left\{\sum_{p=1}^{P}\left(y_p^k - d_p^k\right)\frac{L_{pl}\left[y_p^m - y_p^k\right]}{\sum_{m=1}^{M}\mu_m^k L_p^m}\right\}\mu_m^k \frac{\left(x_i^k - c_i^m\right)}{{b_i^m}^2}\right] \tag{4.72}$$

The update equation of the spread parameter, b_i^m, is in the form:

$$b_i^m(n+1) = b_i^m(n) + m_b\left(b_i^m(n) - b_i^m(n-1)\right) - \eta_b \left.\frac{\partial J}{\partial b_i^m}\right|_n \tag{4.73}$$

where

$$\frac{\partial J}{\partial b_i^m} = \frac{\partial J}{\partial J_m}\frac{\partial J_m}{\partial e_p}\frac{\partial e_p}{\partial y_p}\frac{\partial y_p}{\partial \mu_m}\frac{\partial \mu_m}{\partial \mu_{F_i^m}}\frac{\partial \mu_{F_i^m}}{\partial b_i^m} + \ldots \tag{4.74}$$

If we use a Gaussian shape as in equation AFLS-3, then

$$\frac{\partial \mu_{F_i^m}}{\partial b_i^m} = \mu_{F_i^m}\frac{\left(x_i^k - c_i^m\right)^2}{{b_i^m}^3} \tag{4.75}$$

Therefore,

$$\frac{\partial J}{\partial b_i^m} = \sum_{k=1}^{K}\left[\left\{\sum_{p=1}^{P}\left(y_p^k - d_p^k\right)\frac{L_{pl}\left[y_p^m - y_p^k\right]}{\sum_{m=1}^{M}\mu_m^k L_p^m}\right\}\mu_m^k \frac{\left(x_i^k - c_i^m\right)^2}{{b_i^m}^3}\right] \tag{4.76}$$

All equations derived are used to update all parameters during the training phase of the network. The initialization methods in the center average defuzzification can also be used in this approach. Since the desired output in the classification problem is primarily a binary output representing each class, therefore, the initial spread parameter, L_p^m, can be set to 0.75. This method can be interpreted that we have initially equal confidence in each rule. This spread parameter will be adjusted during training. This system will be applied to classification problems and compared its performance with other networks in Chapter 6.

Chapter 5

Unequal Learning of Classifiers

5.1 Overview

There are three main factors influencing the generalization of a network: the size and efficiency of the training data, the physical complexity of the problem at hand and the architecture of the network used [38]. The first two factors normally gain less attention than the third one. All factors are equally important to overall classification performance of a network. Unequal learning of different classes due to the complexity of classification and/or an uneven number of training data from different classes is likely to be encountered in practice. In an ideal case the number of training data should be equal, with an ample and representative amount for each class. In practice some class data are harder to obtain than others, giving us a problem of unequal numbers of training data. In reality, another case might be that one class may be harder to learn than others. This also causes the network to classify poorly for that class, and the cost of misclassification may be different from one class to another. This cost can be used as a classification performance index to determine how good or bad the classifier is.

This chapter describes a new approach to improve classification accuracy for every class. A method used to solve an unequal learning problem will be presented in the next section and used to train all proposed networks in the next chapter.

5.2 Unequal Learning of each Class

Unequal learning of different classes may be caused by the complexity of a classification problem and/or an unequal number of training data for each class. The complexity of a classification problem may be viewed as a learning difficulty for different classes. A classifier can

obviously classify an easy-to-learn class better than the most difficult one. As mentioned earlier, we want a classifier that can classify all classes well, not only one or two dominant classes. Thus, a slow down in learning must be put into effect for an easy-to-learn class and a speed up in learning for a difficult-to-learn class.

The performance measure used in this case is the average of sum-squared errors of each class. A well-learned class will have a small average of sum-squared error but a poorly learned class will have a large average of sum-squared error. This information during training is useful to reduce the imbalance of learning of such networks. Ideally, we want the network to learn equally from each class. In reality, there is always different learning from each class. This attempt is to reduce the effect of unequal learning during training.

The average of sum-squared error, β_c, for class c is defined as:

$$\beta_c = \frac{1}{N_c} \sum_{i_c=1}^{N_c} \left\{ \sum_{p=1}^{P} \left(y_p^{i_c} - d_p^{i_c} \right)^2 \right\} \tag{5.1}$$

where N_c = number of data in class c

P = number of outputs

$y_p^{i_c}$ = the p^{th} output of a network of the $i_c{}^{th}$ data

$d_p^{i_c}$ = the desired p^{th} output of the $i_c{}^{th}$ data.

The network should learn more about a class having a high β_c than a class having a lesser β_c. This can be done by using a linear, or nonlinear, function to transform the difference in β_c into a learning factor for each class. At this time a bell shaped function is used.

Let η_c be the learning factor due to unequal learning, then

$$\eta_c = \frac{1}{1 + \left(\frac{\beta_c - \beta_{\max}}{L} \right)^2} \tag{5.2}$$

where β_{max} is the maximum value of β_c over all classes, and L is the specified spread parameter to control how the sharpness of the bell function. In this way, if β_c is the maximum value, then, its learning factor, η_c, will be 1. If its β_c is small and far away from the maximum value of β_c, then, its learning factor, η_c, will be a small value depending on the spread parameter, L.

5.3 Unequal Number of Data

In an ideal case the number of training data should be equal and available in sufficient numbers for each class. In practice some class data, i.e., truck classes, are hard to get. This poses a problem of an unequal number of training data. This problem is very likely to be encountered in practice. If we present unequal training data to train a network, that network may favorably classify every class as the most dominant number one. This may be interpreted as an unequal learning problem because the network will obviously learn more about the class having more data than the lesser classes. Since the primary purpose of classifier design is to classify all classes correctly, it is unsuitable to favorably classify everything as the class having the most dominant number as in a statistical sense. There are two logical approaches: first, reduce the number of training data of other classes to be the same as the fewest numbers of data, second, duplicate the training data of all smaller numbers to be the same number as the most dominant one. In the first approach, we lose the opportunity to have the network train and learn with all the data, and the remaining data might help classifying all classes better. In the second approach there is no new useful information added to train a network. The duplicated data will have a repeated effect since it is just duplicated information. The only purpose of the latter approach is to have an equal number of training data for each class. This approach will cause more computation and longer training times than the first one.

To compromise between these two approaches, a learning factor due to uneven data is developed. This method uses all available data to train or test the network while reducing the

effect of an unequal number of data from each class during training. The learning factor can be formulated as:

$$\delta_c = \frac{N_t}{N_c N_p} \tag{5.3}$$

where N_t = total number of available training data

N_c = number of available training data from class c

N_p = number of outputs that is the same as number of classes.

The total errors used to update the network parameters approximately equally come from c different classes. If each data provides equal errors, then, the ideal case of having equal errors from each class is reached. Normally, each data set will have different errors; therefore, this learning factor will make only an equal error approximation.

5.4 Unequal Misclassification Costs among Different Classes

In classification problems different classes may have different misclassification costs. For example, if we want to classify between a toy hand grenade and a real hand grenade, the misclassification costs are obviously different. The misclassification cost of a real hand grenade as a toy one should be much higher than the other way around. In statistical classifiers such as Bayesian classifiers, the misclassification costs are used as risks of classification. The Bayesian classifier minimizes the total risks. In practice these risks are hard to quantify. They depend on the problem at hand. They could vary from problem to problem and person to person. There are no universal rules to quantify these risks. That is the main reason that most researchers treat them as equal risks.

These misclassification risks along with the classification results of a network can be used to measure that network's classification performance. The classification results may be put into a confusion matrix as shown in Table 5.1. The diagonals of this matrix indicate the correct classifications. We hope to design a classification system giving zeros to non-diagonal elements

of this confusion matrix. In practice it is hard to achieve this level of performance. Typically there is overlap among classes. This matrix also gives misclassification information. Some classes are close and overlapped, some not. If only the percentage correct classification is considered, as in most research, the performance of such a network will be ignored. Even though, the outcome is to correctly classify the objects, but the misclassification is sometimes more important than correct classification. The costs may be different from class to class. Classification costs are shown Table 5.2.

Table 5.1 Example of classification results confusion matrix

		CLASSIFIED			
		Class 1	**Class 2**	**Class 3**	**Class 4**
TRUTH	**Class 1**	2045	54	3	0
	Class 2	17	356	11	3
	Class 3	0	22	253	15
	Class 4	3	5	9	459

Table 5.2 Example of classification cost confusion matrix

		CLASSIFIED			
		Class 1	**Class 2**	**Class 3**	**Class 4**
TRUTH	**Class 1**	1	1.2	1.3	1.3
	Class 2	1.2	1	1.2	1.3
	Class 3	1.3	1.2	1	1.2
	Class 4	1.3	1.3	1.2	1

These two matrices will give complete classification information, both correct and incorrect classifications. Thus, total classification costs can be used to compare classification

performances among different classifiers. Most researchers use only a total percentage of correct classifications to compare among classifiers under an equal classification cost assumption.

In Bayesian classifiers the classification costs are used as the classification risks. Most researchers in neural network classification applications treat all costs equally during training. In this research these classification costs are used during training for both MLP and AFLS. In this way the classifiers directly learn to reduce misclassification costs.

5.5 Learning Factor Due to Unequal Learning, Unequal Number and Different Classification Costs

In classification problems, the ideal training is to train a network with equal learning from each class. If the learning factor due to unequal learning and unequal data numbers is used to train a classifier, then a network will approximately learn equally from each class. A more important task is to train a classifier having a total minimum classification cost. All these factors can be combined as one learning factor for each data set during the training phase. This learning modification is done only during the training phase. After training the classifiers will perform a normal classification task.

In this research the total learning factor comes from each learning factor: combining unequal learning, unequal number of training data and different classification costs. This learning factor can be formulated as

$$g_i = d_c \cdot r_c \cdot l_{pq} \tag{5.4}$$

where g_i = total learning factor of the i^{th} data, $i = 1, \ldots, n$.

d_c = the uneven number factor of class c, $c = 1, \ldots, C$.

r_c = the unequal learning factor of the previous epoch of class c, $c = 1, \ldots, C$.

l_{pq} = the classification cost of the given class p and classified as class q,

$p = 1, \ldots, C$ and $q = 1, \ldots, C$.

This learning factor will be added into the objective function for each classifier. The modified objective function is defined as:

$$J = \sum_{k=1}^{K} J_k \tag{5.5}$$

where K is the number of training patterns, J_k is the sum-squared error for the k^{th} pattern. J_k is defined as:

$$J_k = \frac{\gamma_k}{2} \sum_{p=1}^{P} \left(y_p(\bar{x}_k) - d_p(\bar{x}_k) \right)^2 \tag{5.6}$$

where P is the number of outputs and d_p is the desired response of the p^{th} output, y_p is the output of the classifier and γ_k is a total learning factor of the k^{th} pattern,. The learning factor of the k^{th} pattern, γ_k, is defined as in Equation 5.4. To avoid complicated update equations, the unequal learning factor is designed as a function of the previous errors instead of the current errors. In this way, the learning factors will be considered as constants for all derivatives in the error backpropagation algorithm. The original $\frac{\partial J_k}{\partial e_p}$ is defined as

$$\frac{\partial J_k}{\partial e_p} = e_p \tag{5.7}$$

The $\frac{\partial J_k}{\partial e_p}$ of the modified method will be

$$\frac{\partial J_k}{\partial e_p} = \gamma_k e_p \tag{5.8}$$

This is the only modification of the error backpropagation algorithm for the proposed classifiers in the previous chapter. In other words, the original e_p is replaced by $\gamma_k e_p$.

In MLP classifiers e_p in Equation (4.21) is replaced by $\gamma_k e_p$, then the new equation will be

$$\delta_p = \gamma_k e_p f_p'(v_p) \tag{5.9}$$

In the epoch update mode this update equation becomes

$$\delta_p = \sum_{k=1}^{K} \gamma_k e_p^k f_p'(v_p^k) \tag{5.10}$$

All the rest of the update equations are the same.

In AFLS with center average defuzzification (CA) the error, e_p^k, is replaced by $\gamma_k e_p^k$ for all e_p. The update equation (4.40) will become as

$$\frac{\partial J}{\partial y_{pm}} = \frac{\partial J}{\partial J_k}\frac{\partial J_k}{\partial e_p}\frac{\partial e_p}{\partial O_p}\frac{\partial O_p}{\partial y_{pm}} + \cdots$$

$$= \sum_{k=1}^{K} \gamma_k e_p^k \phi_p^k(\bar{x}_k) \tag{5.11}$$

The update equation (4.42) and (4.43) will become as

$$\frac{\partial J}{\partial c_i^m} = \frac{\partial J}{\partial J_k}\frac{\partial J_k}{\partial e_p}\frac{\partial e_p}{\partial O_p}\frac{\partial O_p}{\partial \mu_m}\frac{\partial \mu_m}{\partial \mu_{F_i^k}}\frac{\partial \mu_{F_i^m}}{\partial c_i^m} + \ldots \tag{5.12}$$

$$= \sum_{k=1}^{K} \gamma_k \left\{ \sum_{p=1}^{P} e_p \left(y_{pm} - O_p^k \right) \right\} \phi_m^k \frac{\left(x_i^k - c_i^m \right)}{b_i^{m2}} \tag{5.13}$$

The update equation (4.45) will become as

$$\frac{\partial J}{\partial b_i^m} = \frac{\partial J}{\partial J_k}\frac{\partial J_k}{\partial e_p}\frac{\partial e_p}{\partial O_p}\frac{\partial O_p}{\partial \mu_m}\frac{\partial \mu_m}{\partial \mu_{F_i^k}}\frac{\partial \mu_{F_i^m}}{\partial b_i^m} + \ldots$$

$$= \sum_{k=1}^{K} \gamma_k \left\{ \sum_{p=1}^{P} e_p \left(y_{pm} - O_p^k \right) \right\} \phi_m^k \frac{\left(x_i^k - c_i^m \right)^2}{b_i^{m3}} \tag{5.14}$$

In AFLS with balance of area defuzzification (BOA) the update equation (4.64) becomes as

$$\frac{\partial J}{\partial y_p^m} = \frac{\partial J}{\partial J_m}\frac{\partial J_m}{\partial e_p}\frac{\partial e_p}{\partial y_p}\frac{\partial y_p}{\partial y_p^m} + \cdots$$

$$= \sum_{k=1}^{K}\left\{\gamma_k\left(y_p^k - d_p^k\right)\frac{\mu_m^k L_p^m}{\sum_{m=1}^{M}\mu_m^k L_p^m}\right\} \tag{5.15}$$

The update equation (4.67) will be modified as

$$\frac{\partial J}{\partial L_p^m} = \sum_{k=1}^{K}\left[\gamma_k\left(y_p^k - d_p^k\right)\frac{\mu_m^k}{\left[\sum_{m=1}^{M}\mu_m^k L_p^m\right]}\left\{y_p^m - y_p^k\right\}\right] \tag{5.16}$$

The update equation (4.72) becomes as

$$\frac{\partial J}{\partial c_i^m} = \sum_{k=1}^{K}\left[\gamma_k\left\{\sum_{p=1}^{P}\left(y_p^k - d_p^k\right)\frac{L_{pl}\left[y_p^m - y_p^k\right]}{\sum_{m=1}^{M}\mu_m^k L_p^m}\right\}\mu_m^k\frac{\left(x_i^k - c_i^m\right)}{{b_i^m}^2}\right] \tag{5.17}$$

The update equation (4.76) will become as

$$\frac{\partial J}{\partial b_i^m} = \sum_{k=1}^{K}\left[\gamma_k\left\{\sum_{p=1}^{P}\left(y_p^k - d_p^k\right)\frac{L_{pl}\left[y_p^m - y_p^k\right]}{\sum_{m=1}^{M}\mu_m^k L_p^m}\right\}\mu_m^k\frac{\left(x_i^k - c_i^m\right)^2}{{b_i^m}^3}\right] \tag{5.18}$$

As can be seen from above update equations, only small changes are made to the original error backpropagation learning algorithm for the proposed networks. Since the average sum-squared errors are used to calculate unequal learning factors, this method should be used only in the epoch update mode. Other techniques, such as fast convergence techniques in MLP or FLS, can be modified and used along with this method. As mentioned earlier, after this training method the networks will have the same structures and signal processing capabilities as the original

networks trained with standard error back propagation. These networks trained with this modified error backpropagation are different only the way they train and learn to solve classification problems with unequal learning, unequal number of training data and different classification costs. This training method can be viewed as a training method combined with human knowledge, such as different classification costs, human satisfaction. All networks trained with error backpropagation in classification applications can use this method. If the importance of each class, i.e., classification costs, can be quantified, this method can be used effectively.

Since the total learning factor can be decomposed into three different learning factors, use of only one or any combination of two learning factors can also be done, something that is problem dependent. In this research all three learning factors are used as the total learning factor. This modified error backpropagation will be used to train the proposed networks and compared with the standard error backpropagation and conventional classifier, *k*-NN, in the next chapter.

Chapter 6

Training and Testing Results

All proposed classifiers are trained, tested, and compared with the classical K-nearest neighbor. Both unequal learning factor and equal training data techniques are used and compared for all proposed networks. There are two sets of data: iris and vehicle acoustic data. Iris data is used in the equal training data study. The vehicle acoustic data is thoroughly used and compared in the unequal learning factor study. Results of the unequal learning factor technique improve over the equal training data method.

6.1 Computational Expense of Different Classifiers

Different classifiers have different advantages and disadvantages. One criterion widely used to compare different classifiers are their different computational expenses. Some classifiers require less or no training computational expense, but they may require more computational expense during classification. All classifiers in this research were trained off-line, that is, training data were collected and processed off-line. Thus, the computational expense during training is less constrained than an on-line training. In most classification work classifiers are trained off-line mode rather than on-line. In this research all classifiers were trained in an off-line mode. The computational expense during training and operation are compared. The number of flops required is used as a computational expense measurement.

During training phase a number of flops required in 1 epoch of training is considered. The epoch update mode means that network parameters are updated after all patterns are presented and their errors are calculated. The results are shown in Table 6.1.

Table 6.1 Computational expense during single-epoch of training of each classifier

Classifiers	4 inputs & 3 output (flops)	30 inputs & 5 outputs (flops)
k-NN	0	0
MLP with Learning factor (2 & 12)	19589	8282076
MLP without Learning factor (2 & 12)	16627	8123326
MLP with Learning factor (3 & 16)	24979	10876960
MLP without Learning factor (3 & 16)	22025	10718354
AFLS-BOA with Learning factor	43954	39101381
AFLS-BOA without Learning factor	40975	38942196
AFLS-CA with Learning factor	41839	38464181
AFLS-CA without Learning factor	38860	38304996

In a particular 4 input and 3 output problem there are 114 patterns. This data set is the "iris data" used in training all proposed classifiers. The training data set consists of approximately 75% of each class data. In another 30 input, 5 output problem there are 3979 patterns. They are the feature vectors of the vehicle acoustic data used in training the proposed classifiers. The training data set also consists of 75% from each class.

A learning factor is used with the modified error backpropagation for "unequal learning" problems. Otherwise, classifiers without unequal learning problems use a standard error backpropagation training.

The first two MLPs in Table 6.1 are designed to be used for each problem. The MLPs in the next two rows have the same number of hidden nodes as the number of rules of the AFLSs, that is 3 and 16 rules for each problem, respectively.

The AFLS-BOAs require more computation than other networks during training, requiring almost four times greater than MLPs with the 30 input and 5 output problem. The unequal learning method adds additional computation.

In the forward computation all classifiers are compared by using one pattern from each problem as shown in Table 6.2.

Table 6.2 Computational expense of trained classifiers

Classifiers	4 inputs & 3 output (flops)	30 inputs & 5 outputs (flops)
k-NN	2742	716230
MLP with Learning factor (2 & 12)	58	942
MLP without Learning factor (2 & 12)	58	942
MLP with Learning factor (3 & 16)	78	1246
MLP without Learning factor (3 & 16)	78	1246
AFLS-BOA with Learning factor	141	4229
AFLS-BOA without Learning factor	141	4229
AFLS-CA with Learning factor	132	4149
AFLS-CA without Learning factor	132	4149

The *k*-NN classifier requires no training, but requires many computations for classification. The MLPs still require fewer computations than their AFLSs counterpart. Both classifiers trained with standard error backpropagation and with unequal learning error backpropagation require the same computations in classification stage. All AFLS classifiers require comparable computations. The AFLS-BOAs always require more than AFLS-CAs with the same number of rules used.

6.2 Iris Data

Iris data is a classical data set used as a benchmark classification problem. There are three classes: Setosa, Versicolor, and Virginica. Four features were extracted: sepal length, sepal width, petal length, and petal width. There are 50 data points from each class. Class 1 is linearly separated from the others. Class 2 and class 3 are overlapped.

In this study the iris data is used for two purposes: equal number in each class and for the unequal number problem. In the first problem each class is divided into two sets of data, training and testing. Most researchers use the whole data set as both training and testing data. In this study the data is divided into 75% training and 25% testing from each class. Both training and testing data are randomly picked from the entire data set. In the second problem classes 1 and 2 are combined as one class and class 3 as the other class. In this way one class has twice the number of data than the other class. Again, the data is divided into 75% training and 25% testing from each class.

6.2.1 Iris Data with Equal Number of Training Data

This problem serves as a benchmark problem for all classifiers. The problem is close to the ideal training of all classifiers because of the equal number of training data; but, even so, the complexity of the problem still causes an unequal learning problem.

The initial MLP weights are small random numbers from a uniform distribution from -0.5 to 0.5. This initialization method is used for all MLPs used in this research. The initial parameters of both AFLSs are initialized by the training data picked randomly. The input feature vector is used as the initial centers of the antecedent part. The spread parameters are set to be equal at 0.275. The centers of the consequent part are initialized by the corresponding desired output. In classification problems the desired outputs are normally in binary form. In the AFLS

BOA the spread parameters of the consequent part are initialized by a constant, 0.75. In this way, the initial rules have the same spread in the consequent part of IF-THEN rules. All parameters of the networks are updated during training phase.

The bipolar sigmoid function is used in the hidden layer and the extended sigmoid function is used in the output layer of all MLP networks. In AFLS the Gaussian shape function is used as a membership function. This function is also used in the AFLS-CA throughout this research. The classification costs are given the values as shown in Table 6.3.

Table 6.3 Classification costs in Iris data with the same number of training data

		CLASSIFIED		
		Class #1	Class #2	Class #3
TRUTH	Class #1	1	1.3	1.3
	Class #2	1.3	1	1.2
	Class #3	1.3	1.2	1

The numbers of MLP hidden nodes are determined by an increment method. The network starts with a small number of hidden nodes, then it is incremented until the outputs of the networks saturated. The final number of hidden nodes is determined by the smallest network giving relatively the same results as a bigger one. For example, the MLP used in this problem is selected to have only two hidden nodes since it is the smallest size that gives the same results as those of the larger MLPs.

With the AFLS the numbers of rules are determined by a similar increment method procedure. In this problem there are 3 rules required as shown in Table 6.4. The data used in this process is kept the same for all networks. The learning rates and momentum terms were also tuned during this process for all networks.

Table 6.4 Hidden nodes determination of MLP in Iris data with equal training data

# Hidden nodes	2	3	4	5	6
# Epochs	500	500	500	500	500
%Correct in Class 1	100.00	100.00	100.00	100.00	100.00
%Correct in Class 2	100.00	100.00	100.00	100.00	100.00
%Correct in Class 3	91.67	91.67	91.67	91.67	91.67
Total % Correct	97.22	97.22	97.22	97.22	97.22
Average Costs	1.0056	1.0056	1.0056	1.0056	1.0056

The average cost used to measure classifier's performance is defined as:

$$AverageCost = \frac{\sum_{c=1}^{C}\left[\left(\sum_{t=1}^{C}\gamma_{ct}N_{ct}\right)\Big/N_c\right]}{C} \qquad (6.1)$$

where γ_{ct} : classification costs

N_c : number of data in each class

N_{ct} : Number of data in class *c* and classified as class *t*.

C : number of classes.

This average cost has not only correct classification but also misclassification information. The average classification costs will be used as performance indices of all classifiers.

Table 6.5 Number of rules determination of BOA in Iris data with equal training data

# Rules	2	3	4	5	6
# Epochs	300	300	300	300	300
%Correct in Class 1	33.33	100.00	100.00	100.00	100.00
%Correct in Class 2	100.00	100.00	100.00	100.00	100.00
%Correct in Class 3	83.33	83.33	83.33	83.33	83.33
Total % Correct	72.22	94.44	94.44	94.44	94.44
Average Costs	1.0833	1.0111	1.0111	1.0111	1.0111

With the *k*-NN classifiers the number of nearest neighbors, *k*, is also determined in the same incremental manner. The appropriate number of *k* for this problem is 3 as shown in Table 6.5. There is no training expense with *k*-NN classifiers, but they require many computations for classification, as well as a large storage for its data bank.

Table 6.6 Determination of *k* of *k*-NN in Iris data with equal training data

# *k*	3	5	7	9	11
%Correct in Class 1	100.00	100.00	100.00	100.00	100.00
%Correct in Class 2	100.00	100.00	100.00	100.00	100.00
%Correct in Class 3	91.67	91.67	91.67	91.67	91.67
Total % Correct	97.22	97.22	97.22	97.22	97.22
Average Costs	1.0056	1.0056	1.0056	1.0056	1.0056

Table 6.7 Comparison of the BEST performance of each classifier in Iris data with equal training data

	k-NN	MLP w/o	MLP	CA w/o	CA	BOA w/o	BOA
# Hidden Nodes or # *k* or # Rules	3	2	2	3	3	3	3
# Epoch in Training	0	500	500	200	200	200	200
%Correct in Class 1	100.00	100.00	100.00	100.00	100.00	100.00	100.00
%Correct in Class 2	100.00	100.00	100.00	100.00	100.00	100.00	100.00
%Correct in Class 3	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Total % Correct	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Average Costs	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Once the appropriate number of hidden nodes, number of rules, and number of *k* are determined, all classifiers used are compared directly. The AFLS-CA will use the same number of rules as in the AFLS-BOA.

Table 6.8 Comparison of the MEDIAN performance of each classifier in Iris data with equal training data

	k-NN	MLP w/o	MLP	CA w/o	CA	BOA w/o	BOA
# Hidden Nodes or # *k* or # Rules	3	2	2	3	3	3	3
# Epoch in Training	0	500	500	200	200	200	200
%Correct in Class 1	100.00	100.00	100.00	100.00	100.00	100.00	100.00
%Correct in Class 2	100.00	91.67	100.00	100.00	100.00	100.00	100.00
%Correct in Class 3	91.67	91.67	91.67	91.67	91.67	91.67	91.67
Total % Correct	97.22	94.44	97.22	97.22	97.22	97.22	97.22
Average Costs	1.0056	1.0111	1.0056	1.0056	1.0056	1.0056	1.0056

There are 9 classifiers with 9 sets of training and testing data. The best and median performances are selected from these 9 classifiers as a basis for comparison. Classifiers labeled "w/o" are trained with standard error backprogation and an equal number of data from each class. Classifiers without this label are trained with the modified error backpropagation. All classifiers can solve this classification problem quite easily. There is no difference between equal numbers of training data and the unequal learning method.

6.2.2 Iris Data with Two Unequal Training Data

In this problem, Classes 1 and 2 are combined as one class and Class 3 is another class. The purpose of this problem is to have unequal training data. Again, the number of hidden nodes of MLP is determined. In AFLS the number of rules required is also determined with the same procedure as in the previous problem. The final number of hidden nodes is two. There are 5 rules required in this problem.

Table 6.9 Classification costs used in Iris two-class problem

		CLASSIFIED	
		Class 1	Class 2
TRUTH	Class 1	1	1.3
	Class 2	1.3	1

Table 6.10 Hidden nodes determination of MLP in Iris data with unequal training data

# Hidden nodes	2	3	4	5	6
# Epochs in training	500	500	500	500	500
%Correct in Class 1	96.00	96.00	96.00	96.00	96.00
%Correct in Class 2	100.00	100.00	100.00	100.00	100.00
Total % Correct	97.30	97.30	97.30	97.30	97.30
Average Costs	1.0060	1.0060	1.0060	1.0060	1.0060

Table 6.11 Number of rules determination of BOA in Iris data with unequal training data

# Rules	2	3	4	5	6
# Epochs in training	300	300	300	300	300
%Correct in Class 1	92.00	96.00	96.00	100.00	100.00
%Correct in Class 2	100.00	100.00	100.00	100.00	91.67
Total % Correct	94.60	97.30	97.30	100.00	97.30
Average Costs	1.0120	1.0060	1.0060	1.0000	1.0125

Table 6.12 Determination of *k* of *k*-NN in Iris data with unequal training data

# *k*	3	5	7	9	11
%Correct in Class 1	96.00	96.00	96.00	96.00	96.00
%Correct in Class 2	91.67	91.67	91.67	91.67	91.67
Total % Correct	94.59	94.59	94.59	94.59	94.59
Average Costs	1.0185	1.0185	1.0185	1.0185	1.0185

Table 6.13 Comparison of the BEST performance of each classifier in Iris data with unequal training data

	KNN	MLP w/o	MLP	CA w/o	CA	BOA w/o	BOA
# Hidden nodes or # *k* or # Rules	3	2	2	5	5	5	5
# Epochs in training	0	500	500	200	200	200	200
%Correct in Class 1	100.00	100.00	100.00	100.00	100.00	100.00	100.00
%Correct in Class 2	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Total % Correct	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Average Costs	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Table 6.14 Comparison of the MEDIAN performance of each classifier in Iris data with unequal training data

	k-NN	MLP w/o	MLP	CA w/o	CA	BOA w/o	BOA
# Hidden nodes or # *k* or # Rules	3	2	2	5	5	5	5
# Epochs in training	0	500	500	200	200	200	200
%Correct in Class 1	100.00	100.00	100.00	100.00	96.00	100.00	100.00
%Correct in Class 2	83.33	91.67	91.67	91.67	100.00	91.67	91.67
Total % Correct	94.59	97.30	97.30	97.30	97.30	97.30	97.30
Average Costs	1.0125	1.0125	1.0125	1.0125	1.0060	1.0125	1.0125

Again, all classifiers can solve this classification problem quite easily. There is not much difference between different methods because of a lack in complexity of a problem.

6.3 Vehicle Acoustic Data

Vehicle acoustic data were collected, processed, and extracted from raw acoustic data as explained in Chapters 2 and 3. There are 30 extracted features for each vehicle. There are five classes, passenger cars, pickups and vans, two-axle six-tire trucks, three-axle single unit trucks, and five axle single trailer trucks. There are data representing 2440 passenger cars, 1007 pickups or vans, 587 two-axle six-tire trucks, 309 three-axle single unit trucks, and 963 five axle single trailer trucks in this data set. In a four class study, Classes 1 and 2 are combined as a small vehicle class with no change in the remaining classes. In a two class study, Class 1 and 2 are combined as a small vehicle class and Classes 3 to 5 are combined as a large vehicle class. Training data is randomly picked, 75% of each class from the entire data, and the remaining data is used as the testing data. In this way the testing data is different from the training data. Since the numbers of available data are different for each class, the unequal learning problem is encountered. The unequal learning technique is used to train proposed classifiers and the results are compared with other approaches.

6.3.1 Five-Class Problem

The five class problem is the toughest problem in this research. Each class consists of a variety of vehicles. For example, the two axle six tire class includes all small trucks having two axles and six tires. This class overlaps with other classes, e.g., pickups, heavy duty trucks. Without any modification, the learning of one classifier is shown in Figures 6.1, 6.3, and 6.5. The classifier learned well in certain classes, and it would definitely classify well in those classes. With the use of an unequal learning factor a classifier can learn to classify all classes well. Examples of networks using unequal learning factors are shown in Figures 6.2, 6.4, and 6.6.

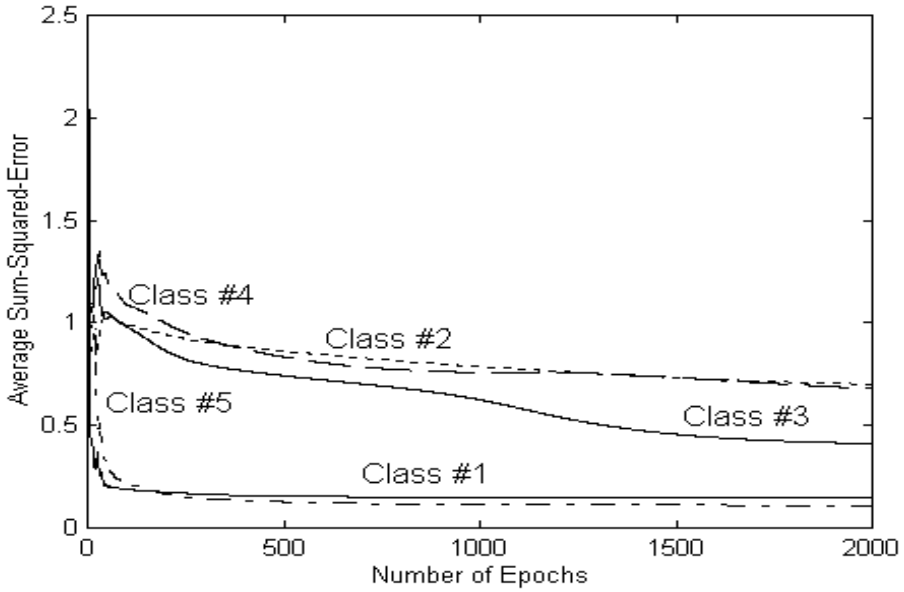

Figure 6.1 Average sum-squared-error of each class WITHOUT learning factor of MLP

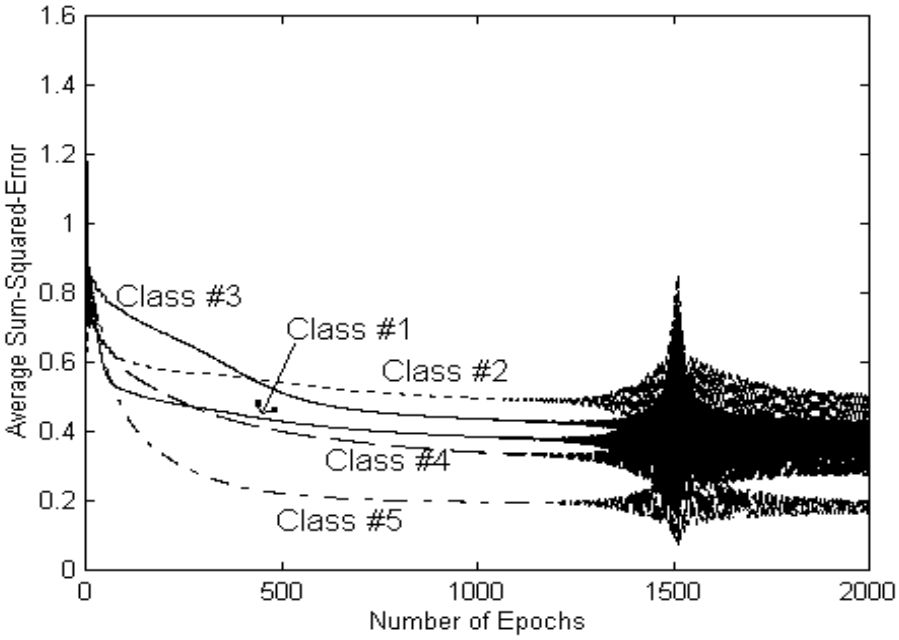

Figure 6.2 Average sum-squared-error of each class WITH learning factor of MLP

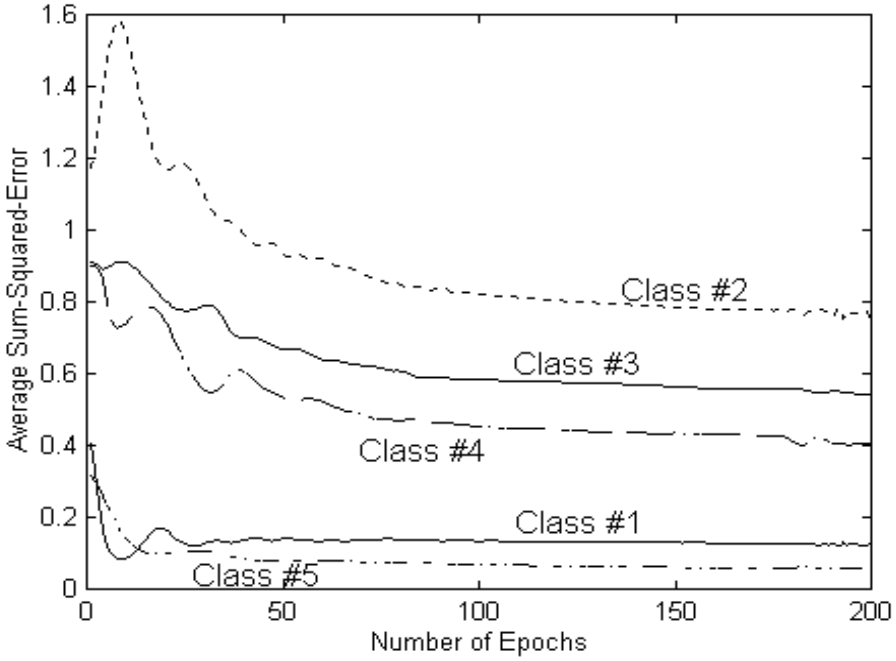

Figure 6.3 Average sum-squared-error of each class WITHOUT learning factor of AFLS-BOA

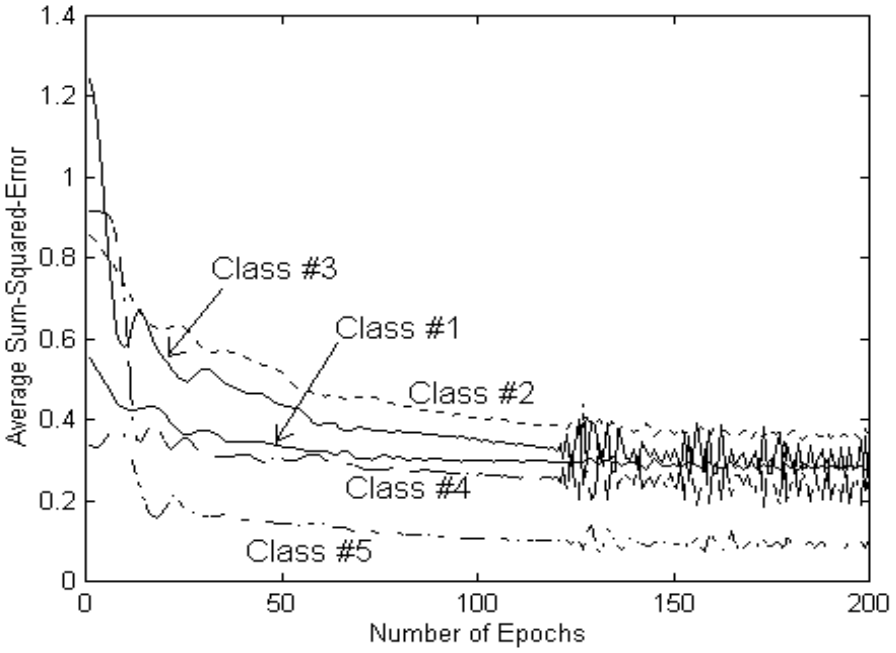

Figure 6.4 Average sum-squared-error of each class WITH learning factor of AFLS-BOA

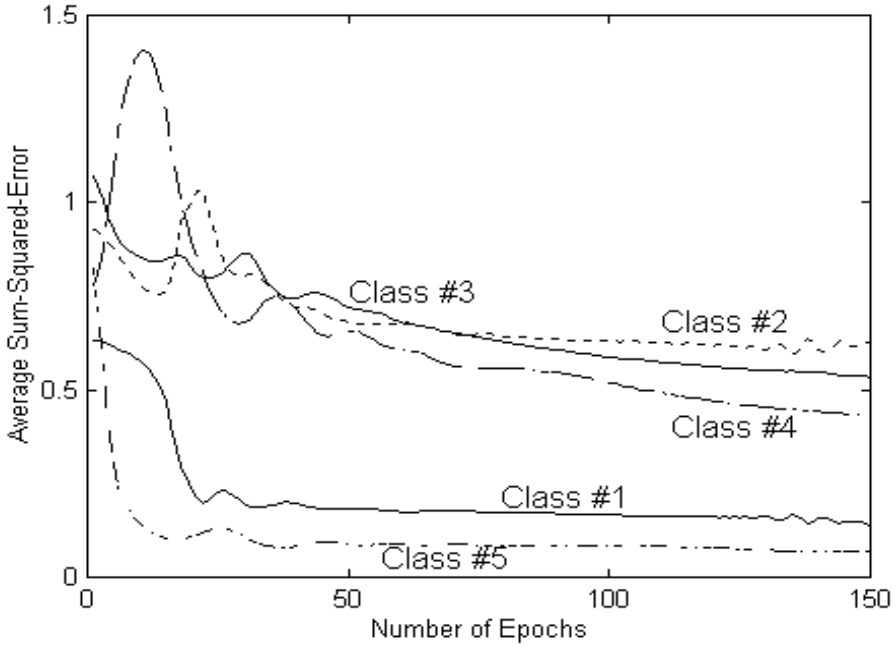

Figure 6.5 Average sum-squared-error of each class WITHOUT learning factor of AFLS-CA

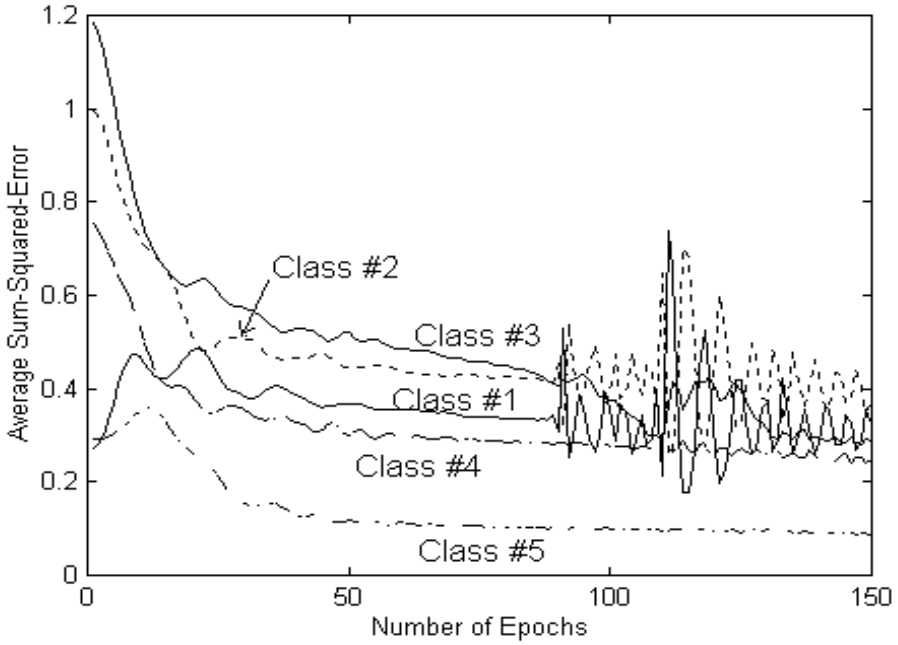

Figure 6.6 Average sum-squared-error of each class WITH learning factor of AFLS-CA

The number of hidden nodes required in this problem is determined by using the same method as in previous problems. The results are shown in Table 6.16. The number of rules in AFLS is also determined with the same procedure. The results are shown in Table 6.17. Number of hidden nodes is set to 12 nodes for this five class problem, compared with approximately 16 rules required.

The *k* value in *k*-NN classifier is also determined and the results are shown in Table 6.18. The value of *k* is set to 5 for this problem.

Table 6.15 Classification costs in five-class problem

		CLASSIFIED				
		Class 1	Class 2	Class 3	Class 4	Class 5
TRUTH	Class 1	1	1.2	1.3	1.3	1.3
	Class 2	1.2	1	1.2	1.3	1.3
	Class 3	1.3	1.2	1	1.2	1.3
	Class 4	1.3	1.3	1.2	1	1.2
	Class 5	1.3	1.3	1.3	1.2	1

In percentage difference comparison Tables the values are defined as:

Pct Difference = Pct of BOA - Pct of a classifier. (6.2)

For example, Pct Difference = 4.43 means that BOA classifier is better than that classifier by 4.43%. Pct Difference = -2.46 means that that classifier is better than BOA classifier by 2.46%. In the average cost comparison the percentage difference of average costs id defined by

$$\%DifferenceInaverageCost = \left[\frac{AverageCostofthatClassifier - AverageCostofBOA}{AverageCostofBOA}\right] x100 \quad (6.3)$$

Table 6.16 Hidden nodes determination of MLP in five-class problem

# hidden nodes	# Epochs	% correct in Class 1	% Correct in Class 2	% Correct in Class 3	% Correct in Class 4	% Correct in Class 5	Total % Correct	Average Costs
5	2000	78.52	53.57	70.75	72.73	86.72	74.08	1.0685
7	2000	79.02	56.35	68.71	72.73	86.72	74.60	1.0677
9	2000	78.52	57.54	69.39	75.32	88.80	75.21	1.0650
11	2000	79.84	58.73	72.79	74.03	88.38	76.26	1.0630
13	2000	81.48	54.37	72.11	75.32	88.38	76.19	1.0642
15	2000	80.33	57.14	70.07	76.62	88.80	76.11	1.0633
17	2000	80.00	57.94	73.47	72.73	89.63	76.41	1.0630
19	2000	80.00	57.14	70.07	76.62	89.21	76.04	1.0633
21	2000	80.16	58.33	72.11	74.03	89.21	76.41	1.0628

Table 6.17 Number of rules determination of BOA in five-class problem

# Rules	# Epochs	% correct in Class 1	% Correct in Class 2	% Correct in Class 3	% Correct in Class 4	% Correct in Class 5	Total % Correct	Average Costs
5	200	84.26	50.79	71.43	72.73	90.46	76.94	1.0653
7	200	80.33	63.49	61.22	41.56	90.87	74.68	1.0775
9	200	78.69	59.13	70.75	67.53	92.53	75.96	1.0648
11	200	79.67	67.06	68.03	72.73	92.53	77.92	1.0590
13	200	81.80	64.68	69.39	72.73	90.87	78.30	1.0594
15	200	80.33	65.08	74.83	72.73	92.53	78.60	1.0568
17	200	80.66	62.70	72.79	62.34	92.12	77.39	1.0636
19	200	81.15	58.33	72.79	61.04	91.29	76.56	1.0667

Table 6.18 Determination of *k* of *k*-NN in five-class problem

k	%Correct in Class 1	%Correct in Class 2	%Correct in Class 3	%Correct in Class 4	%Correct in Class 5	Total % Correct	Average Costs
5	83.28	45.63	47.62	44.16	91.29	71.36	1.0911
7	83.28	45.63	47.62	44.16	91.29	71.36	1.0911
9	83.28	45.63	47.62	44.16	91.29	71.36	1.0911
11	83.28	45.63	47.62	44.16	91.29	71.36	1.0911
13	83.28	45.63	47.62	44.16	91.29	71.36	1.0911

The results show that the AFLS-BOAs are the most consistent and have better performance than the others, as shown in Tables 6.19, 6.20, 6.21, and 6.22. The classical *k*-NN has the worst performance. The MLP and AFLS-CA trained with modified error backpropagation have more consistency than MLP and AFLS-CA trained with standard error backpropagation. In general, the networks trained with the modified error backpropagation improve the classification rates for each class.

Table 6.19 Comparison of the BEST performance of each classifier in five-class problem

	k-NN	MLP w/o	MLP	CA w/o	CA	BOA w/o	BOA
#Hidden nodes or # *k* or # Rules	5	12	12	16	16	16	16
# Epochs in training	0	2000	2000	200	200	200	200
%Correct in Class 1	81.64	74.75	77.38	80.00	74.43	83.11	79.18
%Correct in Class 2	45.24	67.46	63.49	61.90	76.19	59.13	66.27
%Correct in Class 3	55.78	72.11	68.71	72.11	70.75	75.51	70.07
%Correct in Class 4	62.34	81.82	80.52	70.13	74.03	85.71	80.52
%Correct in Class 5	95.02	90.46	91.70	91.70	93.78	93.78	95.02
Total % Correct	73.17	76.34	76.56	77.24	77.84	79.80	78.67
Average Costs	1.0792	1.0565	1.0587	1.0612	1.0540	1.0531	1.0544

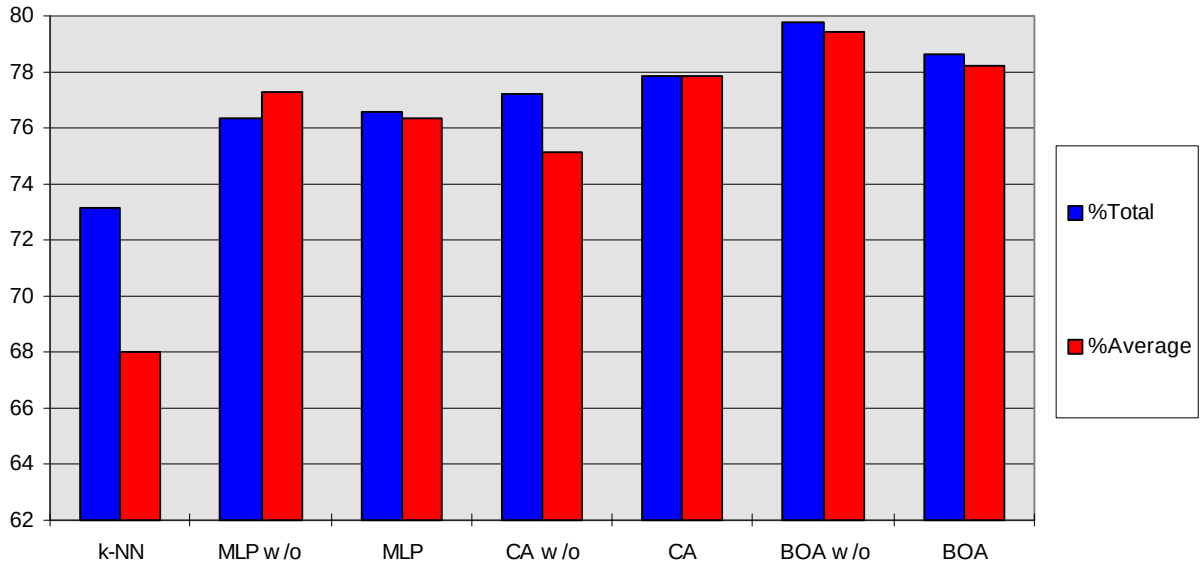

Figure 6.7 Percentage of total and average correct of the BEST performance in five-class problem

Table 6.20 Percentage difference of the BEST performance from BOA in five-class problem

	k-NN	MLP w/o	MLP	CA w/o	CA	BOA w/o	BOA
%Diff. in Class 1	-2.46	4.43	1.80	-0.82	4.75	-3.93	0.00
%Diff. in Class 2	21.03	-1.19	2.78	4.37	-9.92	7.14	0.00
%Diff. in Class 3	14.29	-2.04	1.36	-2.04	-0.68	-5.44	0.00
%Diff. in Class 4	18.18	-1.30	0.00	10.39	6.49	-5.19	0.00
%Diff. in Class 5	0.00	4.56	3.32	3.32	1.24	1.24	0.00
%Diff. Total Correct	5.50	2.33	2.11	1.43	0.83	-1.13	0.00
Average Costs	2.35	0.20	0.41	0.64	-0.04	-0.12	0.00

Table 6.21 Comparison of the MEDIAN performance of each classifier in five-class problem

	k-NN	MLP w/o	MLP	CA w/o	CA	BOA w/o	BOA
#Hidden nodes or # *k* or # Rules	5	12	12	16	16	16	16
# Epochs in training	0	2000	2000	200	200	200	200
%Correct in Class 1	81.48	74.26	76.89	74.10	79.51	78.69	79.67
%Correct in Class 2	42.86	65.87	60.71	62.70	71.03	67.06	67.86
%Correct in Class 3	65.31	70.07	75.51	59.86	70.75	69.39	71.43
%Correct in Class 4	53.25	71.43	79.22	77.92	63.64	79.22	67.53
%Correct in Class 5	92.95	89.63	86.31	86.72	91.29	92.95	92.12
Total % Correct	72.80	74.83	75.51	72.87	78.15	78.07	78.07
Average Costs	1.0826	1.0635	1.0602	1.0680	1.0601	1.0559	1.0588

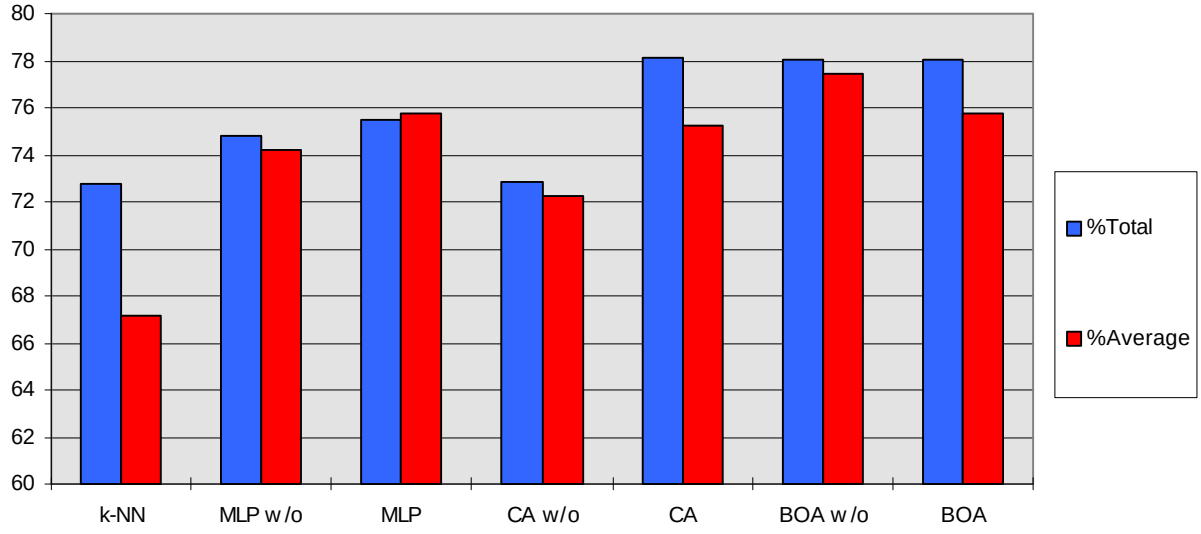

Figure 6.8 Percentage of total and average correct of the MEDIAN performance in five-class problem

Table 6.22 Percentage difference of the MEDIAN performance from BOA in five-class problem

	k-NN	MLP w/o	MLP	CA w/o	CA	BOA w/o	BOA
%Diff. in Class 1	-1.81	5.41	2.78	5.57	0.16	0.98	0.00
%Diff. in Class 2	25.00	1.99	7.15	5.16	-3.17	0.80	0.00
%Diff. in Class 3	6.12	1.36	-4.08	11.57	0.68	2.04	0.00
%Diff. in Class 4	14.28	-3.90	-11.69	-10.39	3.89	-11.69	0.00
%Diff. in Class 5	-0.83	2.49	5.81	5.40	0.83	-0.83	0.00
%Diff. Total Correct	5.27	3.24	2.56	5.20	-0.08	0.00	0.00
Average Costs	2.25	0.44	0.13	0.87	0.12	-0.27	0.00

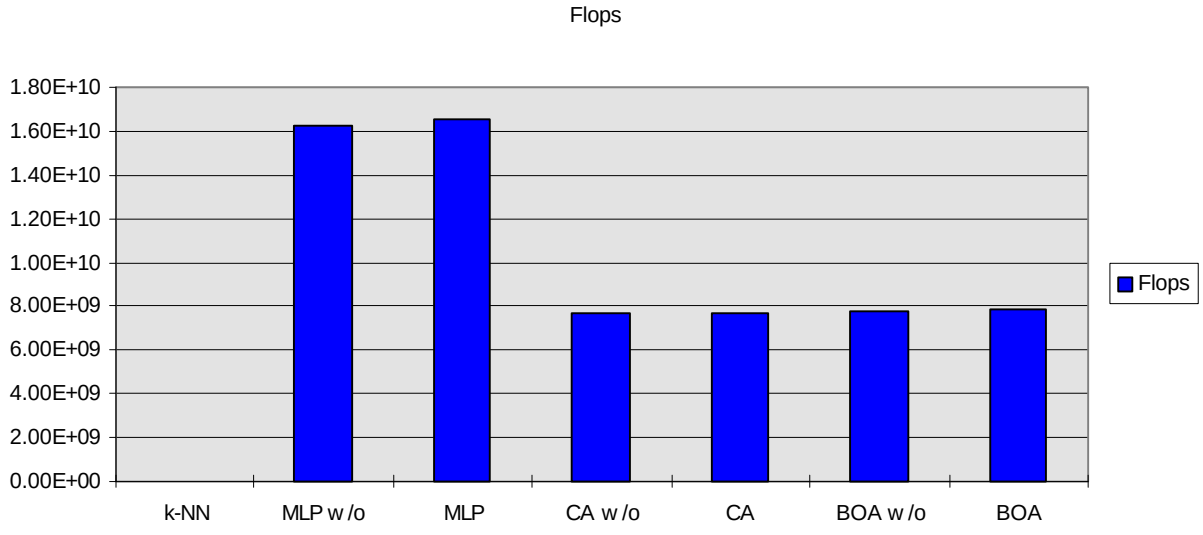

Figure 6.9 Computational expense during training (flops) in vehicle 5-class problem

6.3.2 Four-Class Problem

In a four class problem, Classes 1 and 2 are combined as one small vehicle class. The remaining classes are the same as in previous problem. The number of hidden nodes, number of

rules and the *k* value are determined with the same procedures as in the previous problem. The number of hidden nodes is set at 12, number of rules is 14 and number of *k* is 5.

In this problem the numbers of data are 3447, 587, 309, and 963 for Class #1, #2, #3, #4, respectively. Class 1 "dominates" the other classes. Therefore, we can see clearly the differences between networks using the unequal learning factor and not using it. The networks using the unequal learning factor improve correct classification rates. The more mismatch of the number of data, the more effect the unequal learning factor will have. Again, the AFLS-BOAs have better results than the other classifiers.

Table 6.23 Classification costs used in four-class problem

		CLASSIFIED			
		Class 1	Class 2	Class 3	Class 4
TRUTH	Class 1	1	1.3	1.3	1.3
	Class 2	1.3	1	1.2	1.3
	Class 3	1.3	1.2	1	1.2
	Class 4	1.3	1.3	1.2	1

Table 6.24 Comparison of the BEST performance of each classifier in four-class problem

	k-NN	MLP w/o	MLP	CA w/o	CA	BOA w/o	BOA
#Hidden nodes or # *k* or # Rules	5	12	12	14	14	14	14
# Epochs in training	0	2000	2000	200	200	200	200
%Correct in Class 1	97.33	93.74	93.62	90.14	93.85	93.39	95.59
%Correct in Class 2	61.90	72.11	73.47	61.90	81.63	71.43	79.59
%Correct in Class 3	57.14	70.13	87.01	81.82	74.03	81.82	79.22
%Correct in Class 4	94.61	93.36	88.38	94.61	91.70	90.87	92.12
Total % Correct	90.58	89.90	90.05	87.34	90.96	89.83	92.24
Average Costs	1.0514	1.0405	1.0330	1.0410	1.0346	1.0362	1.0295

Table 6.25 Percentage difference of the BEST performance from BOA in four-class problem

	k-NN	MLP w/o	MLP	CA w/o	CA	BOA w/o	BOA
%Diff. in Class 1	-1.74	1.85	1.97	5.45	1.74	2.20	0.00
%Diff. in Class 2	17.69	7.48	6.12	17.69	-2.04	8.16	0.00
%Diff. in Class 3	22.08	9.09	-7.79	-2.60	5.19	-2.60	0.00
%Diff. in Class 4	-2.49	-1.24	3.74	-2.49	0.42	1.25	0.00
%Diff. Total Correct	1.66	2.34	2.19	4.90	1.28	2.41	0.00
Average Costs	2.13	1.07	0.34	1.12	0.50	0.65	0.00

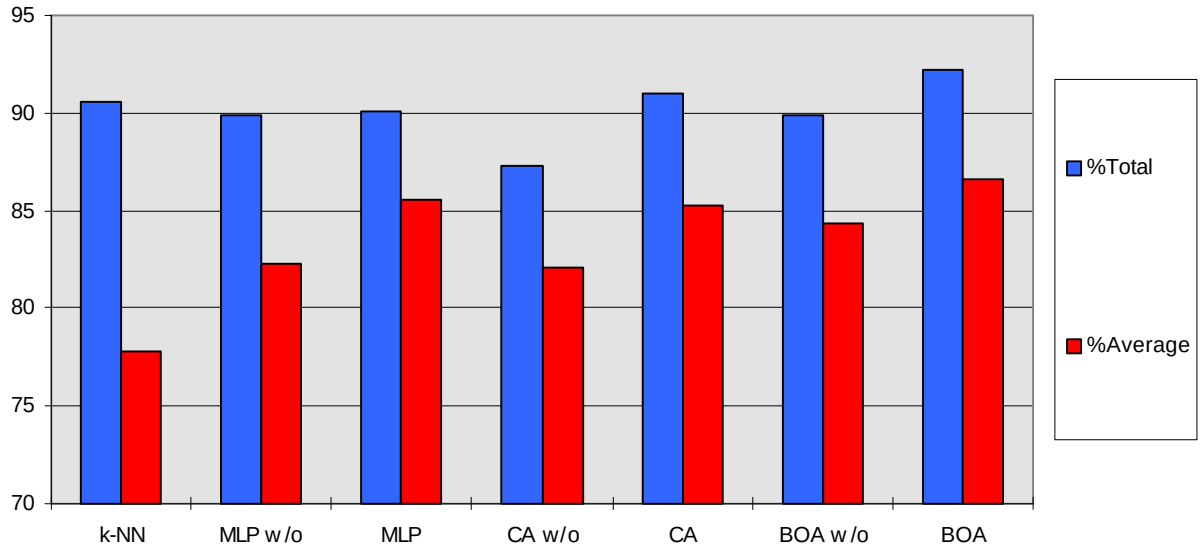

Figure 6.10 Percentage of total and average correct of the BEST performance in four-class problem

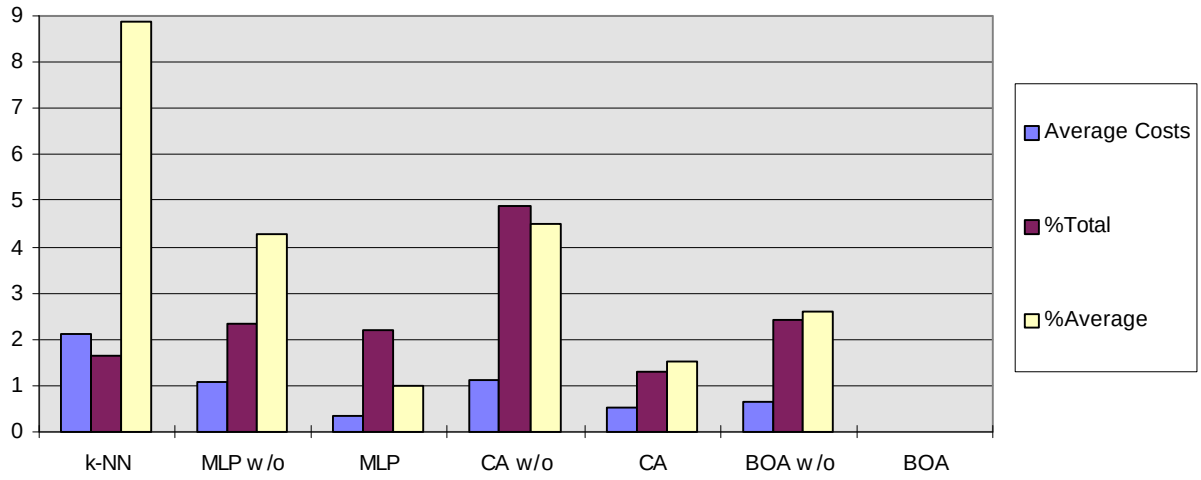

Figure 6.11 Percentage improvement of BOA over other classifiers for BEST performance in four-class problem

Table 6.26 Comparison of the MEDIAN performance of each classifier in four-class problem

	k-NN	MLP w/o	MLP	CA w/o	CA	BOA w/o	BOA
#Hidden nodes or # *k* or # Rules	5	12	12	14	14	14	14
# Epochs in training	0	2000	2000	200	200	200	200
%Correct in Class 1	97.80	95.82	94.90	84.22	93.62	91.76	95.24
%Correct in Class 2	57.14	59.86	72.79	71.43	66.67	65.99	76.87
%Correct in Class 3	54.55	76.62	77.92	76.62	57.14	76.62	72.73
%Correct in Class 4	94.61	88.38	90.87	91.29	93.78	93.36	89.21
Total % Correct	90.20	89.37	90.73	83.65	88.55	88.32	90.81
Average Costs	1.0542	1.0456	1.0358	1.0446	1.0516	1.0413	1.0363

Table 6.27 Percentage difference of the MEDIAN performance from BOA in four-class problem

	k-NN	MLP w/o	MLP	CA w/o	CA	BOA w/o	BOA
%Diff. in Class 1	-2.56	-0.58	0.34	11.02	1.62	3.48	0.00
%Diff. in Class 2	19.73	17.01	4.08	5.44	10.20	10.88	0.00
%Diff. in Class 3	18.18	-3.89	-5.19	-3.89	15.59	-3.89	0.00
%Diff. in Class 4	-5.40	0.83	-1.66	-2.08	-4.57	-4.15	0.00
%Diff. Total Correct	0.61	1.44	0.08	7.16	2.26	2.49	0.00
Average Costs	1.73	0.90	-0.05	0.80	1.48	0.48	0.00

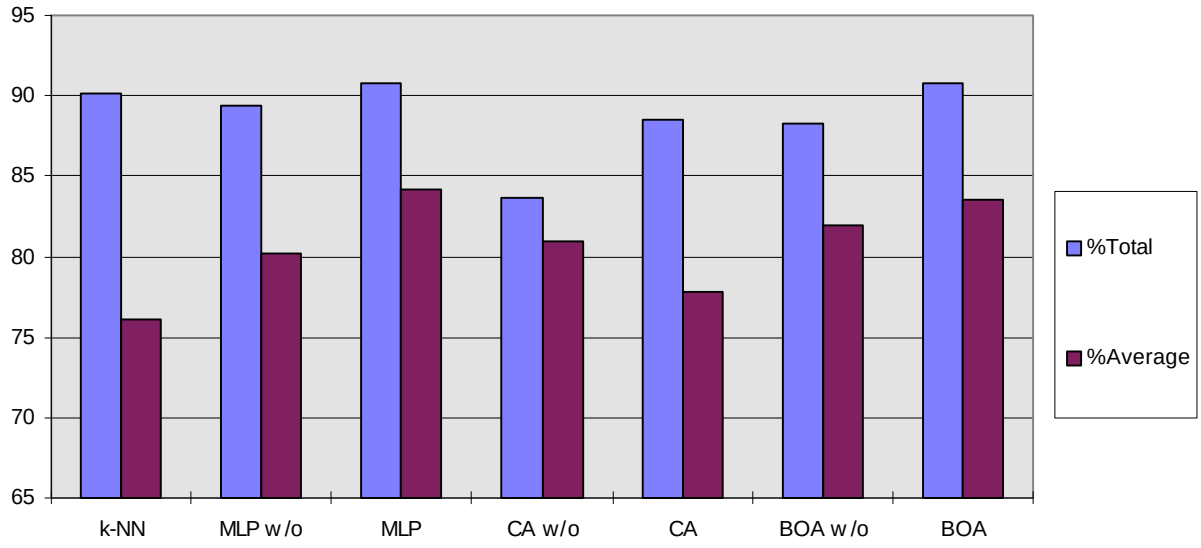

Figure 6.12 Percentage of total and average correct of the MEDIAN performance in four-class problem

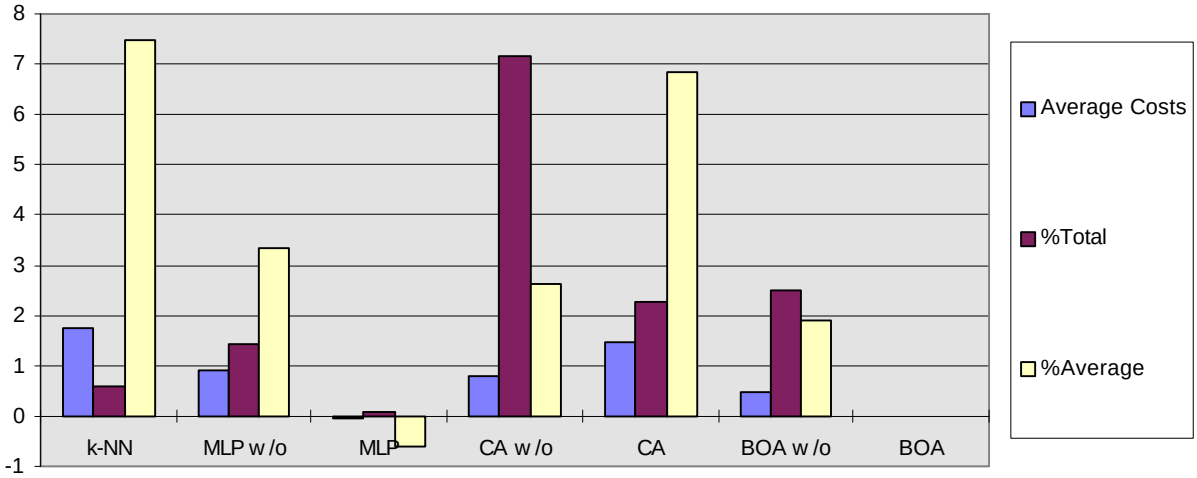

Figure 6.13 Percentage improvement of BOA over other classifiers for MEDIAN performance in four-class problem

6.3.3 Small versus Large Vehicle Classification

In this problem vehicles in Classes 1 and 2 are combined as a small vehicle class and Classes 3, 4, and 5 are combined as a large vehicle class. There will be 3447 in one class and 1859 in the other class. The number of data in each class is not heavily mismatched, therefore, the effect of unequal learning factor is not great, as the results show. Again the AFLS-BOAs have better classification performances than the other classifiers.

Table 6.28 Classification costs in a small versus large vehicle problem

		CLASSIFIED	
		Class 1	Class 2
TRUTH	Class 1	1	1.3
	Class 2	1.3	1

Table 6.29 Comparison of the BEST performance of each classifier in a small versus large vehicle problem

	k-NN	MLP w/o	MLP	CA w/o	CA	BOA w/o	BOA
#Hidden nodes or # *k* or # Rules	3	4	4	6	6	6	6
# Epochs in training	0	500	500	200	200	200	200
%Correct in Class 1	97.91	97.33	97.56	96.98	97.45	96.75	97.80
%Correct in Class 2	93.33	95.70	95.70	93.55	97.20	95.70	97.20
Total % Correct	96.31	96.76	96.91	95.78	97.36	96.38	97.59
Average Costs	1.0131	1.0105	1.0101	1.0142	1.0080	1.0113	1.0075

Table 6.30 Percentage difference of the BEST performance from BOA in a small versus large vehicle problem

	k-NN	MLP w/o	MLP	CA w/o	CA	BOA w/o	BOA
%Diff. in Class 1	-0.11	0.47	0.24	0.82	0.35	1.05	0.00
%Diff. in Class 2	3.87	1.50	1.50	3.65	0.00	1.50	0.00
%Diff. Total Correct	1.28	0.83	0.68	1.81	0.23	1.21	0.00
Average Costs	0.56	0.30	0.26	0.67	0.05	0.38	0.00

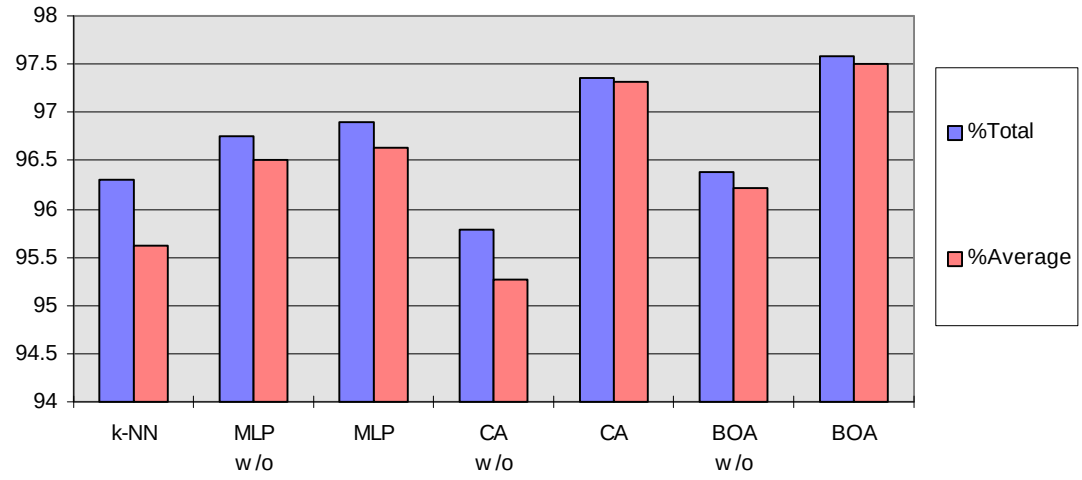

Figure 6.14 Percentage of total and average correct of the BEST performance in small vs. Large vehicle problem

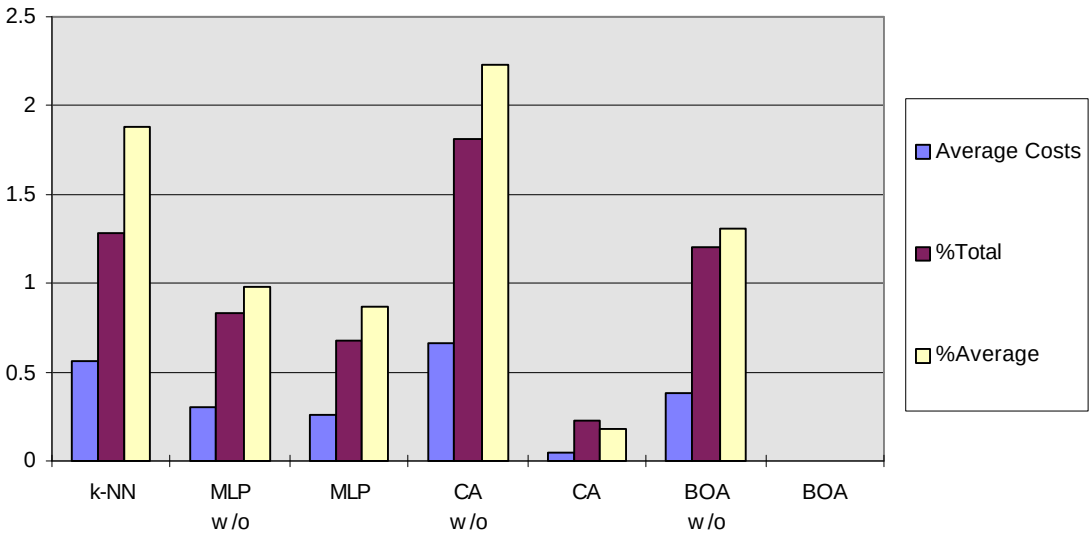

Figure 6.15 Percentage improvement of BOA over other classifiers for BEST performance in small vs. Large vehicle problem

Table 6.31 Comparison of the MEDIAN performance of each classifier in a small versus large vehicle problem

	k-NN	MLP w/o	MLP	CA w/o	CA	BOA w/o	BOA
#Hidden nodes or # *k* or # Rules	3	4	4	6	6	6	6
# Epochs in training	0	500	500	200	200	200	200
%Correct in Class 1	98.49	97.33	97.33	92.46	96.40	96.64	97.22
%Correct in Class 2	92.47	95.48	95.48	93.12	93.76	91.61	96.13
Total % Correct	96.38	96.68	96.68	92.69	95.48	94.88	96.83
Average Costs	1.0136	1.0108	1.0108	1.0216	1.0147	1.0176	1.0100

Table 6.32 Percentage difference of the MEDIAN performance from BOA in a small versus large vehicle problem

	k-NN	MLP w/o	MLP	CA w/o	CA	BOA w/o	BOA
%Diff. in Class 1	-1.27	-0.11	-0.11	4.76	0.82	0.58	0.00
%Diff. in Class 2	3.66	0.65	0.65	3.01	2.37	4.52	0.00
%Diff. Total Correct	0.45	0.15	0.15	4.14	1.35	1.95	0.00
Average Costs	0.36	0.08	0.08	1.15	0.47	0.75	0.00

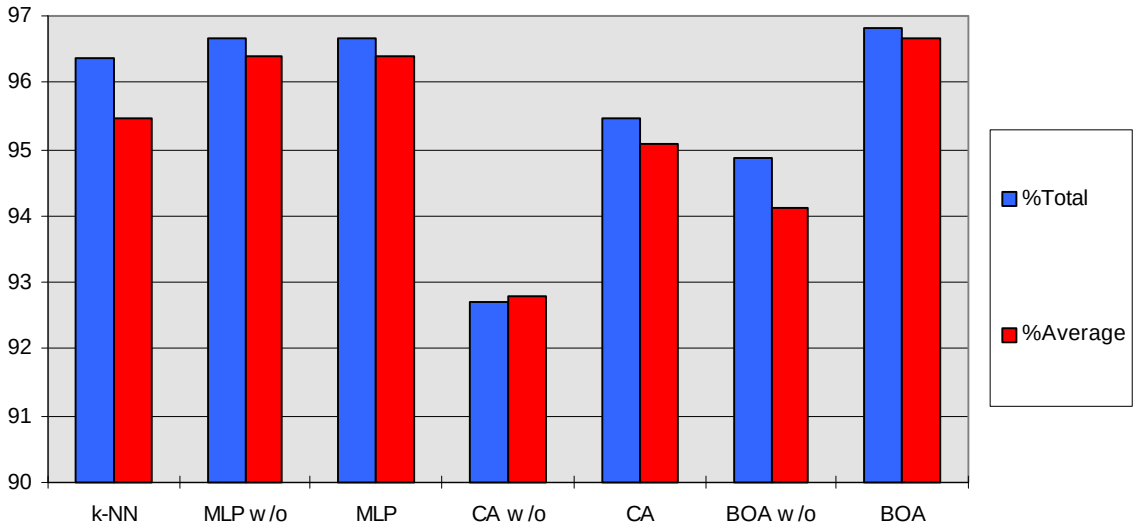

Figure 6.16 Percentage of total and average correct of the MEDIAN performance in small vs. Large vehicle problem

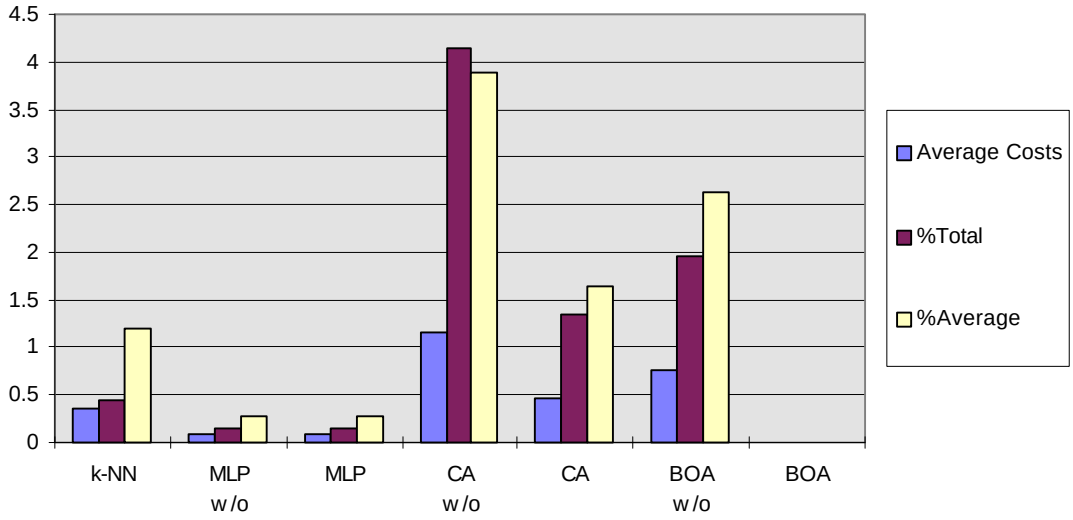

Figure 6.17 Percentage improvement of BOA over other classifiers for MEDIAN performance in small vs. Large vehicle problem

6.4 Dimension Reduction Study

The extracted features may contain redundant information. In this study, there are only 30 features, but in some case there may be more than 100 extracted features. Only significant features should be extracted. There are two methods used in this example problem; principal component analysis (PCA) and a heuristic approach. The PCA is considered the most effective method available. With the heuristic approach, a feature is eliminated if its distribution is small and there is minimal separation among classes.

Table 6.33 Comparison of the BEST performance of each classifier in five-class problem of dimension reduction study

	k-NN #1	KNN #2	MLP #1	MLP #2	CA #1	CA #1	BOA #1	BOA #2
#Hidden nodes or # *k* or # Rules	5	5	12	12	16	16	16	16
# Epochs in training	0	0	2000	2000	200	200	200	200
%Correct in Class 1	85.90	83.77	80.98	73.61	78.52	77.05	77.70	78.36
%Correct in Class 2	49.21	44.84	62.30	64.29	68.25	69.05	63.10	67.06
%Correct in Class 3	58.50	61.90	74.15	72.11	67.35	64.63	75.51	69.39
%Correct in Class 4	62.34	62.34	81.82	76.62	83.12	75.32	85.71	79.22
%Correct in Class 5	95.02	93.36	89.21	89.63	94.19	94.61	93.36	90.46
Total % Correct	76.19	74.45	78.22	74.76	78.45	77.24	78.00	77.47
Average Costs	1.0739	1.0775	1.0560	1.0609	1.0538	1.0586	1.0524	1.0570

Table 6.34 Percentage difference of the BEST performance from BOA (30 features) in five-class problem of dimension reduction study

	k-NN #1	KNN #2	MLP #1	MLP #2	CA #1	CA #1	BOA #1	BOA #2
%Diff. in Class 1	-6.72	-4.59	-1.80	5.57	0.66	2.13	1.48	0.82
%Diff. in Class 2	17.06	21.43	3.97	1.98	-1.98	-2.78	3.17	-0.79
%Diff. in Class 3	11.57	8.17	-4.08	-2.04	2.72	5.44	-5.44	0.68
%Diff. in Class 4	18.18	18.18	-1.30	3.90	-2.60	5.20	-5.19	1.30
%Diff. in Class 5	0.00	1.66	5.81	5.39	0.83	0.41	1.66	4.56
%Diff. Total Correct	2.48	4.22	0.45	3.91	0.22	1.43	0.67	1.20
Average Costs	1.85	2.19	0.15	0.62	-0.06	0.40	-0.19	0.25

Table 6.35 Comparison of the MEDIAN performance of each classifier in five-class problem of dimension reduction study

	k-NN #1	*k*-NN #2	MLP #1	MLP #2	CA #1	CA #1	BOA #1	BOA #2
#Hidden nodes or # *k* or # Rules	5	5	12	12	16	16	16	16
# Epochs in training	0	0	2000	2000	200	200	200	200
%Correct in Class 1	84.26	84.10	67.05	73.11	79.18	73.77	79.84	76.07
%Correct in Class 2	40.87	43.65	76.59	66.27	66.67	70.24	67.46	66.67
%Correct in Class 3	60.54	54.42	65.99	70.07	69.39	72.11	65.99	74.15
%Correct in Class 4	58.44	66.23	85.71	75.32	76.62	67.53	80.52	71.43
%Correct in Class 5	92.53	90.87	87.14	85.89	91.29	92.95	88.80	92.12
Total % Correct	73.40	73.32	73.47	73.93	77.77	76.04	77.62	76.71
Average Costs	1.0816	1.0810	1.0592	1.0643	1.0572	1.0616	1.0577	1.0589

Table 6.36 Percentage difference of the MEDIAN performance from BOA (30 features) in five-class problem of dimension reduction study

	k-NN #1	KNN #2	MLP #1	MLP #2	CA #1	CA #1	BOA #1	BOA #2
%Diff. in Class 1	-4.59	-4.43	12.62	6.56	0.49	5.90	-0.17	3.60
%Diff. in Class 2	26.99	24.21	-8.73	1.59	1.19	-2.38	0.40	1.19
%Diff. in Class 3	10.89	17.01	5.44	1.36	2.04	-0.68	5.44	-2.72
%Diff. in Class 4	9.09	1.30	-18.18	-7.79	-9.09	0.00	-12.99	-3.90
%Diff. in Class 5	-0.41	1.25	4.98	6.23	0.83	-0.83	3.32	0.00
%Diff. Total Correct	4.67	4.75	4.60	4.14	0.30	2.03	0.45	1.36
Average Costs	2.15	2.10	0.04	0.52	-0.15	0.26	-0.10	0.01

Table 6.37 Comparison of the BEST performance of each classifier in four-class problem of dimension reduction study

	k-NN #1	*k*-NN #2	MLP #1	MLP #2	CA #1	CA #1	BOA #1	BOA #2
#Hidden nodes or # *k* or # Rules	5	5	12	12	14	14	14	14
# Epochs in training	0	0	2000	2000	200	200	200	200
%Correct in Class 1	98.14	97.56	94.78	94.78	95.24	85.38	95.48	96.40
%Correct in Class 2	64.63	55.78	73.47	73.47	78.23	69.39	73.47	72.79
%Correct in Class 3	63.64	62.34	84.42	81.82	77.92	71.43	77.92	75.32
%Correct in Class 4	92.53	95.02	87.55	88.38	89.21	90.87	93.78	94.61
Total % Correct	91.41	90.43	90.50	90.50	91.26	83.80	91.71	92.24
Average Costs	1.0469	1.0524	1.0342	1.0343	1.0338	1.0485	1.0338	1.0351

Table 6.38 Percentage difference of the BEST performance from BOA (30 features) in four-class problem of dimension reduction study

	k-NN #1	*k*-NN #2	MLP #1	MLP #2	CA #1	CA #1	BOA #1	BOA #2
%Diff. in Class 1	-2.55	-1.97	0.81	0.81	0.35	10.21	0.11	-0.81
%Diff. in Class 2	14.96	23.81	6.12	6.12	1.36	10.20	6.12	6.80
%Diff. in Class 3	15.58	16.88	-5.20	-2.60	1.30	7.79	1.30	3.90
%Diff. in Class 4	-0.41	-2.90	4.57	3.74	2.91	1.25	-1.66	-2.49
%Diff. Total Correct	0.83	1.81	1.74	1.74	0.98	8.44	0.53	0.00
Average Costs	1.69	2.22	0.46	0.47	0.42	1.85	0.42	0.54

Table 6.39 Comparison of the MEDIAN performance of each classifier in four-class problem of dimension reduction study

	k-NN #1	*k*-NN #2	MLP #1	MLP #2	CA #1	CA #1	BOA #1	BOA #2
#Hidden nodes or # *k* or # Rules	5	5	12	12	14	14	14	14
# Epochs in training	0	0	2000	2000	200	200	200	200
%Correct in Class 1	97.91	98.14	93.39	95.01	95.01	88.40	94.78	95.59
%Correct in Class 2	59.18	55.10	74.83	68.71	70.07	66.67	72.79	72.79
%Correct in Class 3	57.14	54.55	77.92	80.52	80.52	44.16	77.92	76.62
%Correct in Class 4	92.53	94.61	87.14	85.89	88.38	93.36	88.80	90.87
Total % Correct	90.28	90.20	89.30	89.60	90.20	84.33	90.28	91.11
Average Costs	1.0551	1.0572	1.0390	1.0393	1.0377	1.0604	1.0368	1.0359

Table 6.40 Percentage difference of the MEDIAN performance from BOA (30 features) in four-class problem of dimension reduction study

	k-NN #1	*k*-NN #2	MLP #1	MLP #2	CA #1	CA #1	BOA #1	BOA #2
%Diff. in Class 1	-2.67	-2.90	1.85	0.23	0.23	6.84	0.46	-0.35
%Diff. in Class 2	17.69	21.77	2.04	8.16	6.80	10.20	4.08	4.08
%Diff. in Class 3	15.59	18.18	-5.19	-7.79	-7.79	28.57	-5.19	-3.89
%Diff. in Class 4	-3.32	-5.40	2.07	3.32	0.83	-4.15	0.41	-1.66
%Diff. Total Correct	0.53	0.61	1.51	1.21	0.61	6.48	0.53	-0.30
Average Costs	1.81	2.02	0.26	0.29	0.14	2.33	0.05	-0.04

In PCA method the correlation matrix defined by Equation 3.4 was calculated. Then, the eigenvalues of this matrix were calculated. The largest eigenvalue is 23.8361. Example of eigenvalues in ascending orders of correlation matrix is shown in Figure 6.7. The first six eigenvalues that are relatively small are eliminated so that there are 24 features left.

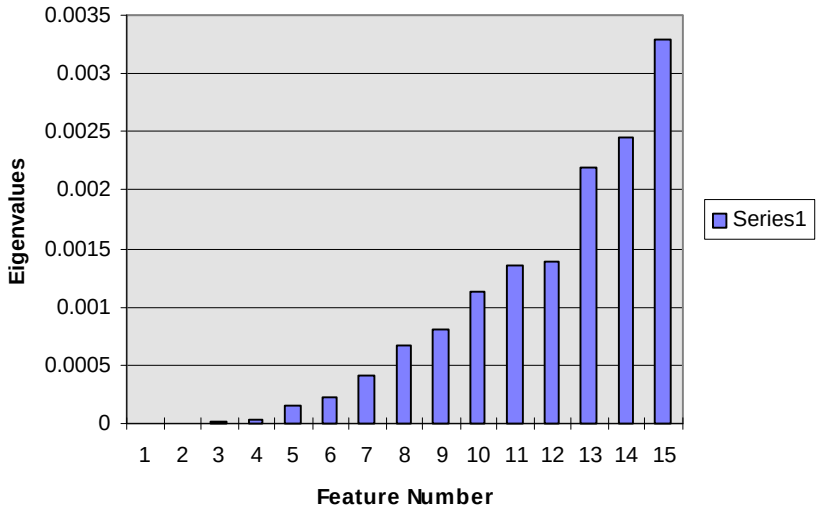

Figure 6.18 Example of some eigenvalues of correlation matrix

In the heuristic approach six insignificant features are eliminated. These features are feature number 7, 8, 17, 20, 25, and 27. The results show that all classifiers learn better from the data using the PCA method than the heuristic method. If there is further dimension reduction, PCA is the choice.

6.5 Conclusions

The results show that the AFLS-BOAs are better classifiers than MLPs, AFLS-CAs, and *k*-NN classifiers. All classifiers do very well in Iris classification problems. In vehicle classification problems *k*-NN classifiers are the worst classifiers of all proposed classifiers. A modified error backpropagation that uses an unequal learning factor in training is a very effective method if the numbers of data in each class are different; if they are relatively equal, the standard error backpropagation is good enough to solve classification problem. The PCA method is the best available method to reduce the input dimension.

The acoustic energy information is not enough to discriminate between closed classes such as passenger cars and pickups or vans. There is a need to have more information such as a number of axles, wheel spacing to improve correct classification rate. A fusion of sensors may be the answer to this problem.

Chapter 7

Summary and Future Research Directions

This dissertation research involves classification system design for vehicle acoustic signal classification. There are three important parts in this design: acoustic sensor design, feature analysis and extraction, and classifier design. The main focus is on the classifier design. Two main networks, multilayer perceptron networks (MLPs) and adaptive fuzzy logic systems (AFLSs), are considered, analyzed, and used as classifiers.

In acoustic sensor design a circular array of microphones is designed and built as a prototype sensor to detect acoustic signals. The circular array with multiple rings has a constant sidelobe level property, that is, its sidelobe levels are invariant with frequency. Its main beamwidth, on the other hand, varies with frequency in the same manner as a uniformly spaced array. The design procedure for circular arrays with multiple rings is rare, making circular arrays unpopular. Another reason is that the circular array configuration is hard to implement. In this research a modified genetic algorithm is used as a design tool in a circular array design.

Modified genetic algorithms (MGAs) are genetic algorithms (GAs) that are modified to work directly with real numbers while preserving the main features of standard GAs. There are three main advantages of MGAs over GAs. First, the resolution of the search grid can be adjusted easily and effectively during evolution, a process that is difficult with GAs. Second, there is no representation conversion required if the problem is already in a real number representation. Third, MGAs have a better rate of convergences than GAs for the same accuracy.

A circular array of microphones was designed, constructed and used in the data collection process. One difficulty encountered is that the assumption that all microphones have similar characteristics. The microphones used in this research varied in their frequency so much that the

beam pattern of the array was compromised. If many microphones are available, a selection process of picking ones with similar frequency responses should be done to overcome this problem. In this research the resources were limited, therefore, the array does not operate up to its design specifications. Its operational frequency bandwidth is narrower and at a lower frequency than it was originally designed.

Before acoustic signals can be classified, they must first be detected. The detection algorithm is developed to detect a presence of a passing vehicle. This detection algorithm will initialize and end the feature extraction process. Since different classes of vehicles have different sizes, the ending point of feature extraction was adapted according to an initial prediction by a fuzzy logic system. The information of duration and loudness are used as the inputs to this prediction system. After a vehicle is detected significant features are extracted. There are 30 features extracted for each vehicle. All features are time domain features extracted directly from acoustic energy information.

In classifier design MLPs and AFLSs are used because of their trainability and generalization. MIMO-AFLSs are developed in contrast to the normally decomposed MISO-AFLSs for each output. Thus, the MIMO-AFLS will have a compact form as its MLPs counter part. All fuzzy set theory and its operations are still supported and applicable to this MIMO-AFLS. There are two MIMO-AFLSs designed in this research, The first system consists of singleton fuzzifier, fuzzy rule base, fuzzy inference engine, and center average defuzzifier. The second system consists of the same components except a different defuzzifier, balance of area defuzzifier. The balance of area defuzzifier (BOA) was developed in this research. The BOA uses the shape information of fuzzy membership functions in the consequence part of the IF-THEN rules to obtain the result. Its output is close to the centroid of area defuzzification (COA) while requiring much less computation. With these choices of components, the AFLS can be trained by several training algorithms such as error backpropagation or genetic algorithms. In this research the designed AFLSs are trained by an error backpropagation algorithm. With MLP the

training algorithms adjust its synaptic weights to reduce its objective function, e.g., sum-squared-error function. With an AFLS the training algorithms adjust its center and spread parameters during training. The advantage of using an AFLS over a MLP is that its parameters can be initialized more effectively than MLPs. With good initialization the AFLS trains much faster than a corresponding MLP. Another advantage is that its structure is suitable to incorporate human expert knowledge easily.

Unequal numbers of available training data of different classes are normally encountered in practice. This unequal number of training data from different classes, along with the complexity of the problem, will make the classifier learn classes unequally. Mostly, the classifier will tend to classify the dominant class well and classify less well classes having fewer numbers of data. To solve this problem, the unequal learning factor is developed to be used during training phase of the classifier in this research.

In classification problems one class may be more important than others, i.e., classification costs, especially misclassification costs, may be different from class to class. If the classification costs are available and quantified, they may provide useful information to train the classifier. In this research the classification costs are quantified heuristically and used along with unequal learning factors to train all classifiers.

The performances of all proposed classifiers are better than the *k*-NN for the vehicle classification problem. A modified error backpropagation using an unequal learning factor in training is a very effective method if the number of data in each class is much different, i.e., 20 times greater. If they are relatively equal, the equal number of training data method using standard error backpropagation is good enough to solve the classification problem. The AFLS-BOA is a very effective system; its results are better than other proposed classifiers.

The advantages of using AFLS-BOA may be summarized as

a) easy, effective and straight forward to initialize its parameters
b) can be trained faster
c) easily to incorporate human knowledge

7.1 The main contributions of the dissertation can be summarized as follows:

- Developed modified genetic algorithm to work directly with real numbers.
- Designed and built an acoustic circular array of microphones to be used as traffic sensor.
- Developed a detection algorithm to be used with acoustic sensor.
- Developed a MIMO adaptive fuzzy logic system that can be used not, only in classification problems, but also in other applications such as system identification applications.
- Developed a new defuzzification method. The method is named balance of area (BOA) method. This defuzzification method can be used with MIMO-AFLS and can be trained from examples.
- Developed unequal learning factors used in training a classifier for classification problems with unequal numbers of training data from different classes.
- Demonstrated possibility of an acoustic sensor system with 97.95% correct classification rate between small and large vehicles on 1327 vehicles, 92.24% correct classification rate in four-class problem on 1327 vehicles, and 78.67% correct classification rate in five-class problem on 1327 vehicles.

7.2 List of future research directions

- Develop a microphone selection process to eliminate unacceptable microphone variations.
- Develop a new technique to eliminate the effect of large vehicle in adjacent lane.
- Develop a sensor fusion technique to have information from different sensors to detect and classify a vehicle.
- Develop a method of using human expert knowledge to corporate with AFLS in classification application.
- Develop a learning method to optimize network's parameters and size by using Gas or MGAs.

- Develop a fusion of classifiers' technique to take advantages of different classifiers. Different classifiers have different strengths. A system can be as follows:

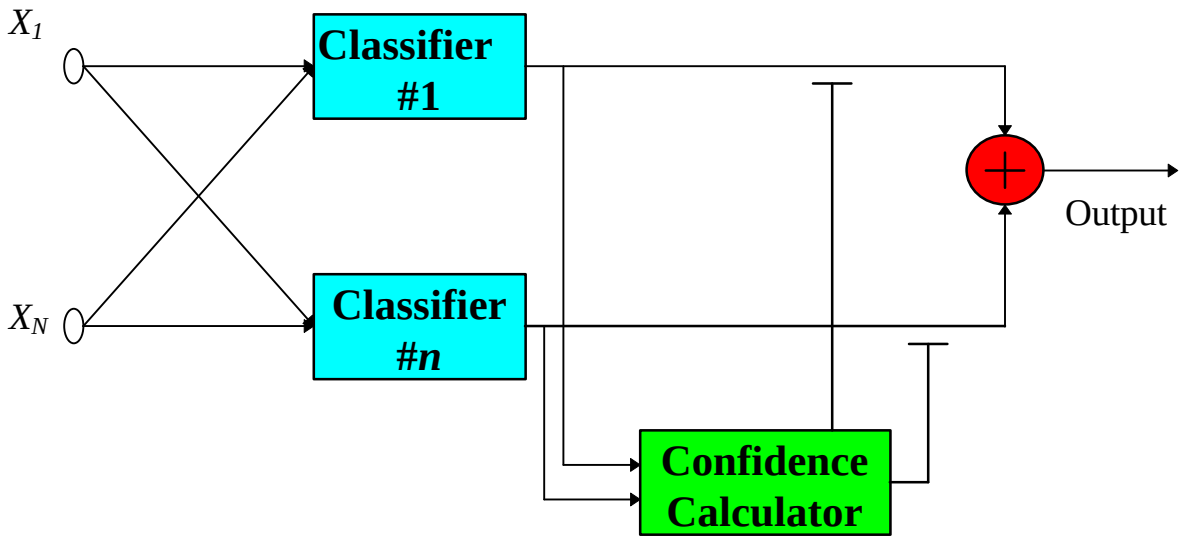

Figure 7.1 Fusion of classifiers

- Develop sub-task classifiers as

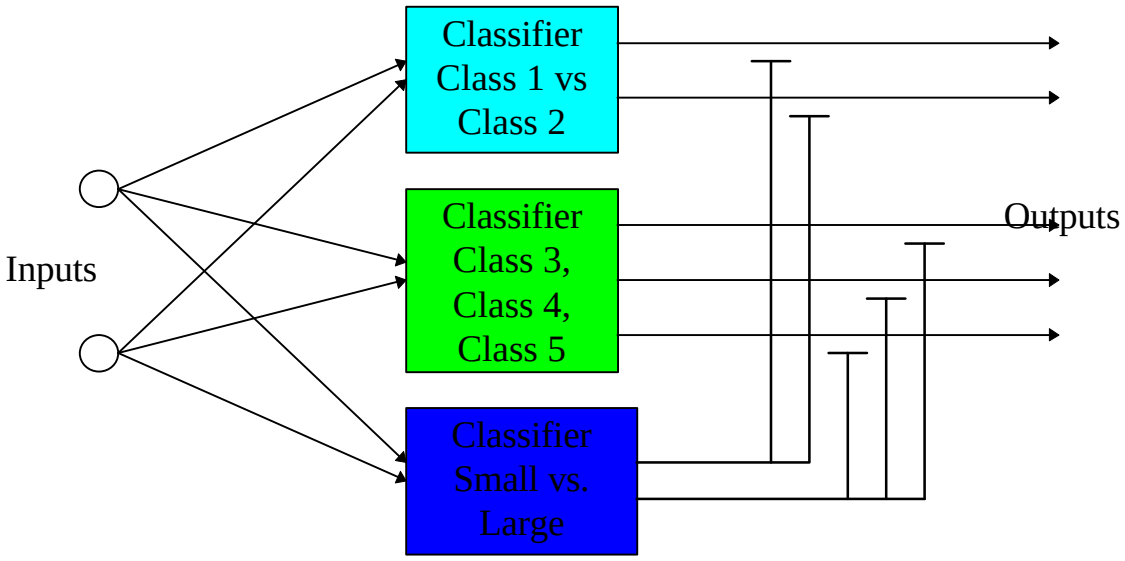

Figure 7.2 Sub-task Classifiers

♦ Develop a prediction and classification system:

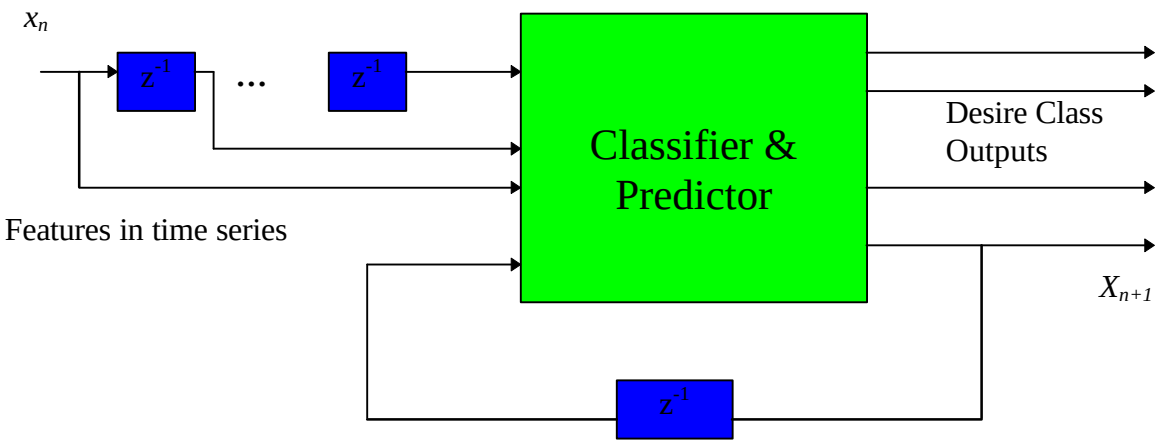

Figure 7.3 Classifier and predictor systems

Bibliography

[1] S. Abe and M. S. Lan, "A Classifier Using Fuzzy Rules Extracted Directly from Numerical Data", Second IEEE International Conference on Fuzzy Systems, vol. II, March 28-April 1, 1993, San Francisco, CA, pp. 1191-1198.

[2] S. Abe, M. S. Lan and R. Thawonmas, "Tuning of a Fuzzy Classifier Derived from Data", Proceedings of the Third IEEE Conferfence on Fuzzy Systems, Orlando, Florida, June 26-29, 1994, vol. II, pp. 786-791.

[3] R. Anand, K. G. Mehrotra, C. K. Mohan, and S. Ranka, "An Improved Algorithm for Neural Network Classification of Imbalanced Training Sets", IEEE Transactions on Neural Networks, vol. 4, no. 6, November 1993, pp. 962-969.

[4] T. R. Anderson, "Speaker Independent Phoneme Recognition with an Auditory Model and a Neural Network: A Comparison with Traditional Techniques", ICASSP, May 14-17, 1991, Toronto, Ontario, Canada, pp. 149-152.

[5] C. A. Balanis, Antenna Theory Analysis and Design, John Wiley & Sons, New York, 1982.

[6] A. Basu and T. Svendsen, "A Time-Frequency Segmental Neural Network for Phoneme Recognition", IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 2, April 19-22, 1994, pp. 509-512.

[7] K. G. Beauchamp and C. K. Yuen, Data Acquisition for Signal Analysis, George Allen & Unwin , Boston, 1980.

[8] S. Beck, L. Deuser, R. Still, and J. Whiteley, "A Hybrid Neural Network Classifier of Short Duration Acoustic Signals", International Joint Conference on Neural Networks, vol. 1, July 8-12, 1991, pp. 119-124.

[8] A. A. Beex, J. F. Tilki and G. Zakaria, Final Report on Signal Enhancement for Auditory Operator Input, The Bradley Department of Electrical Engineering, Virginia Polytechnic Institute and State University, 1994.

[9] A. Beltratti, S. Margarita, and P. Terna, Neural Networks for Economic and Financial Modeling, International Thomson Computer Press, New York, 1996.

[10] A. Bendiksen and K. Steiglitz, “Neural Networks for Voiced/Unvoiced Speech Classification”, ICASSP, April 3-6, 1990, New Mexico, pp. 521-524.

[11] Y. Bennani, F. F. Soulie, and P. Gallinari, “A Connectionist Approach for Automatic Speaker Identification”, ICASSP, April 3-6, 1990, New Mexico, pp. 265-268.

[12] J. Bezdek and S. Pal, Fuzzy Models For Pattern Recognition, IEEE Press Inc., New York, 1992.

[13] H. Bourlard and N. Morgan, “Continous Speech Recognition by Connectionist Statistical Methods”, IEEE Transactions on Neural Networks, vol. 4, no. 6, November 1993, pp. 893-909.

[14] R. H. Cabell, “The Automatic Identification of Aerospace Acoustic Sources”, Master's Thesis, Virginia Polytechnic Institute and State University, 1989.

[15] G. Chakraborty, N. Shiratori, and S. Noguchi, “A Quickly Trained Artificial Neural Network with Single Hidden Layer Gaussian Units”, 1993 IEEE International Conference on Neural Networks, Vol. 1, San Francisco, California, March 28-April 1, 1993, pp.466-472.

[16] S. L. Chiu, “A Cluster Estimation Method with Extension to Fuzzy Model Identification”, Proceedings of The Third IEEE Conferfence on Fuzzy Systems, Orlando, Florida, June 26-29, 1994, vol. II, pp. 1240-1245.

[17] M. Clifford, Microphones 2nd Edition, Tab Books Inc, Blue Ridge Summit, PA, 1982.

[18] Y. L. Cun, “A Theoretical Framework for Back-Propagation”, Proceedings of the Connectionist Models Summer School, 1988, pp. 21-28.

[19] D. Dasgupta and D. R. McGregor, “Designing Application-Specific Neural Networks using the Structured Genetic Algorithm”, International Workshop on Combinations of Genetic Algorithms and Neural Networks, Baltimore, Maryland, June 6, 1992, pp. 87-96.

[20] R. N. Dave, “Robust Fuzzy Clustering Algorithms”, Second IEEE International Conference on Fuzzy Systems, vol. II, March 28-April 1, 1993, San Francisco, CA, pp. 1281-1286.

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

[22] M. E. Delany, “Traffic noise”, Acoustics and Vibration Progress, vol. 1, John Wiley & Sons, New York, 1974.

[23] J. R. Deller, Jr., J. G. Proakis, and J. H. L. Hansen, Discrete-Time Processing of Speech Signals, Macmillan Publishing Company, New York, 1993.

[24] X. Driancourt and P. Gallinari, "A Speech Recognizer Optimaly Combining Learning Vector Quantization, Dynamic Programming and Multi-Layer Perceptron", ICASSP, March 23-26, 1992, San Francisco, CA, pp. 609-612.

[25] R. Duda and P. Hart, Pattern Classification and Scene Analysis, New York : Wiley Interscience, 1973.

[26] J. C. Dunn, "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Seperated Clusters", Journal Cybernetics, vol. 3, no. 3, 1973, pp. 32-57.

[27] S. E. Fahlman and C. Lebiere, "The Cascade-Correlation Learning Architecture", Advance in Neural Information Processing System 2, Morgan Kaufmann Publishers, San Mateo, CA, 1990, pp.524-532.

[28] Field Test of Monitoring of Urban Vehicle Operations Using Non-Intrusive Technologies, vol. Task Two Report : Initial Field Test Results, U.S. Department of Transportation Federal Highway Administration, May 1996.

[29] M. Franzini, K. F. Lee, and A. Waibel, "Conectionist Viterbi Training: A New Hybrid Method for Continuous Speech Recognition", ICASSP, April 3-6, 1990, New Mexico, pp. 425-428.

[30] I. Flood, "A Gaussian-Base Feedforward Network Architecture and Complementary Training Algorithm", 1991 International Joint Conference on Neural Networks, Vol.1, Nov. 18-21, 1991, Singapore, pp.171-176.

[31] M. T. Gately and P. A. Penz, "A Geometrical Overview of Neural Networks", TI Technical Journal, November-December 1990, pp. 4-15.

[32] P. D. Gader and J. M. Keller, "Applications of Fuzzy Set Theory to Handwriting Recognition", Proceedings of The Third IEEE Conferfence on Fuzzy Systems, Orlando, Florida, June 26-29, 1994, vol. II, pp. 910-917.

[33] B. Golomb and T. Sejnowski, “Sex Recognition from Faces Using Neural Networks”, Applications of Neural Networks, Murray, A. (Editor), Kluwer Academic Publishers, 1995, pp. 71-92.

[34] M. M. Goodwin and G. W. Elko, “Constant Beamwidth Beamforming”, IEEE Antennas and Propagation Society International Symposium, 1993, pp. 169 -172.

[35] A. D. Gordon, Classification : Methods for the Exploratory Analysis of Multivariate Data, Chapman and Hall, New York, 1981.

[36] M. M. Gupta and G. K. Knopf, “Neuro-Vision Systems: A Tutorial”, Neuro-Vision Systems: Principles and Applications, IEEE Press, 1994, pp.1-34.

[37] P. Haffner, “Connectionist Word-Level Classification in Speech”, ICASSP, March 23-26, 1992, San Francisco, CA, pp. 621-624.

[38] S. Haykin, Neural Networks : A Comprehensive Foundation, Macmillan College Publishing Company, New York, 1994.

[39] G. E. Hinton, “Connectionist Learning Procedures”, Artificial Intelligence, Volume 40, Number 1, 1989, pp. 185-234.

[40] D. R. Hush and B. G. Horne, “Progress in Supervised Neural Networks, What’s New Since Lippmann?”, IEEE Signal Processing Magazine, January 1993, pp.8-39.

[41] H. Ishibuchi, K. Nozaki, N. Yamamoto and H. Tanaka, “Genetic Operations for Rule Selection in Fuzzy Classification System”, Proceedings of the Fifth International Fuzzy Systems Association World Congress, vol. I, July 4-9, 1993, pp. 15-18.

[42] H. Ishibuchi, K. Nozaki, N. Yamamoto and H. Tanaka, “Acquisition of Fuzzy Classification Knowledge Using Genetic Algorithm”, Proceedings of The Third IEEE Conferfence on Fuzzy Systems, Orlando, Florida, June 26-29, 1994, vol. II, pp.1963-1968.

[43] H. Iwamida, S. Katagiri, E. McDermott, and Y. Tohkura, “A Hybrid Speech Recognition System using HMMS with an LVQ-Trained Codebook”, ICASSP, April 3-6, 1990, New Mexico, pp. 489-492.

[44] R. A. Jacobs and M. I. Jordan, “A competitive modular connectionist architecture.”, In Advances in Neural Information Processing Systems 3, Morgan Kaufmann Publishers, San Mateo, CA, 1991, pp.767-773.

[45] R. D. James and S. Sampan, AT&T SmartSonic Development Report, Center for Transportation Research, Virginia Polytechnic Institute and State University, 1995.

[46] R. D. James and S. Sampan, “Vehicle Classification of Acoustic Signals Using Neural Networks”, ITS America Fifth Annual Meeting and Exposition, March 15-17, 1995.

[47] J. R. Jang and C. T. Sun, “Neuro-Fuzzy Modeling and Control”, Proceedings of The IEEE, Volume 83, Number 3, March 1995, pp. 378-406.

[48] R. E. Kalil, “Synapse Formation in the Developing Brain”, Bio Systems, vol. 23, 1990, pp. 297-303.

[49] E. R. Kandel, “Small Systems of Neurons”, Neuro-Vision Systems: Principles and Applications, IEEE Press, 1994, pp.80-89.

[50] D. G. Kimber, M. A. Bush, and G. N. Tajchman, “Speaker-Independent Vowel Classification using Hidden Markov Models and LVQ2”, ICASSP, April 3-6, 1990, New Mexico, pp. 497-500.

[51] G. J. Klir and B. Yuan, Fuzzy Sets and Fuzzy Logic Theory and Applications, Prentice Hall PTR, New Jersey, 1995.

[52] T. Kohonen, “An introduction to Neural Computing”, Neural Networks, Volume 1, 1988, pp.8-16.

[53] T. Kohonen, “The Self-Organizing Map”, Proceedings of IEEE, volume 78, Number 9, September 1990, pp. 1464-1480.

[54] Y. Komori, “A Neural Fuzzy Training Approach For Continuous Speech Recognition Improvement”, ICASSP, March 23-26, 1992, San Francisco, CA, pp. 405-408.

[55] Y. Komori, K. Hatazaki, T. Tanaka, and T. Kawabata, “Combining Phoneme Identification Neural Networks Into an Expert System Using Spectrogram Reading Knowledge”, ICASSP, April 3-6, 1990, New Mexico, pp. 505-508.

[56] B. Kosko, “Unsupervised Learning in Noise”, IEEE Transactions on Neural Networks, vol. 1, no. 1, March 1990, pp. 44-57.

[57] B. Kosko, Neural networks for signal processing, Prentice-Hall, Inc., A simon & Schuster Company, Englewood Cliffs, NJ 07632, 1992.

[58] B. Kosko, Neural Networks and Fuzzy Systems : A Dynamical Systems Approach to Machine Intelligence, Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[59] B. Kosko, “Fuzzy Systems as Universal Approximator”, IEEE International Conference on Fuzzy Systems, March 8-12, 1992, San Diego, California, pp. 1153-1162.

[60] R. Krishnapuram, “Generation of Membership Functions via Possibilistic Clustering”, Proceedings of The Third IEEE Conferfence on Fuzzy Systems, Orlando, Florida, June 26-29, 1994, vol. II, pp. 902-908.

[61] C. C. Lee, “Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Part I”, IEEE Transactions on Systems, Man, and Cybernetics, vol. 20, no. 2, March/April 1990, pp. 404-418.

[62] C. C. Lee, “Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Part II”, IEEE Transactions on Systems, Man, and Cybernetics, vol. 20, no. 2, March/April 1990, pp. 419-432.

[63] D. S. Lee, S. N. Srihari and R. Gaborski, “Bayesian and neural network pattern regnition : a theoretical connection and empirical results with handwritten characters”, Artificial Neural Networks and Statistical Pattern Recognition : Old and New Connections, I.K. Sethi and A.K. Jain (Editors), Elsevier Science Publishers, 1991.

[64] S. C. Lee and E. T. Lee, “Fuzzy Neural Networks”, Mathematical Biosciences, vol. 23, 1975, pp. 151-177.

[65] H. C. Leung and V. W. Zue, “Phonetic Classification Using Multi-Layer Perceptron”, ICASSP , April 3-6, 1990, New Mexico, pp. 525-528.

[66] R. P. Lippmann, “An Introduction to Computing with Neural Nets”, IEEE Acoustics, Speech, and Signal Processing Magazine, April 1987, pp. 4-22.

[67] J. M. Mendel, “Fuzzy Logic Systems for Engineering: A Tutorial”, Proceeding of The IEEE, volume 83, Number 3, March 1995, pp. 345-377.

[68] D. Michie, D. J. Spiegelhalter and C. C. Taylor, Machine Learning, Neural and Statistical Classification, Ellis Horwood Limited, West Sussex, England, 1994.

[69] K. Mitsubuchi, S. Isaka, and Z. Y. Zhao, "A Fuzzy Rule Generation System", Proceedings of the Fifth International Fuzzy Systems Association World Congress, vol. I, July 4-9, 1993, pp. 11-14.

[70] D. P. Morgan and C. L. Scofield, Neural Networks and Speech Processing, Kluwer Academic Publishers, Boston, 1991.

[71] G. C. Mouzouris and J. M. Mendel, "Non-Singleton Fuzzy Logic Systems", Proceedings of The Third IEEE Conferfence on Fuzzy Systems, Orlando, Florida, June 26-29, 1994, vol. I, pp. 456-461.

[72] S. Nakamura, H. Sawai, and M. Sugiyama, "Speaker-Independent Phoneme Recognition Using Large-Scale Neural Networks", ICASSP, March 23-26, 1992, San Francisco, CA, pp. I-409-412.

[73] V. Nedeljkovic, "A Novel Multilayer Neural Networks Training Algorithm that Minimizes the Probability of Classification Error", IEEE Transactions on Neural Networks, vol. 4, no. 4, July 1993, pp. 650-659.

[74] L. T. Niles and H. F. Silverman, "Combining Hidden Markov Model and Neural Network Classifiers", ICASSP, April 3-6, 1990, New Mexico, pp. 417-420.

[75] T. Nishina, M. Hagiwara, and M. Nakagawa, "Fuzzy Inference Neural Networks which Automatically Partition a Pattern Space and Extract Fuzzy If-Then Rules", Proceedings of The Third IEEE Conferfence on Fuzzy Systems, Orlando, Florida, June 26-29, 1994, vol. II, pp. 1314-1319

[76] K. Nozaki, H. Ishibuchi and H. Tanaka, "Performance Evaluation of Fuzzy-Rule-Based Classification Methods", Proceedings of the Fifth International Fuzzy Systems Association World Congress, vol. I, July 4-9, 1993, pp. 167-170.

[77] K. Nozaki, H. Ishibuchi and H. Tanaka, "Trainable Fuzzy Classification Systems Based on Fuzzy If-Then Rules", Proceedings of The Third IEEE Conference on Fuzzy Systems, IEEE World Congress on Computational Intelligence, vol. 1, June 26-29, 1994, pp. 498-502.

[78] E. Oja, “Data Compression, Feature Extraction, and Autoassociation in Feedforward Neural Networks”, Artificial Neural Networks, T. Kohonen, K. Makisara, O. Simula, and I. Kangas (Editors), Elsevier Science Publishers, North-Holland, 1991.

[79] J. Oglesby and J.S. Mason, “Optimisation of Neural Models for Speaker Identification”, International Conference on Acoustics, Speech, and Signal Processing (ICASSP) , April 3-6, 1990, New Mexico, pp. 261-264.

[80] G. F. Page, J. B. Gomm and D. Williams, Application of Neural Networks to modelling and control, Chapman & Hall, New York, 1993.

[81] Y. H. Pao, Adaptive Pattern Recognition and Neural Networks, Addison-Wesley Publishing Company, Inc., 1989.

[82] G. Porges, Applied Acoustics, The Heckman Binery, Inc., N. Manchester, Indiana, 1977.

[83] B. Qiu, P. Im and A. Pleasants, “The design of Neural Network Configuration for Object Recognition”, IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 2, April 19-22, 1994, pp.561-564.

[84] C. Raymond, S. Boverie and A. Titli, “First Evaluation of Fuzzy MIMO Control Laws”, Proceedings of The Third IEEE Conferfence on Fuzzy Systems, Orlando, Florida, June 26-29, 1994, vol. II, pp. 545-548.

[86] B. D. Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, Cambridge, UK, 1996.

[87] S. Roychowdhury, B. H. Wang, and S. K. Ahn, “Radial Defuzzification Method”, Proceedings of The Third IEEE Conferfence on Fuzzy Systems, Orlando, Florida, June 26-29, 1994, vol. II, pp. 1153-1158.

[88] D. W. Ruck, S. K. Rogers, M. Kabrisky, M. E. Oxley, and B. W. Suter, “The Multilayer Perceptron as an Approximation to a Bayes Optimal Discriminant Function”, IEEE Trans. Neural Networks, vol. 1, no. 4, December 1990, pp. 296-298.

[89] V. H. Rumsey, Frequency Independent Antennas, Academic Press, New York, 1996.

[90] E. H. Ruspini, “A New Approach to Clustering”, Inform. Control, vol. 15, no. 1, July 1969, pp. 22-32.

[91] J. D. Schaffer, D. Whitley, and L. J. Eshelman, “Combinations of Genetic Algorithms and Neural Networks: A Survey of the State of the Art”, International Workshop on Combinations of Genetic Algorithms and Neural Networks, Baltimore, Maryland, June 6, 1992, pp. 1-37.

[92] R. J. Schalkoff, Pattern Recognition: Statistical, Structural and Neural Approaches, John Wiley & Sons, Inc., New York, 1992.

[93] E. A. Scott, “Recognition of Aerospace Acoustic Sources Using Advanced Pattern Recognition Techniques”, Master's Thesis, Virginia Polytechnic Institute and State University, 1991.

[94] Second Interim Report, Interstate 95, Multi-State Traffic Monitoring Evaulation Project, September 1994.

[95] S. M. Selby, Standard Mathematical Tables, Seventeenth Edition, The Chemical Rubber Co., 1969.

[96] E. Singer and R. P. Lippmann, “A Speech Recognizer Using Radial Basis Function Neural Networks in an HMM Framework”, ICASSP, March 23-26, 1992, San Francisco, CA, pp. I-629-632.

[97] SmartSonic Acoustic Vehicle Detection System, Specifications, International Road Dynamics, Inc., January 1996.

[98] R. W. B. Stephens and H. G. Leventhall, Acoustics and Vibration Progress, John Wiley & Sons, New York, 1974.

[99] H. Tanaka, H. Ishibuchi, and S. Yoshikaw, “Discriminant Analysis Based on Exponential Possibility Distributions”, Proceedings of The Third IEEE Conferfence on Fuzzy Systems, Orlando, Florida, June 26-29, 1994, vol. II, pp.802-807.

[100] J. S. Taur and S. Y. Kung, “Fuzzy-Decision Neural Network”, IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 2, April 19-22, 1994, pp. 577-580.

[101] Traffic Monitoring Guide, Federal Highway Administration, U.S. Department of Transportation, June 1985.

[102] Traffic Monitoring Systems Development Study, Virginia Department of Transportation, December, 1993.

[103] H. F. VanLandingham and S. Sampan, “Evolutionary Algorithms for Design”, accepted for presentation for SOUTHEASTCON’97, Blacksburg, VA, April 12-14, 1997.
[104] P. Vuorimaa, “Use of The Fuzzy Self-Organizing Map in Pattern Recognition”, Proceedings of The Third IEEE Conferfence on Fuzzy Systems, Orlando, Florida, June 26-29, 1994, vol. II, pp. 798-801.
[105] L. X. Wang, Adaptive Fuzzy Systems and Control, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1994.
[106] L. X. Wang, “Fuzzy Systems are Universal Approximators”, IEEE International Conference on Fuzzy Systems, March 8-12, 1992, San Diego, California, pp. 1163-1170.
[107] L. X. Wang and J. M. Mendel, “Back-Propagation Fuzzy System as Nonlinear Dynamic System Identifiers”, IEEE International Conference on Fuzzy Systems, March 8-12, 1992, San Diego, California, pp. 1409-1418.
[108] D. B. Ward, R. A. Kennedy, and R. C. Williamson, “Design of Frequency-Invariant Broadband Far-Field Sensor Array”, IEEE Antennas and Propagation Society International Symposium, 1994, pp. 1274-1277.
[109] S. M. Weiss and I. Kapouleas, “An Empirical Comparison of Pattern Recognition, Neural Nets, and Machine Learning Classification Methods”, Proceedings of the International Joint Conference on Artificial Intelligence, 1989, pp. 781-787.
[110] W. Wei and J. M. Mendel, “A Fuzzy Classifier That Uses Both Crisp Samples and Linguistic Knowledge”, Proceedings of The Third IEEE Conferfence on Fuzzy Systems, Orlando, Florida, June 26-29, 1994, vol. II, pp.792-797.
[111] P. J. Werbos, “Links Between Artificial Neural Networks (ANN) and Statistical Pattern Recognition”, Artificial Neural Networks and Statistical Pattern Recognition : Old and New Connections, I.K. Sethi and A.K. Jain (Editors), Elsevier Science Publishers, 1991.
[112] B. Widrow, M. A. Lehr, “30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation”, Proceedings of IEEE, volume 78, Number 9, September 1990, pp. 1415-1442.

[113] B. Widrow, R. G. Winter, and R. A. Baxter, “Layered Neural Nets for Pattern Recognition”, IEEE Acoustics, Speech, and Signal processing, vol. 36. no. 7, 1988, pp. 1109-1118.

[114] Y. F. Wong, “How Gaussian Radial Basis Function Work”, International Joint Conference on Neural Networks, Vol.III, July 8-12, 1991, Seattle, WA, pp. 133-138.

[115] X. Ye and N. K. Loh, “Dynamic System Identification using Recurrent Radial Basis Function Network”, proceeding of the American Control Conference, Vol. 3, San Francisco, California, June 1993, pp.2912-2916.

[116] B. Yu and B. Yuan, “A Feature Selection Method for Multi-class-set Classification”, International Joint Conference on Neural Networks, June 7-11, 1992, pp. 567-572.

[117] L. A. Zadeh, “Fuzzy Sets”, Inform. Control, vol. 8, 1965, pp. 338-353.

[118] L. A. Zadeh, “Outline of a New Aproach to Analysis of Complex Systems and Decision Process”, IEEE Trans. Syst., Man, Cybern., vol. SMC-3, no. 1, January 1973, pp.28-44.

[119] X. J. Zeng, and M. G. Singh, “Approximation Theory of Fuzzy Systems”, Proceedings of The Third IEEE Conferfence on Fuzzy Systems, Orlando, Florida, June 26-29, 1994, vol. II, pp. 1916-1921

[120] M. Zhu and K. Fellbaum, “A Connectionist Model for Speaker-Independent Isolated Word Recognition”, ICASSP, April 3-6, 1990, New Mexico, pp. 529-532.

Vita

Somkiat Sampan

Somkiat Sampan was born in Chiangmai, Thailand, in 1962. He graduated from the Armed Force Academy Preparatory School, Bangkok, Thailand, in 1982. He earned the B.S. in Electrical Engineering from Virginia Military Institute, Lexington, Virginia, in 1987. He then earned the M.S. in Electrical Engineering from Virginia Polytechnic Institute and State University, Blacksburg, Virginia, in 1989. He then was an instructor in the Electrical Engineering Department at Chulachomkloa Royal Military Academy, Nakhon Nayok, Thailand until 1992. Since then he has been pursuing higher education at Virginia Polytechnic Institute and State University, while also working as a graduate research assistant at the Center for Transportation Research. He anticipates receiving his doctorate degree in Electrical Engineering in May, 1997.

His research interests are in the areas of neural networks, fuzzy logic systems, traffic sensors, pattern classfication, signal processing, system identification and control. His hobbies include computer programming, reading, running and traveling.