Data Mining Using Intelligent Systems: an Optimized Weighted Fuzzy Decision Tree Approach

By

XuQin Li

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

School of Engineering

December 2010

THE UNIVERSITY OF WARWICK

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my sincere appreciation to my supervisor, Prof. Evor L. Hines for his wonderful support, excellent supervision and endless guidance during the entire period of my PhD studies and writing of this thesis. I would also like to extend my sincere gratitude to Dr. Mark S Leeson for his kind assistance, advice and encouragement for my research work.

I would also like to thank my wife, Min, for her support and tolerance of my PhD study, as well as all night companionship during the past several years. Special thanks go to my parents, parents-in-law and my sister for their help and support that they have given me in so many ways through the good and bad times.

Thanks also go to Dr. Jianhua Yang, Dr. Harsimrat Singh, Mr. Huseyin Kusetogullari, Mr. Reza Ghaffari and Dr. Wei Ren at the School of Engineering for their fruitful discussion and support.

Last but not least, I thank the financial support from the Warwick Postgraduate Research Fellowship (WPRF) and UK Overseas Research Students Awards Scheme (ORSAS).

Declaration

This thesis is presented in accordance with the regulations for the degree of doctor of philosophy. The work described in this thesis is entirely original and my own, except where otherwise indicated.

List of Publications

Book Chapters

XuQin Li, Evor L. Hines, Mark S. Leeson, Daciana. D. Iliescu, (2010), “Enhancing the classification of Eye Bacteria Using Bagging to Multilayer Perceptron”. Intelligent Systems for Machine Olfaction: Tools and Methodologies, Medical Information Science Reference.

Xu-Qin Li, M. S. Leeson, E. L. Hines, D. S. Huang, J. Yang, (2010), “*Neural Networks for Solving Linear and Quadratic Programming Problems with Modified Newton's and Levenberg-Marquardt Methods*”, in ‘Advances in Mathematics Research’, Volume 11, Editors: A. R. Baswell, Nova Publishers;

XuQin Li, Carlos Ramirez, Evor L. Hines, Mark S. Leeson, Phil Purnell, and M. Pharaoh, (2008), “Discriminating Fiber-Reinforced Plastic Signatures Related to Specific Failure Mechanisms: Using Self-Organizing Maps and Fuzzy C-Means”, intelligent systems: Techniques and Applications, Shaker Publications;

Jianhua Yang, Evor. L. Hines, Ian. Guymer, Daciana Iliescu, Mark. S. Leeson, G. P. King and XuQin. Li, (2007), “*A Genetic Algorithm-Artificial Neural Network Method for the Prediction of Longitudinal Dispersion Coefficient in Rivers*”, Advancing Artificial Intelligence through Biological Process Applications, Medical Information Science Reference;

Journal Papers

Harsimrat Singh, XuQin Li, Evor Hines, Nigel Stocks, (2007), “*Classification and feature extraction strategies for multi channel multi trial BCI data*”. International Journal of Bioelectromagnetism, Vol. 9, No. 4, pp. 233 – 236;

XuQin Li, Mark. S. Leeson, Evor. L. Hines, D. D. Iliescu, (2010, under review), "*Improving Classification Rate Using Optimized Weighted Fuzzy Decision Trees*", Journal of Intelligent and fuzzy system

Conference Papers

XuQin Li, Carlos Ramirez, Evor L. Hines, Mark S. Leeson, Phil Purnell, and M. Pharaoh, "*Pattern recognition of fiber-reinforced plastic failure mechanism using intelligent computing techniques*", IEEE World Congress on Computational Intelligence, Hong Kong, June 1-6, 2008

Abstract

Data mining can be said to have the aim to analyze the observational datasets to find relationships and to present the data in ways that are both understandable and useful. In this thesis, some existing intelligent systems techniques such as Self-Organizing Map, Fuzzy C-means and decision tree are used to analyze several datasets. The techniques are used to provide flexible information processing capability for handling real-life situations. This thesis is concerned with the design, implementation, testing and application of these techniques to those datasets. The thesis also introduces a hybrid intelligent systems technique: Optimized Weighted Fuzzy Decision Tree (OWFDT) with the aim of improving Fuzzy Decision Trees (FDT) and solving practical problems.

This thesis first proposes an optimized weighted fuzzy decision tree, incorporating the introduction of Fuzzy C-Means to fuzzify the input instances but keeping the expected labels crisp. This leads to a different output layer activation function and weight connection in the neural network (NN) structure obtained by mapping the FDT to the NN. A momentum term was also introduced into the learning process to train the weight connections to avoid oscillation or divergence. A new reasoning mechanism has been also proposed to combine the constructed tree with those weights which had been optimized in the learning process. This thesis also makes a comparison between the OWFDT and two benchmark algorithms, Fuzzy ID3 and weighted FDT.

SIx datasets ranging from material science to medical and civil engineering were introduced as case study applications. These datasets involve classification of composite material failure mechanism, classification of electrocorticography (ECoG)/Electroencephalogram (EEG) signals, eye bacteria prediction and wave

overtopping prediction. Different intelligent systems techniques were used to cluster the patterns and predict the classes although OWFDT was used to design classifiers for all the datasets. In the material dataset, Self-Organizing Map and Fuzzy C-Means were used to cluster the acoustic event signals and classify those events to different failure mechanism, after the classification, OWFDT was introduced to design a classifier in an attempt to classify acoustic event signals. For the eye bacteria dataset, we use the bagging technique to improve the classification accuracy of Multilayer Perceptrons and Decision Trees. Bootstrap aggregating (bagging) to Decision Tree also helped to select those most important sensors (features) so that the dimension of the data could be reduced. Those features which were most important were used to grow the OWFDT and the curse of dimensionality problem could be solved using this approach. The last dataset, which is concerned with wave overtopping, was used to benchmark OWFDT with some other Intelligent Systems techniques, such as Adaptive Neuro-Fuzzy Inference System (ANFIS), Evolving Fuzzy Neural Network (EFuNN), Genetic Neural Mathematical Method (GNMM) and Fuzzy ARTMAP.

Through analyzing these datasets using these Intelligent Systems Techniques, it has been shown that patterns and classes can be found or can be classified through combining those techniques together. OWFDT has also demonstrated its efficiency and effectiveness as compared with a conventional fuzzy Decision Tree and weighted fuzzy Decision Tree.

Abbreviations

AE	Acoustic Event
AI	Artificial Intelligence
AN	Artificial neuron
ANFIS	Adaptive Neuro-Fuzzy Inference System
ANN	Artificial Neural Network
ART	Adaptive Resonance Theory
BCI	Brain-Computer Interface
BP	Back-Propagation
CART	Classification and Regression Tree
CSP	Common Spatial Patterns
DM	Data Mining
DT	Decision Tree
ECoG	Electrocorticography
EEG	Electroencephalogram
EFuNN	Evolving Fuzzy Neural Network
EN	Electronic Nose
FCM	Fuzzy C-Means
FDT	Fuzzy Decision Tree
FFT	Fast Fourier Transformation
FIS	Fuzzy Inference System
FL	Fuzzy Logic
FNN	Feedforward Neural Network

GA	Genetic Algorithm
GNMM	Genetic Neural Mathematical Method
ICA	Independent Component Analysis
ID3	Iterative Dichotomizer 3
IS	Intelligent System
IST	Intelligent Systems Technique
KDD	Knowledge Discovery from Data
LM	Levenberg-Marquardt
MF	Membership Function
MLP	Multi-Layer Perceptron
MSE	Mean Square Error
NN	Neural Network
OWFDT	Optimized Weighted Fuzzy Decision Tree
PCA	Principal Components Analysis
PNN	Probabilistic Neural Network
PR	Pattern Recognition
RBF	Radial Basis Function
SEM	Scanning Electron Microscope
SC	Soft Computing
SOM	Self-Organizing Map
WFDT	Weighted Fuzzy Decision Tree

Chapter I: Introduction to Data Mining

1.1 Introduction to data mining

Advances in data collection and storage technology have enabled researchers to accumulate vast amounts of data. With the enormous amount of data stored in files, databases, and other repositories, researchers from many areas have been stimulated to adopt and develop new techniques for data analysis in different fields of science. Powerful means of analyzing, interpreting and extracting interesting knowledge that could help in decision-making processes have also been designed (Gaber, 2010). Data mining is such a technology that combines data analysis methods (intelligent systems techniques) with sophisticated algorithms for processing large volumes of data.

Data mining is the process of applying neural networks, clustering, fuzzy logic and decision tree etc. to data with the intention of uncovering hidden patterns and extracting information (Kantardzic, 2002). It involves an integration of these different techniques from different disciplines which can facilitate the extraction of knowledge from a large amount of data (Tan, Steinbach, & Kumar, 2005).

1.2 Data Mining and Knowledge Discovery in Databases

The term ‘data mining’ is often regarded as an integral part of Knowledge Discovery in Databases (KDD), which is overall process of extracting implicit, previously unknown and potentially useful information from raw data (Larose, 2005). There have been notable successes in the use of statistical, computational and machine learning techniques to discover scientific knowledge in the field of engineering, biology, medicine and physics etc.

The KDD process consists of applying data analysis and computing algorithms which produce a particular enumeration of patterns over data. In addition to that, KDD will also discover regularities to improve future decisions. Figure 1-1 shows the relationship between KDD and intelligent systems techniques (Venugopal, G.Srinivasa, & Patnaik, 2009).

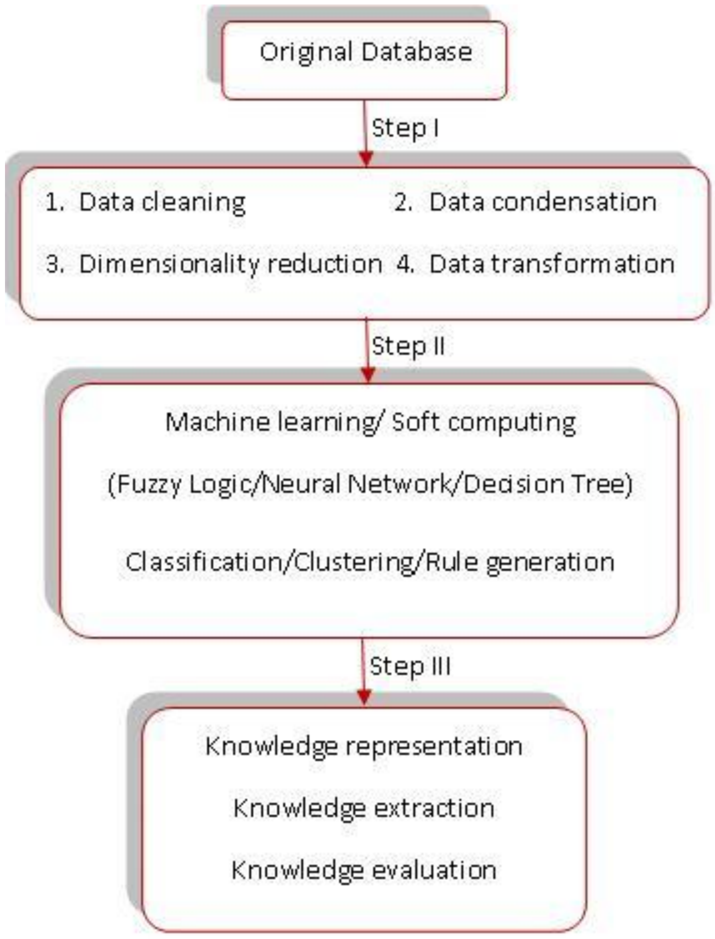


Figure 1-1: KDD and intelligent systems techniques

The steps in the KDD process are briefly explained below (Venugopal, et al., 2009):

The first step is concerned with data preprocessing which consists of the following four elements:

- Data cleaning to remove noise and irrelevant data from collection;
- Data integration which involves combining of multiple and heterogeneous data sources;
- Data selection where data relevant to analysis task is retrieved from the database;

- Data transformation where consolidated data is transformed into forms appropriate for the mining procedure;

The second step is concerned with data mining process which consists of the following two elements:

- Data mining which is essential and where intelligent systems techniques are applied in order to discover and extract patterns from datasets;
- Pattern evaluation where patterns representing knowledge are measured and identified;

The third step is concerned with knowledge presentation which consists of the following element:

- Knowledge presentation where knowledge is presented to final users in a visualized way to help them understand and interpret the mining results.

Although data mining is not isolated with data preprocessing and knowledge presentation steps, this thesis will focus on the data mining process which involves design and implementation of intelligent systems techniques.

Data mining is concerned with building models to determine patterns from observed data. The models play the role of inferring knowledge. A model is simply an algorithm or a set of rules that connects a collection of inputs to a particular target or output (Larose, 2006). Neural Networks, fuzzy inference systems, decision trees, regression and other intelligent systems techniques covered in this thesis are all for creating models (Berry &

Linoff, 2004). Deciding whether the model reflects useful knowledge or not is a part of the overall KDD process. But in order to evaluate the knowledge discovered in the KDD process, the models are usually tested using a test dataset to evaluate the performance of the models.

1.3 Data mining challenges

Although traditional data mining techniques have achieved great success, they also encountered practical difficulties in meeting challenges posed by new type of datasets. Pang-Ning Tan *et al.* had identified the following challenging problems: mainly high dimensionality, heterogeneous and complex data and non-traditional analysis (Tan, et al., 2005):

1.3.1 High Dimensionality

A data set may have hundreds or even thousands of attributes (features) compared to several a few decades ago. One extreme exists in bio-informatics: microarray, consisting of an array series of thousands of microscopic spots, each containing a specific DNA sequence, has enabled gene expression in tens of thousands of features. For most of data analysis algorithms, the computational complexity increases exponentially as the dimensionality (the number of features) increases (Liu & Motoda, 2007).

1.3.2 Heterogeneous and complex data

Traditional datasets often contain attributes of the same type, either continuous numerical or categorical. As datasets and data mining applications in industry, business, science, engineering, bio-informatics, medicine and other fields have grown, the need for techniques that can deal with heterogeneous attributes has also risen. Recent years have also seen the emergence of more complex data objects.

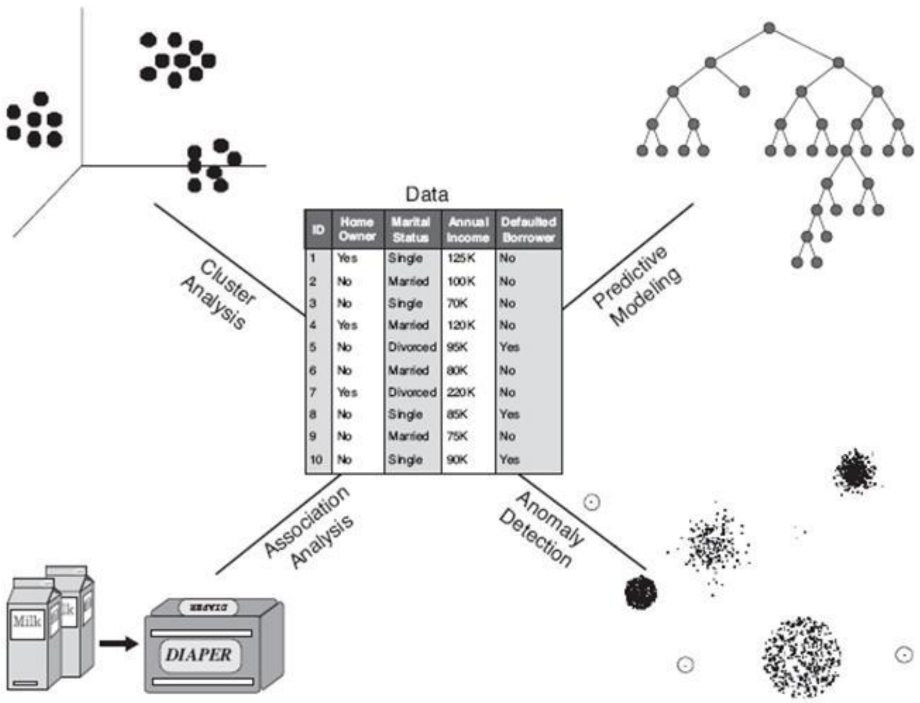
1.3.3 Non-traditional analysis

The traditional analysing approach is based on a hypothesize-and-test paradigm. In other words, the experiment design and data analysis are both related to the hypothesis. But this process can be extremely labour-intensive and could even stop working if the number of hypothesis and test paradigm is in the thousands (Cao, Wegman, & Solka, 2005). As a result, there is no hypothesis-and-test procedure in the data mining and the process of hypothesis generation and evaluation is automatically incorporated in the algorithms.

1.4 Data mining tasks

Data mining tasks are generally divided into two major categories: predictive and descriptive. The former aims to predict the value of a particular attribute (target or dependent variable) based on values of other attributes and the latter will derive patterns (correlations, trends, clusters, trajectories and anomalies) which can typically denote the underlying relationships in data (Tan, et al., 2005).

Figure 1-2 shows four of the core data mining tasks. Predictive modelling and cluster analysis are the main tasks for the datasets analysed in this thesis.



ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	80K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Figure 1-2: Four of the core data mining tasks (excerpt from Tan, et al., 2005)

Association analysis aims at discovering patterns that lay behind the datasets and describing associated features in the data. The patterns discovered can typically be represented either in the form of implication rules or feature subsets (Lawrence, Kudyba, & Klimberg, 2008). The goal of association analysis is to extract the most interesting and representative patterns in an effective manner (Dunne, 2007).

Anomalies are also referred to as outliers. Anomaly detection, sometimes referred to as outlier detection, involves detecting instances in a given dataset that do not conform to a pre-defined class set or identifying cases that are unusual with the dataset that is seemingly homogeneous. Anomaly detection is a very important tool in a variety of domains, such as fraud detection, fault detection, system health monitoring etc.

Predictive modelling involves building a model to predict the target variable as a function of the explanatory variables (Larose, 2006). Two typical types of predictive modelling tasks are classification and regression. The former is used for discrete target variables and the latter is used for continuous target variables.

Classification is one of the most common tasks in the data mining domain discussed in this thesis. It consists of examining the attributes of a newly presented instance and assigning it to one of a predefined set of classes. The classification task is characterized by a well-and-pre-defined definition of the classes, and a training data set consisting of pre-classified examples (Hand, Smyth, & Mannila, 2001). The task of classification is to build a model that can be applied to unclassified data instance in order to classify this unseen instance (Larose, 2006).

Cluster analysis seeks to segment heterogeneous observations into a number of more homogeneous subgroups or clusters. Usually, the observations are grouped together based on self similarities, so observations that belong to the same cluster are most similar to each other than observations that belong to other clusters (Witten & Frank, 2005).

A class is a set of data samples with some similarity or relationship and all samples in this class are assigned the same class label to distinguish them from samples in other classes. A cluster is a collection of objects which are similar locally. Clusters are usually generated in order to further classify objects into relatively larger and meaningful categories (Wang & Fu, 2005). What distinguishes clustering from classification is that the clustering process doesn’t assign and rely on a predefined class set. But in classification, each new instance is classified into a predefined class through some intelligent systems techniques which are trained with some predefined instances. Clustering can be employed for dealing with data which have not been put into classes. Some classification methods cluster data into small groups first before proceeding to classification (Berry & Linoff, 2004). In the context of intelligent systems technique, clustering can be regarded as a kind of unsupervised learning and classification can be regarded as supervised learning.

This thesis will focus mainly on the following data mining tasks: classification and clustering, prediction and rule extraction.

1.5 Intelligent data mining techniques

Data mining may be chosen to deal with datasets which contain a huge amount of data and may be ambiguous and even partly conflicting. In order to make the data and information mining process more robust, intelligent systems techniques for searching and learning the data and information require tolerance toward imprecision, uncertainty and exceptions. The intelligent systems incorporating these intelligent systems techniques will have approximate reasoning capabilities and be capable of handling partial truth (Han & Kamber, 2006). These are also some typical characteristics of soft computing.

Since there is no generic term for intelligent computing techniques used in data mining. In this thesis, the term Intelligent Systems (ISs) will be used interchangeably with intelligent systems techniques and soft computing, although under most circumstances, Intelligent Systems (ISs) is preferred. Soft computing is an important technique domain in the area of data mining and intelligent and knowledge-based systems (Karray & Silva, 2005). Soft computing differs from conventional (hard) computing because of its tolerance of imprecision, uncertainty, partial truth and approximation. So the main principle of soft computing is to exploit the tolerance for imprecision, uncertainty, approximation and partial truth in order to achieve tractability, robustness and low-cost solution (Han & Kamber, 2006; Venugopal, et al., 2009).

The principal computing paradigms of soft computing are fuzzy logic, neural networks, probabilistic reasoning and genetic algorithms. These methodologies of soft computing are complementary and cooperative rather than competitive and exclusive (Engelbrecht, 2002). A problem can be solved most effectively by combining different techniques such as fuzzy logic, neural networks, genetic algorithms and probabilistic reasoning rather than using them exclusively and individually. For example, ANFIS, which incorporates fuzzy logic and neural networks, has proved to be successful for solving many practical problems. These techniques can be viewed as a foundation component for intelligent computing in data mining techniques.

1.6 Research objectives

Each of the IS techniques contributes a distinct methodology for addressing problems in its domain. This may be done in a cooperative, rather than a competitive, manner. The result is a more intelligent and robust system providing a human-interpretable, low-cost, approximate solution, as compared to traditional techniques (Ian & Jacek, 2000).

The unique contribution of this thesis is in the implementation of a hybrid IS DM technique, which incorporates fuzzy logic, decision tree and neural networks, for solving novel practical problems, the detailed description of this technique (Optimized Weighted Fuzzy Decision Tree, OWFDT), and the illustrations of several applications solved by Optimized Weighted Fuzzy Decision Tree.

The primary objective of this work is to design an IS system that can be applied effectively to some DM tasks such as those listed in Figure 1-2. The devised intelligent systems will also focus on solving the following issues particularly (Wang & Fu, 2005):

- Achieve a high and acceptable classification or prediction accuracy;
- Reduce the dimension of the data set and ease the computational complexity;
- Extract some rules if applicable

The thesis also aims to explore the possibilities of applying this hybrid IS DM technique to material, biological and civil engineering applications, since the problems in these fields are quite complex and the datasets available are in massive quantities. This thesis will explore the solution of such problems using newly-proposed systematic approach, OWFDT. The thesis will also conduct a benchmark study between the proposed technique and other techniques to test the performance of the newly-proposed technique.

1.7 Thesis outline

Chapter one is a brief overview of data mining, including its concept, its challenges and tasks. The research objective and overall structure of the thesis is also listed in this chapter.

Chapter two gives a brief introduction to the theoretical background of the techniques used in the thesis including fuzzy logic, neural networks (self-organizing map, multiple-layer perceptron), decision tree and some other hybrid intelligent systems, such as Adaptive Neuro-Fuzzy Inference Systems, Fuzzy ARTMAP etc.

Chapter three introduces the Optimized Weighted Fuzzy Decision Tree, which will be used in the following chapters. A detailed description is presented first, and this is followed by the simulation results after applying OWFDT to three datasets.

Chapter four analyzes the datasets collected from material science and clustered the data in order to predict the possible composite material failure mechanism. The OWFDT is also used to construct a classifier for prediction and classification of those failure mechanisms.

In chapter five an eye bacteria dataset collected using electronic nose is analyzed using bagging to Multi-Layer Perceptron and Decision Tree. OWFDT is also adopted as an intelligent systems technique to predict the eye bacteria species.

Chapter six is concerned with the analysis of a wave overtopping dataset and conduction of a benchmark study between some well-established hybrid intelligent techniques and OWFDT.

Chapter seven presents the conclusions and suggestions for further study.

References

Berry, M. J. A., & Linoff, G. S. (2004). *Data mining techniques: for marketing, sales, and customer relationship management*: Wiley.

Cao, C. R., Wegman, E. J., & Solka, J. L. (2005). *Handbook of Statistics: Data Mining and Data Visualization*: Elsevier North-Holland.

Dunne, R. A. (2007). *A Statistical Approach to Neural Networks for Pattern Recognition*: John Wiley & Sons,.

Engelbrecht, A. P. (2002). *Computational Intelligence: An Introduction*. Chichester: John Wiley.

Gaber, M. M. (2010). *Scientific Data Mining and Knowledge Discovery: Principles and Foundations*: Springer-Verlag.

Han, J., & Kamber, M. (2006). *Data mining: concepts and techniques*. Amsterdam: Boston: Elsevier: San Francisco: CA Morgan Kaufmann.

Hand, D. J., Smyth, P., & Mannila, H. (2001). *Principles of data mining*: MIT Press.

Ian, C., & Jacek, M. Z. (2000). *Knowledge-based neurocomputing*: MIT Press.

Kantardzic, M. (2002). *Data Mining: Concepts, Models, Methods, and Algorithms*: Wiley-IEEE Press.

Karray, F. O., & Silva, C. W. D. (2005). *Soft Computing and Intelligent Systems Design: Theory, Tools and Applications*: Addison-Wesley.

Larose, D. T. (2005). *Discovering Knowledge in Data: An Introduction to Data Mining*: John Wiley & Sons, Inc.

Larose, D. T. (2006). *Data Mining Methods and Models*. New Jersey: John Wiley & Sons, Inc.

Lawrence, K. D., Kudyba, S., & Klimberg, R. K. (2008). *Data Mining Methods and Application*: Auerbach Publications Taylor & Francis Group.

Liu, H., & Motoda, H. (2007). *Computational Methods of Feature Selection (Chapman \& Hall/Crc Data Mining and Knowledge Discovery Series)*: Chapman \& Hall/CRC.

Tan, P.-N., Steinbach, M., & Kumar, V. (2005). *Introduction to Data Mining, (First Edition)*: Addison-Wesley Longman Publishing Co., Inc.

Venugopal, K. R., G.Srinivasa, K., & Patnaik, L. M. (2009). *Soft Computing for Data Mining Applications*. New York: Springer-Verlag.

Wang, L., & Fu, X. (2005). *Data Mining with Computational Intelligence (Advanced Information and Knowledge Processing)*: Springer.

Witten, I. H., & Frank, E. (2005). *Data mining : practical machine learning tools and techniques*. Amsterdam Boston Mass: Elsevier/Morgan Kaufman.

Chapter II: Introduction to Intelligent Systems Techniques

Chapter one has introduced the concept of data mining and outlined the research objectives of the thesis. The current chapter will provide a solid theoretical background of some intelligent systems techniques which are widely used in the subject of data mining. This chapter will concentrate on these techniques including artificial neural networks (NN), fuzzy logic (FL) and decision tree (DT) which are core techniques that OWFDT is based on. The last section of this chapter will introduce some hybrid intelligent systems techniques.

2.1 Artificial neural network introduction

Neural networks (NNs) first started to be developed in the 1940s, motivated by, for example, an attempt to simulate the human brain on computers. It has been successfully applied to prediction problems, pattern recognition, classification and optimization problems (Hagan, Demuth, & Beale, 1996; Haykin, 1999).

Usually, ANNs consist of simple parallel interconnected and usually adaptive processing elements or neurons whose connectivity mimics the neurobiological system (Chow, 2007). These neurons are typically distributed over several layers. Forward Neural Network (FNN) is one of the most popular network topologies. Normally, the neurons in a FNN are arranged into three layers as shown in Figure 2-1. Neurons in each layer perform a different activity as described in the following three steps (Kartalopoulos, 1997):

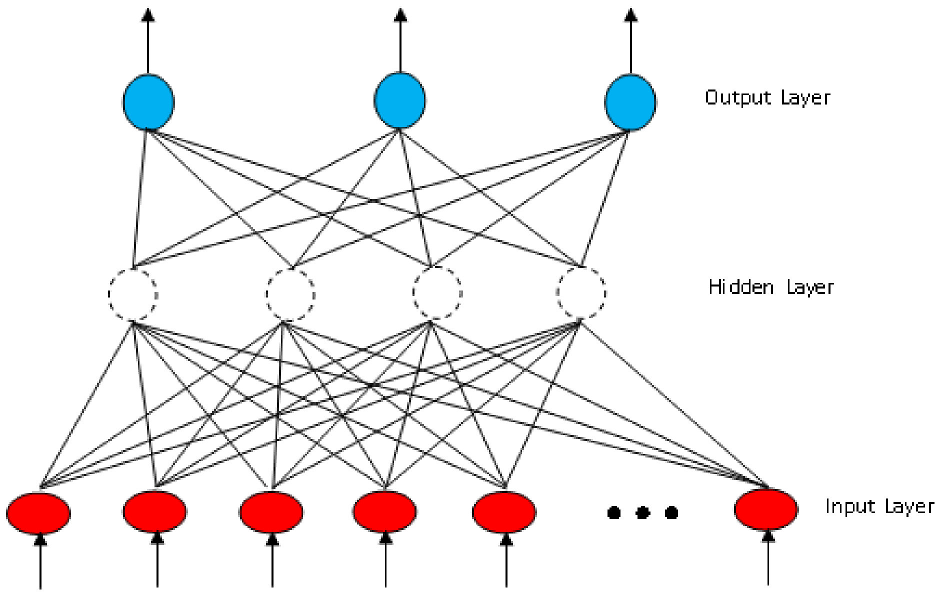


Figure 2-1: Basic architecture of a typical neural network

(1) The input neurons receive information in the form of input values. They transfer the data to the next layer. Neurons in the input layer perform no neural function.

(2) Hidden neurons receive the output from the input neurons or other hidden neurons through the connections. Each connection between input neurons and hidden neurons has a weight by which to multiply the signal before it enters the hidden neurons. A hidden neuron usually receives more than one signal. The signals will be combined and transferred to an appropriate activation function. The output of the activation function will then be usually transferred to output neurons but sometimes via hidden neurons (as shown in Figure 2-2). Figure 2-2 shows the structure and the function of a neuron.

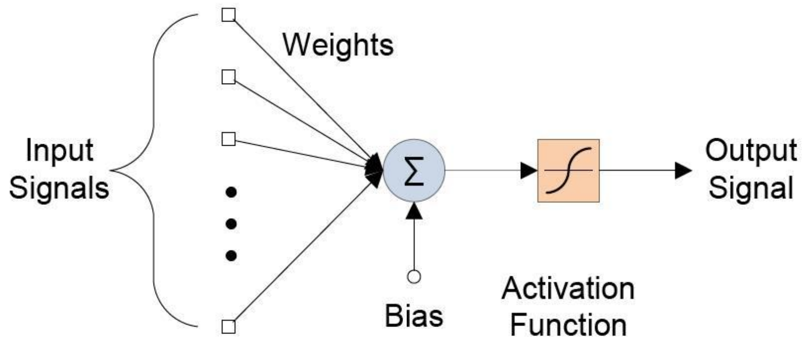


Figure 2-2: An example artificial neuron

(3) Output layer neurons behave similarly to the ones in the hidden layer except that the pattern of the outcome from output neurons may be interpreted as the outcome of using the NN to process the input data.

Training the system is the key to determining whether or not the NN will be able to process the input data appropriately. In a supervised NN the training process allows NNs to compare the output result with the desired output for every training input. The difference between the generated output and the desired output is called the “error”. The NN will adjust the weight of each connection in such a way as to reduce the error (Du & Swamy, 2006). At the beginning, the NN might produce a large error. However, if the problem can be solved by NN then, after several cycles of training with the set of data, the error should reduce.

Multilayer Perceptron (MLP) is one of fully connected FNN.

2.1.1 Multilayer Perceptron (MLP)

The multilayer perceptron (MLP) is able to learn arbitrarily complex non-linear relationships. Usually a three-layer feed-forward network, MLP is the most popular type of ANN in practical use. Figure 2-1 shows the structure of an MLP, in which the processing elements are organized in a regular architecture (or topology) of three distinct groups of neurons (input, hidden and output layer) interconnected using unidirectional weights. The number of input nodes is typically set to correspond with the number of dimensions in the problem. The number of neurons in the hidden layer is determined experimentally and the number of output classes in the analyzed dataset determines the number of outputs.

Each neuron in MLP performs a weighted sum of inputs and transforms it using a nonlinear activation function (e.g. sigmoid transfer function) (Haykin, 1999). An MLP is able to learn from data by adjusting the weights in the network using a gradient descent technique known as back-propagation (BP) of errors. MLP adopts a supervised training process in which both the training vectors and their associated targets are presented to the network. During the forward pass, the MLP processes all of its input in a feedforward manner; the output of each neuron is calculated and feeds the next layer through to the output. Let us consider a neuron *h*, with *j*th input vector which consists of *n* elements x_{ij} $(i = 1, \dots n)$, the summation function a_{jh} accumulates the sum of the products of the input signals x_{ij} with associated weights w_{ih}. It assumes a fixed weight,θ_h, which is then transformed by the activation function *f(.)* (e.g. sigmoid) to produce the single output z_{jh}, the overall computations follows that given in equation (1):

$$z_{jh} = f(a_{jh}) = \frac{1}{1 + \exp(-a_{jh})} = f\left(\sum_{i=1}^{n}(w_{ih}x_{ij} - \theta_h)\right) \tag{1}$$

$$(i = 1, \dots n)$$

The *error* is then calculated by determining the difference between the actual generated output z_{jh} and the target output t_{jh} using the expression $\delta_{jh} = z_{jh} - t_{jh}$. The error term is often called delta and hence when the delta learning rule is used, the component difference expression becomes $\delta_{jh} = (z_{jh} - t_{jh})(1 - t_{jh})$. In the *backward pass*, the stochastic approximation procedure back-propagates this error to adjust the weight values during each presentation of the *j*th training sample on each iteration (or epoch) τ. Various training algorithms can be used to improve the operation of BP (Hagan, et al., 1996):

- BP gradient descent with momentum (BPGDM). The new set of weights $w_{jh}^{(\tau)}$ comprise of a combination of the old weight values, $w_{jh}^{(\tau-1)}$ (from the previous epoch) and a weight update or delta, $\Delta w_{jh}^{(\tau)}$ (see equation (2)). The change in weights here is based on two parameters; 1) η, the learning rate, a small positive number (the default is generally 0.9) that determines the rate of convergence to a solution of minimum error, and 2) μ, the momentum term, a small positive number (default is generally 0.5) is often added to improve the speed and stability of the learning.

$$w_{jh}^{(\tau)} = w_{jh}^{(\tau-1)} + \Delta w_{jh}^{(\tau)} = w_{jh}^{(\tau-1)} - \eta \delta_{jh} z_{jh}^{(\tau)} + \mu \Delta w_{jh}^{(\tau-2)} \qquad (2)$$

- Faster BP training. BPGDM is often too slow for practical problems and other high performance algorithms were introduced, such as conjugate gradient, quasi-Newton and Levenberg-Marquardt (BPLM) (Chow, 2007; Ham & Kostanic, 2000). The BPLM was designed to approach a second-order training speed using the Jacobian matrix $\mathbf{J}$, which contains the first derivatives of the network errors with respect to the weights and biases. The algorithm uses equation (3) where μ is an updating (decreasing) scalar.

$$w_{jh}^{(\tau)} = w_{jh}^{(\tau-1)} - [\mathbf{J}^{\mathrm{T}}\mathbf{J} + \boldsymbol{\mu}\mathbf{R}_j]^{-1}\mathbf{J}^{\mathrm{T}}\delta_{jh} \qquad (3)$$

Using BP, the weights and biases associated with the neurons are modified to minimize the mapping error, when these stabilise, the network is said to be trained (the total sum squared error can be used to measure the network performance). The updating procedure is repeated for a number of epochs until the network error has fallen to a small constant error level. Once the network is trained, it can be used to predict the membership of unseen samples in a validation set. The classification of new patterns is performed by propagating the new patterns through the network and the output neuron with the highest score indicates the predicted class.

2.2.2 Self-organizing map (SOM)

Here Self-Organizing Map (SOM) was chosen as the structure of neural networks for analyzing the material data set in the thesis, because it is an unsupervised learning technique suitable for datasets with no pre-defined classes.

The SOM algorithm was developed by Kohonen to transform a data set of arbitrary dimensions into a one or more dimensional discrete map(Kohonen T, 1990). According to Kohonen, SOM's are connectionist techniques capable of generating topology which can preserve clustering information. An example of the network architecture of a Kohonen SOM is shown in Figure 2-3 (Godin, Huguet, & Gaertner, 2005). It consists of a two dimensional array of *m x m* discrete units and the Kohonen network associates each of the vector inputs to a representative output.

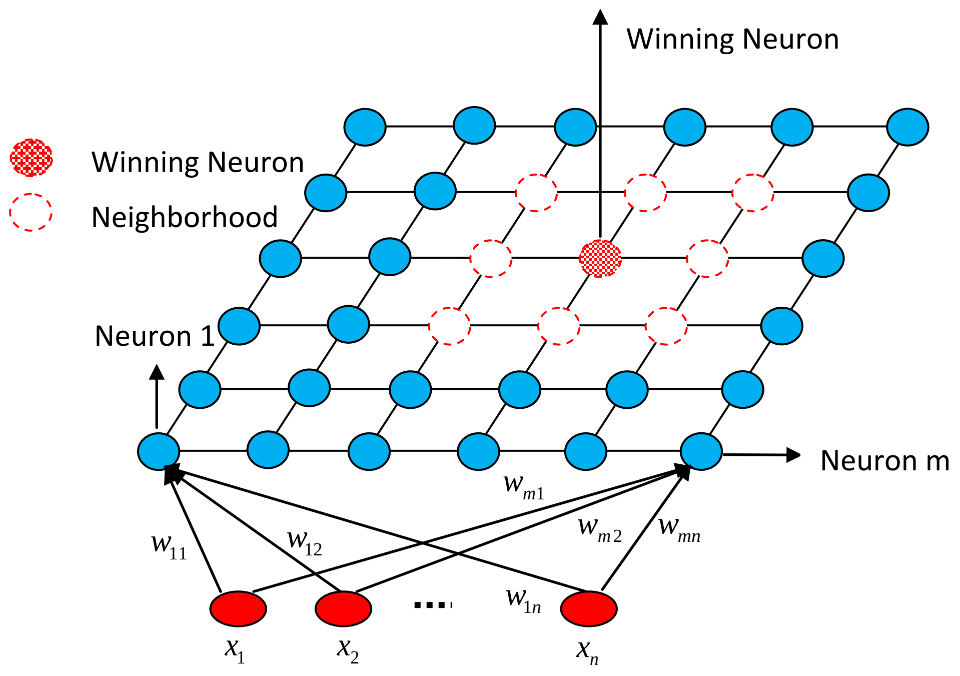


Figure 2-3: Schematic of a Self Organizing Map topology

Suppose w_{ij} are the components of the weight vector W_j, connecting the inputs i to output node j; the x_i are components of the input vector X, the output of the neuron is the quadratic (Euclidean) distance d_j between the weight vector and the input vector (see Figure 2-3). The description of each step to use the data to train the SOM is summarized as follows (Ham & Kostanic, 2000):

- Initialize all the weights to random values between 0 and 1.
- Randomly select an input vector X; present it to all the neurons of the network and evaluate the corresponding quadratic distance output d_j according to the following equation:

$$d_j = \|X - W_j\|^2 = \sum_{i=1}^{n} (w_{ij} - x_i)^2 \qquad (4)$$

where *n* is the number of input vector components.

- Select the neuron with the minimum output d_j as the wining neuron, i.e. the nearest vector to the input vector. Let j^*denote the index of the winner, the minimum output will be:

$$d_{j^*} = \min_{j\in[1,2,\ldots,m\hat{}2]} \left\| X - W_j \right\|^2 \qquad (5)$$

where *m x m* is the number of neurons.

- Update the weight of the wining neuron according to Equation (6).

$$w_{ij^*}(\tau+1) = w_{ij^*}(\tau) + \eta(\tau)[x_i(\tau) - w_{ij^*}(\tau)] \qquad (6)$$

where τ is the learning iteration count and η is the gain term.

- The neighbors of the wining neuron, defined by the neighborhood function $N(j^*)$ (the neighborhood function $N(j^*)$ defines how many neurons in the neighborhood of the winning one will be updated for each learning input) are also updated following Equation (7) and (8):

$$w_{ij}(\tau+1) = w_{ij}(\tau) + \eta(\tau)[x_i(\tau) - w_{ij}(\tau)] \qquad (7)$$

If $\in N(j^*)$, neighborhood of j^*

$$w_{ij}(\tau+1) = w_{ij}(\tau) \qquad (8)$$

If $\notin N(j^*)$, neighborhood of j^*

- Repeat the learning process until all the input vectors *X* have been used at least once.

2.2 Fuzzy logic

The concept of Fuzzy Logic (FL) was conceived by Lotfi Zadeh, a professor at the University of California at Berkley, and presented as a way of processing data by allowing partial set membership rather than crisp set membership or non-membership (Babuska, 1998). It is basically a multi-valued logic that allows intermediate values to be defined between conventional evaluations.

2.2.1 Fuzzy sets

A fuzzy set is a set without a crisp and clearly-defined boundary or without binary membership characteristics. Unlike an ordinary set where each object (or element) either belongs or does not belong to the set, fuzzy set can contain elements with only a partial degree of membership. In other words, there is a 'softness' associated with the membership of elements in a fuzzy set (Kartalopoulos, 1997). An example of a fuzzy set could be 'the set of tall people.' There are people who clearly belong to the above set and others that cannot be considered as tall. Since the concept of 'tall' is not precisely defined (for example, >2m), there will be a gray zone in the associated set where the membership is not quite obvious. As another example, consider the variable 'temperature'. It can take a fuzzy value (e.g., cold, cool, tepid, warm, and hot). Each fuzzy value such as 'hot' is called a fuzzy descriptor. It may be represented by a fuzzy set because any temperature that is considered to represent 'hot' belongs to this set and any other temperature does not belong to the set. A crisp set or a precise temperature interval such as 25 °C to 30 °C cannot be indentified to represent warm temperatures.

2.2.2 Membership function

A fuzzy set can be represented by a membership function. This function indicates the grade (degree) of membership within the set, of any element of the universe of discourse (e.g. the set of entities over which certain variables of interest in some formal treatment may range). The membership function maps every element of the universe to numerical values in the interval [0, 1]. Specifically,

$$\mu_A(x): X \rightarrow [0, 1] \tag{9}$$

where $\mu_A(x)$ is the membership function of the fuzzy set *A* in the universe *X*. Stated in another way, fuzzy set *A* is defined as a set of ordered pairs (H. Li, Philip, & Huang, 2000):

$$A = \{\big(x, \mu_A(x)\big); x \in X, \mu_A(x) \in [0, 1]\} \tag{10}$$

The membership function $\mu_A(x)$ represents the grade of possibility that an element *x* belongs to the set *A*. It is a curve that defines how each point in the input space is mapped to a membership value. A membership function value of zero indicates that corresponding element is definitely not an element of the fuzzy set. A membership function value of unity implies that the corresponding element is definitely an element of the fuzzy set. A grade of membership greater than 0 and less than 1 corresponds to a non-crisp (or fuzzy) membership, and the corresponding elements fall on the fuzzy boundary of the set. The closer the $\mu_A(x)$ is to 1 the more the *x* is considered to belong to *A*, and similarly, the closer it is to 0 the less it is considered to belong to *A*.

The following is a summary of the characteristics of fuzzy set and membership function (Tsoukalas & Uhrig, 1996):

- Fuzzy sets describe vague concepts (e.g., fast runner, hot weather, and weekend days).
- A fuzzy set admits the possibility of partial membership in it. (e.g., Friday is sort of a weekend day, the weather is rather hot).
- The degree to which an object belongs to a fuzzy set is denoted by a membership value between 0 and 1. (e.g., Friday is a weekend day to the degree 0.8).
- A membership function associated with a given fuzzy set maps an input value to its appropriate membership value.

2.2.3 Considerations of fuzzy decision-making

Instead of being represented as a set of complex, nonlinear differential equations, a complex system can be represented by a fuzzy knowledge base as a simple set of input-output relations (or rules). These rules contain fuzzy terms which are common linguistic expressions of knowledge about a practical situation such as small, fast, high, and very. Designing the rule base is an important step in the development of a fuzzy knowledge-based system. Inferences can be made either by matching a set of available information (or observation, or data) with the rule base of the system or by matching data with individual rules and then combining (aggregating) these individual rule-based inferences (say, using the max operation) (Karray & Silva, 2005). Combining individual rule-based inferences gives identical inferences to what one would get by applying the compositional rule of inference, albeit more conveniently.

A typical rule consists of the antecedent part(s) and the consequent part. The antecedent part of a rule corresponds to the input variables (input space) of the fuzzy decision-making system. The input variables may be connected by AND connectives; for example, 'IF Temperature is S*i* AND Humidity is D*j*, AND… ' Here, S*i* and D*j* are fuzzy states of the corresponding fuzzy variables. Different rules in the rule base will involve different fuzzy states of the antecedent variables. Each antecedent variable is assumed to be orthogonal to (independent of) other antecedent variable, and occupies a disjoint subspace of the input space (Nauck, Klawonn, & Kruse, 1997). But there exists some overlap between various fuzzy states of a particular antecedent variable, as provided in a fuzzy context where, in the real world, changes are not abrupt but rather smooth.

The inferred decision or conclusion is presented by the consequent part of a rule. Since, with loss of generality, multiple decision variables can be separated into multiple rules with single decision variables, it is practical to only consider rules with single consequent variables.

2.2.4 Extensions to fuzzy decision-making

A typical fuzzy rule explored so far can be formed as:

$$IF\ x\ is\ A_i\ AND\ IF\ y\ is\ B_i\ THEN\ z\ is\ C_i \tag{11}$$

where A_i, B_i and C_i are fuzzy states governing the *i*-th rule of the rule base. Here, the knowledge base (rule) is represented as fuzzy protocols of the equation (11) and represented by membership functions for A_i, B_i and C_i, and the inference is obtained by applying the compositional rules of inference. Such rules are referred to as the Mamdani approach (Mamdani system or Mamdani model) (N. K. Kasabov, 2002), named after the person who proposed the application of this approach. The consequent part (result) of the rule is a fuzzy membership function, which typically has to be defuzzified for use in practical tasks.

But there are advantages to consider crisp or special shape membership functions for the consequent part as well. Such variation has resulted in Sugeno model (or Takagi-Sugeno-Kang model or TSK model), where the output variable is presented in terms of a functional relation of the inputs (N. K. Kasabov, 2002). Such rules (with crisp functions as the consequent) are typically written as:

$$IF\ x\ is\ A_i\ AND\ IF\ y\ is\ B_i\ THEN\ c_i = f_i(x, y) \tag{12}$$

For rule *i*, where f_i is a crisp function of the input variables (antecedent) *x* and *y*. When f_i is a constant, rules of equation (12) constitutes a zero order Sugeno model and when f_i is a first order polynomial, it is called a first-order Sugeno model. Note that the antecedent parts of this rule are the same as for the Mamdani model in equation (11), where A_i, and B_i are fuzzy sets whose membership functions are functions of *x* and *y*, respectively. But the consequent part is a crisp function of the condition variables which differs from the Mamdani model (Tsoukalas & Uhrig, 1996). The inference $\hat{c}(x, y)$ of the fuzzy knowledge-based system of Sugeno model is obtained directly as a crisp function of the input variables *x* and *y*, as follows.

- First, a weighting parameter $w_i(x, y)$ for rule *i* is obtained corresponding to the input membership functions, as in the case of Mamdani approach, by using either the 'min' operator or the 'product' operator. For example, using the 'min' operator we form

$$w_i(x, y) = \min[\mu_{A_i}(x), \mu_{B_i}(y)] \tag{13}$$

- Then the crisp inference $\hat{c}(x, y)$ is determined as a weighted average of the individual rule inferences (crisp) $c_i = f_i(x, y)$ according to

$$\hat{c}(x, y) = \frac{\sum_{i=1}^{r} w_i c_i}{\sum_{i=1}^{r} w_i} = \frac{\sum_{i=1}^{r} w_i(x, y) f_i(x, y)}{\sum_{i=1}^{r} w_i(x, y)} \tag{14}$$

where *r* is the total number of rules. For any data *x* and *y*, the knowledge-based action $\hat{c}(x, y)$ can be computed from equation (14), without requiring any defuzzification (Karray & Silva, 2005).

Mamdani type and Sugeno type of inference systems vary in the way outputs are determined. The Sugeno model is particularly useful when the consequents are described analytically through crisp functions, as in conventional crisp control, rather than linguistically. The Sugeno approach is commonly used in applications of direct control and in simplified fuzzy models. The Mamdani approach, even through popular in low-level direct control, is particularly appropriate for knowledge representation and processing in expert systems and in high-level (hierarchical) control systems (H. Li, et al., 2000).

2.2.5 Fuzzy C-means algorithm

Fuzzy C-means algorithm (FCM) is widely used as a clustering tool to find the cluster within a dataset. The objective of cluster analysis is to classify objects according to similarities among them, and organize a dataset into subsets. Usually, clustering techniques are considered to be among the unsupervised methods, they do not use prior information about groupings before clustering. Fuzzy C-means algorithm can detect the underlying structure in a data set, thus it can be used for classification and pattern recognition (Höppner, Klawonn, Kruse, & Runkler, 1999).

Different kinds of definitions of a cluster can be formulated, depending on the objective of the clustering. Usually we regard a cluster as a group of objects which bear more similarities to one another than to the members of other clusters. We can define similarity by means of a distance norm. It can be measured among the data vectors themselves, or as a distance from a data vector to some prototypical object (perspective center) of the cluster (Theodoridis & Koutroumbas, 1999). The prototypes are vectors of the same dimension as the data objects and are usually not known beforehand, and usually they are sought by the clustering algorithms heuristically with the partitioning of the data.

The Fuzzy C-means algorithm, also known as Fuzzy ISODATA, is a data clustering technique wherein each data point belongs to a cluster to some degree that is specified by a membership grade. This technique was originally introduced by Jim Bezdek in 1984 (Bezdek, Ehrlich, & Full, 1984). It provides a method that shows how to group data points that populate some multidimensional space into a specific number of different clusters. The FCM-based algorithms are in practice the most widely used fuzzy clustering algorithms.

The objective of the FCM clustering algorithm is to minimize an objective function called C-means functional (Lampinen, Laurikkala, Koivisto, & Honkanen, 2005), which is formulated as:

$$J(U,V) = \sum_{j=1}^{C}\sum_{i=1}^{N}(\mu_{ij})^m \left\|x_i - v_j\right\|^2 \tag{15}$$

where $X = \{x_1, x_2, \dots, x_N\}, x_i \in R^n$ represents a given set of feature data. $V = \{v_1, v_2, \dots, v_C\}$ are the cluster centers. $U = (\mu_{ij})_{N\times C}$ is a fuzzy partition matrix, in which each member μ_{ij} indicates the degree of membership between the data vector x_i and the cluster *j*. The values of matrix *U* should satisfy the following conditions

$$\mu_{ij} = [0,1], 1 \le i \le N;\ 1 \le j \le C \tag{16}$$

$$\sum_{j=1}^{C}\mu_{ij} = 1, 1 \le i \le N \tag{17}$$

The exponent $m \in [1,\ \infty)$ is the weighting exponent, which determines the fuzziness of the clusters. The larger the exponent weight *m* is, the fuzzier the partition matrix becomes. The most commonly used distance norm is the standard Euclidean distance $d_{ij} = \|x_i - v_j\|$, although Babuska suggests that other distance norms could produce better results (Babuska, 1998).

Minimization of the cost function *J (U, V)* is a nonlinear optimization problem, which can be achieved with the following iterative algorithm (Bezdek, et al., 1984):

Step 1: Initialize the membership matrix U with random values so that the conditions in Equations (16) and (17) are satisfied.

Choose appropriate exponent *m* and the termination criteria.

Step 2: Calculate the cluster centers *V* according to the equation:

$$v_j = \frac{\sum_{i=1}^{N} (\mu_{ij})^m \, x_i}{\sum_{i=1}^{N} (\mu_{ij})^m}, \forall\, j = 1,2, \dots, C \tag{18}$$

Step 3: Calculate the new distance norms:

$$d_{ij} = \|x_i - v_j\|, \forall\, i = 1,2, \dots, N, \forall\, j = 1,2, \dots, C \tag{19}$$

Step 4: Update the fuzzy partition matrix *U*:

If $d_{ij} > 0$ (indicating that $x_i \neq v_j$)

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C} (\frac{d_{ij}}{d_{ik}})^{\frac{2}{m-1}}} \tag{20}$$

Else $\mu_{ij} = 1$

Step 5: If the termination criterion has been met, stop.

Else go to Step 2.

We can see that the FCM algorithm is a simple iteration through Equations (18) to (20).

A suitable termination criterion will be used to evaluate the cost function (Equation 15) and to see whether it is below a certain tolerance value or if its improvement, compared to the previous iteration, is below a certain threshold. We can also use the maximum number of iteration cycles as a termination criterion.

2.3 Decision Tree

2.3.1 Introduction to the decision tree

A decision tree partitions the input space (also known as the feature or attribute space) of a dataset into mutually exclusive regions, each of which is assigned a label, a value or an action to characterize its data points. It is a tree structure consisting of nodes and branches, whose nodes are designated as an internal or terminal node. An internal node is one that splits into two children, while a terminal node, also known as a leaf, does not have any children and is associated with a label or value that characterizes the given data. Figure 2-4 shows a typical decision tree and its function to partition the space into different subspaces.

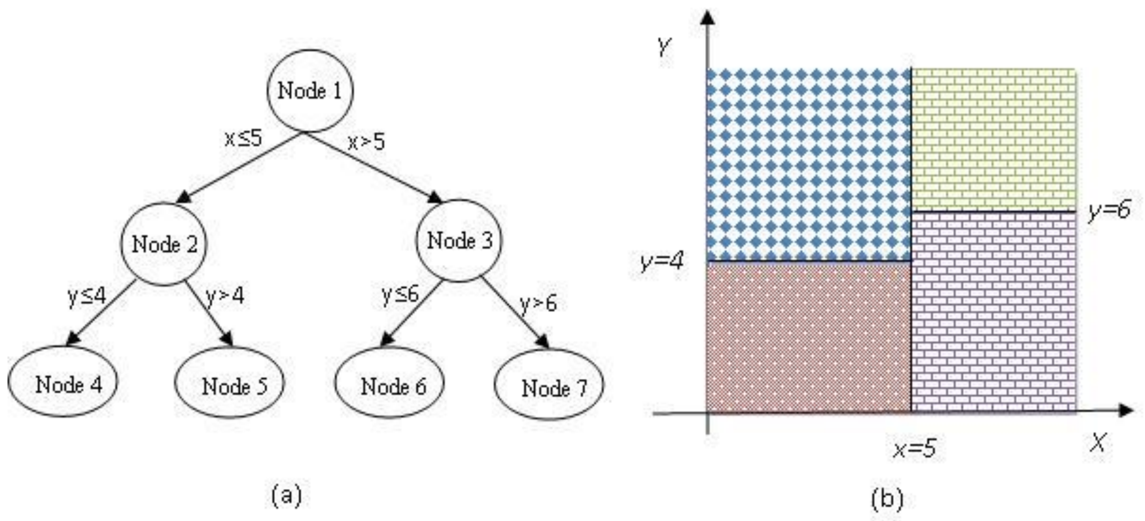


Figure 2-4: the Architecture of a basic Decision Tree

Decision trees used for classification problems are often called classification trees, and each terminal node contains a label that indicates the predicted class of a given feature vector. In the same vein, decision trees used for regression problems are often called regression trees, and the terminal node labels may be constants or equations that specify the predicted output value of a given input vector. To construct an appropriate decision tree, an algorithm first grows the tree extensively based on a sample (training) dataset, and then prunes the tree back based on a minimum cost-complexity principle. The result is a sequence of trees of various sizes; the final tree selected is the tree that performs best when another independent (test) dataset is presented (J.-S. R. Jang & Sun, 1997). Generally, the algorithm does this through two phases: tree growing and tree pruning, which will be explained in the following sections.

2.3.2 How to build a tree

Usually, there are exponentially many decision trees that can be constructed from a given set of attributes. Finding the optimal tree which is more accurate than others is computationally infeasible because of the exponential size of the searching space. Nevertheless, efficient algorithms have been developed to induce a reasonably accurate, albeit suboptimal, decision tree in a reasonable amount of time. These algorithms usually employ a greedy strategy that grows a decision tree by making a series of locally optimum decisions about which attribute to use for partitioning the data. One such algorithm is Hunt's algorithm, which is the basis of many existing decision tree induction algorithms, including ID3, C4.5, and CART (Classification and Regression Tree algorithm). In Hunt's algorithm, a decision tree is grown in a recursive fashion by partitioning the training records into successively purer subsets (P.-N. Tan, Steinbach, & Kumar, 2005).

Let D_t be the set of training records that are associated with node t and $y = \{y_1, y_2, \dots, y_c\}$ be the class labels. The following is a recursive definition of Hunt's algorithm.

Step 1: if all the records in D_t belong to the same class y_t, then t is a leaf node labeled as y_t.

Step 2: if D_t contains records that belong to more than one class, an attribute test condition is selected to partition the records into smaller subsets. A child node is created for each outcome of the test condition and the records in D_t are distributed to the children based on the outcomes. The algorithm is then recursively applied to each child node.

2.3.3 Design issues of decision tree induction

A learning algorithm for inducing DTs must address the following two issues:

- How to choose the best criteria to split the training instances. Each recursive step of the tree-growing process must select an attribute test condition to divide the records into smaller subsets; to implement this step, the algorithm must provide a method for specifying the test condition for different attribute types as well as an objective measure for evaluating the goodness of each test condition (Ian & Jacek, 2000).
- What the stopping rule is to terminate the splitting procedure. A stopping condition is needed to terminate the tree-growing process. One possible choice is to continue splitting internal nodes until each one contains observations from the same class, in which case some nodes might have only one observation in the node. Another option is to continue splitting nodes until there is some maximum number of observations left in a node or the terminal node is pure (all observations belong to one class).

2.3.4 Measures for selecting the best split

Many measures can be used to determine the best way to split the training instances. These measures are defined in terms of the class distribution of the instances before and after splitting.

Let $p(i|t)$ denote the fraction of instances belonging to class *i* at a given node t. we sometimes omit the reference to node t and express the fraction as p_i. In a two class problem, the class distribution at a node can be written as (p_0, p_1), where $p_0 + p_1 = 1$.

The measures developed for selecting the best split are often based on the degree of impurity of the child nodes. The smaller the degree of impurity, the more skewed the class distribution. For example, a node with class distribution (0,1) has zero impurity, whereas a node with uniform class distribution (0.5, 0.5) has the highest impurity. Three of the most widely accepted impurity measures are (Umanol, et al., 1994):

$$Entropy(t) = -\sum_{i=1}^{C} p(i|t) log_2\, p(i|t) \tag{21}$$

$$Gini(t) = 1 - \sum_{i=1}^{c} p(i|t)^2 \tag{22}$$

$$Classification\ error(t) = 1 - \max_i [p(i|t)] \tag{23}$$

where *C* is the number of classes and $0log_0 = 0$ is defined in entropy calculations.

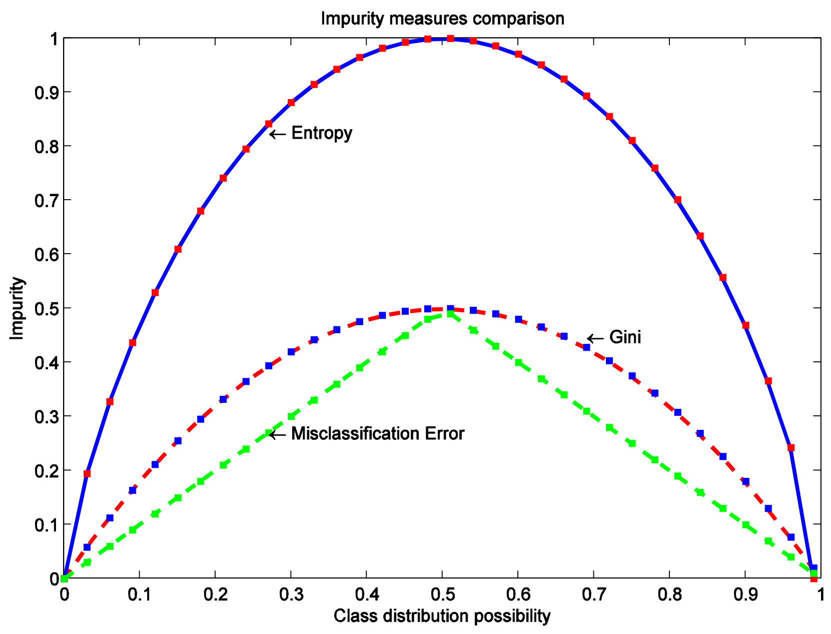


Figure 2-5: A comparison of three different impurity measures

Figure 2-5 shows a comparison of the values of the impurity measures for binary classification problems. It shows that all three measures attain their maximum value when the class distribution is uniform and achieve the minimum values for the measures when all the records belong to the same class.

2.3.5 Pruning the tree

The decision trees, as some other classifiers, may be subject to problems such as overfitting problem. As the number of nodes of the decision tree increases, the tree will have fewer training and test error. However, once the tree becomes too large, its test error rate begins to increase even though its training error rate continues to decrease. So obviously, the size and the complexity of the decision tree have a determining effect on the overfitting, which makes the pruning process necessary to overcome this issue.

The pruning process will determine the ideal size and complexity of the decision tree which will produce the lowest generalization error. The most popular way to estimate the generalization error of the induced tree is to do so by using resubstitution error (similar to training error which is the misclassification error committed on training instances).

It is assumed that the training set is a good representation of the overall data. Consequently, the resubstitution estimate approach using the training error, otherwise known as resubstitution error, can be used to provide an optimistic estimate for the generalization error (Martinez & Martinez, 2007). Under this assumption, a decision tree induction algorithm simply selects the model that produces the lowest training error rate as its final model.

After a decision tree is grown to its maximum size, the pruning procedure will be introduced to trim the fully grown tree in a bottom up fashion. Trimming can be done by replacing a subtree with (P.-N. Tan, et al., 2005):

- a new leaf node whose class label is determined from the majority class of records affiliated with the subtree, or
- the most frequently used branch of the subtree.

The tree pruning step terminates when no further improvement is observed. Unlike pre-pruning (the tree-growing process is terminated before generating a fully grown tree that perfectly fits the entire training data) which can suffer from premature termination of the tree-growing process, post-pruning tends to give better results than pre-pruning because it makes the pruning process based on the fully grown tree.

2.3.6 Classify a new instance

In general, a decision tree is employed as follows: First, a datum (usually a vector composed of several attributes or elements) is presented to the starting node (or root node) of the decision tree. Depending on the result of a decision function used by an internal node, the tree will branch to one of the node's children. This is repeated until a terminal node is reached and a label or value is assigned to the given input data (Martinez & Martinez, 2007).

2.4 Other Hybrid intelligent systems techniques

Hybrid Neural fuzzy systems are based on an architecture which integrates in an appropriate parallel structure a neural network and a fuzzy logic based system (Tsoukalas & Uhrig, 1996). These two parts work as one synchronized entity.

Hybrid neuro-fuzzy systems have a parallel architecture, and exploit similar learning paradigms, as is the case for neural networks. The parallelism of the system can be viewed as a mapping of the fuzzy system into a neural network structure (N. K. Kasabov, 2002). In other words, each functional module of the fuzzy logic system corresponds to a particular layer of the neural network. The resulting system can be interpreted either as a neural network or a fuzzy logic system.

Hybrid neuro-fuzzy systems have the same architecture as that of traditional fuzzy systems except that a layer of hidden neurons performs each of the tasks of the fuzzy inference system. As such, a fuzzy logic inference system can be implemented as a five-layer neural network. This type of architecture is the most commonly used among neuro-fuzzy inference systems, and it uses the Sugeno-type fuzzy inferencing (J.-S. R. Jang & Sun, 1997).

2.4.1 ANFIS

The acronym ANFIS derives its name from *adaptive neuro-fuzzy inference system*. It was proposed by J S Jang (J. S. R. Jang, 1993) and has become a standard technique that has been widely used in many applications (J.-S. R. Jang & Sun, 1997; Lin & Lee, 1996). It uses a hybrid-learning algorithm to identify parameters for Sugeno-type fuzzy inference systems. It applies a combination of the least-squares method and the gradient descent method for training membership function (MF) parameters to emulate a given training dataset (Karray & Silva, 2005; Soyguder & Alli, 2009).

ANFIS is a multilayer feed forward network where each node performs a particular function on incoming signals. It is normally represented by a six-layer FNN as shown in Figure 2-6. To perform a desired input-output mapping, adaptive learning parameters are updated based on gradient learning rules (J. S. R. Jang, 1993; Soyguder & Alli, 2009). Both square and circle node symbols in Figure 2-6 are used to represents different properties of adaptive learning, among which the rule layer represents a set of fuzzy rules. The ANFIS model is one of the implementation of a first order Sugeno fuzzy inference system, and the rules are of the form:

$$R_p: If\ x_1\ is\ A_{1,p}\ and\ \ x_2\ is\ A_{2,p} \dots and\ \ x_n\ is\ \ A_{n,p} then\ o_p = \alpha_{0,p} + \alpha_{1,p}x_1 + \cdots + \alpha_{n,p}x_n \tag{24}$$

where x_i is *i*th input variable associated with a linguistic term in the antecedent part of the *p*th rule with $i = 1,2, \dots, n$ and $A_{i,p}$ is the linguistic label associated with it in that rule. $A_{i,p}$ has its associated fuzzy membership function given by $\mu_{A_{i,p}}(x_i)$. o_p is the consequent output of the *p*-th rule and $\alpha_{0,p}, \alpha_{1,p}, \dots, \alpha_{n,p}$ are the Sugeno parameters.

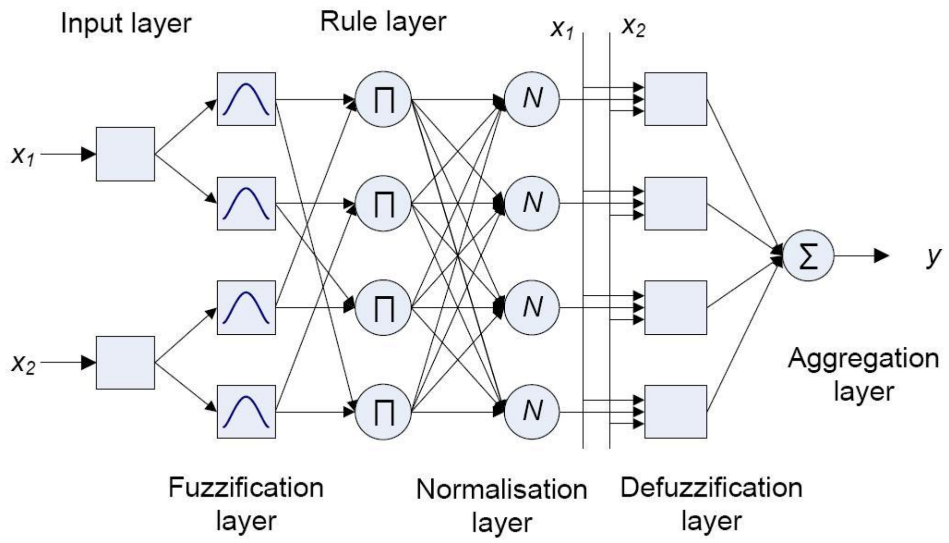


Figure 2-6: A six layer ANFIS structure

Specifically, ANFIS only supports Sugeno-type systems, and these must have the following properties:

- Be first or zeroth order Sugeno-type systems.
- Have a single output, obtained using weighted average defuzzification. All output membership functions must be the same type and either be linear or constant.

- Have no rule sharing. Different rules cannot share the same output membership function, namely every rule has a specific output function so the number of output membership functions must be equal to the number of rules.
- Have unity weight for each rule.

2.4.2 Evolving Fuzzy Neural Network (EFuNN)

The Evolving Fuzzy Neural Network (EFuNN) proposed by Kasabov (N. Kasabov, 1998, 2007; N. Kasabov, 2008) implements a strategy of dynamically growing and pruning the connectionist (i.e. NN) architecture and parameter values. EFuNN is implemented in the NeuCom package developed at Auckland University of Technology[1].

A typical EFuNN structure consists of five layers (Figure 2-7).

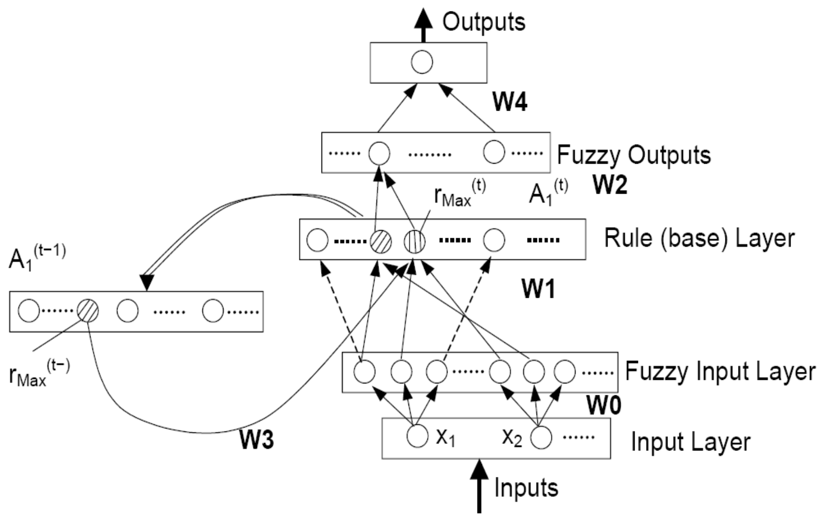


Figure 2-7: Architecture of Evolving Fuzzy Neural Network (N. Kasabov, 2007)

[1] The NeuCom project homepage: http://www.aut.ac.nz/research/research-institutes/kedri/research-centres/centre-for-data-mining-and-decision-support-systems/neucom-project-home-page

The input layer represents input variables. The second layer of nodes (fuzzy input neurons or fuzzy inputs) represents fuzzy quantization of each input variable space. For example, two fuzzy input neurons can be used to represent “small” and “large” fuzzy values for a particular input variable. Different membership functions can be attached to these neurons. The number and the type of MF can be dynamically modified. The task of the fuzzy input nodes is to transfer the input values into membership degrees to which they belong to the membership function (del-Hoyo, Martin-del-Brio, Medrano, & Fernandez-Navajas, 2009).

The third layer contains rule (case) nodes that evolve through supervised and/or unsupervised learning. The rule nodes represent prototypes (exemplars, clusters) of input–output (I/O) data associations that can be graphically represented as associations of hyperspheres from the fuzzy input and the fuzzy output spaces. Each rule node is defined by two vectors of connection weights— and the latter (W2 as in Figure 2-7) being adjusted through supervised learning based on the output error, and the former (W1 as in Figure 2-7) being adjusted through unsupervised learning based on similarity measure within a local area of the problem space. A linear activation function, or a Gaussian function, is used for the neurons of this layer (N. Kasabov, 1998; N. Kasabov, 2001).

The fourth layer of neurons represents fuzzy quantization of the output variables, similar to the input fuzzy neuron representation. Here, a weighted sum input function and a saturated linear activation function is used for the neurons to calculate the membership degrees to which the output vector associated with the presented input vector belongs to each of the output membership functions.

The fifth layer represents the values of the output variables. Here a linear activation function is used to calculate the defuzzified values for the output variables (N. Kasabov, 2001).

Each rule node, e.g., r_j , represents an association between a hypersphere from the fuzzy input space and a hypersphere from the fuzzy output space, the $w1(r_j)$ connection weights representing the coordinates of the center of the sphere in the fuzzy input space, and the $w2(r_j)$ connection weights representing the coordinates in the fuzzy output space.

Through the process of associating (learning) of new data points to a rule node r_j, the centers of this node hyperspheres adjust in the fuzzy input space depending on the distance between the new input vector and the rule node through a learning ratel_j and in the fuzzy output space depending on the output error through the Widrow–Hoff least mean square (LMS) algorithm (delta algorithm). This adjustment can be represented mathematically by the change in the connection weights of the rule node r_j from $w1(r_j^t)$ and $w2(r_j^t)$ to $w1\left(r_j^{t+1}\right)$and $w2(r_j^{t+1})$, respectively, according to the following vector operations (N. Kasabov, 2001):

$$W1\left(r_j^{(t+1)}\right) = W1\left(r_j^{(t)}\right) + l_{j,1} * (W1\left(r_j^{(t)}\right) - x_f) \qquad (25)$$

$$W2\left(r_j^{(t+1)}\right) = W2\left(r_j^{(t)}\right) + l_{j,2} * \left(A2 - y_f\right) * A1\left(r_j^{(t)}\right) \tag{26}$$

where $A2 = f_2(W2 * A1)$ is the activation vector of the fuzzy output neurons in the EFuNN structure when *x* is presented; $A1\left(r_j^{(t)}\right) = f_2(D(W1\left(r_j^{(t)}\right), x_f))$is the activation of the rule node; $l_{j,1}$ and $l_{j,2}$ are the current learning rates of rule node r_j for its input layer and its output layer of connections respectively.

Equations (25) and (26) have defined a standard EFuNN structure that has the first part updated in an unsupervised mode based on a similarity measure and the second in a supervised mode based on output error.

2.4.3 Fuzzy ARTMAP

Fuzzy ARTMAP is a self-organizing architecture that is capable of rapidly learning to recognize, test hypotheses and predict the consequences of virtually any input. It involves a combination of neural and fuzzy operations that together give these useful capabilities. Like other versions of Adaptive Resonance Theory (ART), its use is almost exclusively for classification, and it has only one user-selectable parameter which determines the fitness or coarseness of the patterns into which the inputs are fitted. It can learn virtually every training pattern in a few training iterations in an unsupervised mode.

Fuzzy ARTMAP adapts a competitive learning model based on ART. It is an extension of ART1 (for binary inputs) and ART2 (for continuous inputs) for fuzzy inputs (G. A. Carpenter, Grossberg, Markuzon, Reynolds, & Rosen, 1992; Gail A. Carpenter, Grossberg, & Rosen, 1991; Georgiopoulos, Huang, & Heileman, 1994). Fuzzy ARTMAP consists of two ART modules, i.e. ARTa and ARTb, as in Figure 2-8. Both ARTa and ARTb are fuzzy ARTs (i.e. accepting fuzzy inputs), each of which is comprised of three layers: normalization layer, input layer and recognition layer. The main purpose of the map field F_{ab} is to classify a fuzzy pattern into the given class, or to re-start the matching procedure (Liu & Li, 2004).

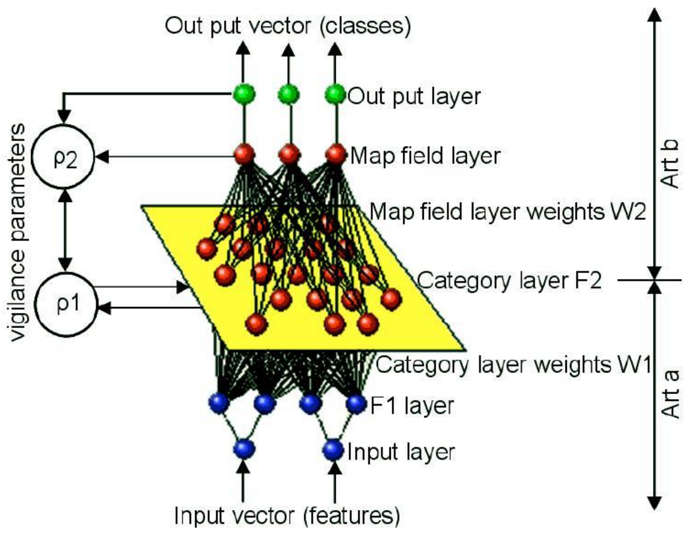


Figure 2-8: Fuzzy ARTMAP architecture (Mannan, Roy, & Ray, 1998)

Fuzzy ARTMAP has proven itself as a supervised incremental learning system in pattern recognition and *M-to-N* dimensional mapping (Downs, Harrison, Kennedy, & Cross, 1996). The two fuzzy ARTs are connected using a series of weight connections between the F_2 layers both in ART_a andART_b. Those connections are weighted with a value between 0 and 1. Learning in Fuzzy ARTMAP encompasses the recruitment of new prototype vectors and expansion of the boundary of existing prototype vectors in the feature space. Like other incremental NNs, the Fuzzy ARTMAP growth criterion is subject to a similarity measure between the input pattern and the prototypes stored in the network (S.

C. Tan, Rao, & Lim, 2008). The Matlab package implementation of Fuzzy ARTMAP is available from the lab led by Carpenter[2].

2.4.4 GNMM

GNMM was proposed by Jianhua Yang in 2009 (Yang, 2009). Utilizing GAs and MLPs, this technique is capable of pattern classification and analysis. It not only inherits the advantages of ANN, such as robustness and nonlinearity but also incorporates a GA and mathematical programming to optimize input feature selection and rule extraction. By incorporating GA, GNMM can optimize the number of input features to the MLP automatically. By employing a mathematical programming module, GNMM is also able to identify regression rules extracted from the trained MLPs.

It consists of three steps: (1) GA-based input feature selection, (2) Multi-Layer Perceptron (MLP) modeling and (3) rule extraction based on mathematical programming (Yang, 2009). In the first step, GAs are introduced in order to get an optimal set of MLP inputs. The mutation rate and hence the exploration/exploitation balance are adjusted using an adaptive method which is based on the average fitness of successive generations. The elite group and appearance percentage are also introduced to minimize the randomness associated with GAs. In the second step, MLP modeling is the core part of this data mining technique for performing classification/prediction tasks. Before the training process, an Independent Component Analysis (ICA) based weight initialization algorithm is used to determine optimal weights. The LM algorithm is used to achieve a second-order speedup compared to conventional BP training (Chow, 2007;

[2] CNS Tech Lab, Boston University, http://techlab.bu.edu/resources/software

Ham & Kostanic, 2000). In the third step, mathematical programming can be used to identify the parameters of extracted multivariate polynomial rules. In addition to that, mathematical programming can also explore features from the extracted rules based on data samples associated with each rule. Therefore, the methodology can provide regression rules and features not only in the separate multi-dimensional spaces with data instances, but also in the spaces without data instances.

2.5 Challenges of Knowledge Presentation

Although neural networks and fuzzy logic can be powerful in dealing with practical problems, they don't provide a friendly interface for humans to understand. Neural networks are often referred as 'black-boxes' because it is quite difficult to understand the results or otherwise to extract transferrable knowledge from the results and the neural network structure. When the neural networks are successfully trained, no information in symbolic form is available, which present obstacles for humans to verify or interpret.

This issue has to be addressed by using neural networks to extract knowledge to replace human specialists for knowledge acquisition. In general, rules can be extracted at the level of hidden and output units or can be extracted by mapping inputs directly into output which evaluate the input-output relationship as closely as possible. But this process could be tedious especially when rules are extracted at the level of hidden and output units or infeasible when the rules are extracted by mapping input into output.

In order to overcome these difficulties, hybrid systems can be introduced to both train the neural networks and represent knowledge which is interpretable to humans. For example, when neural networks are combined with fuzzy logic, the extracted rules or knowledge will present a different perspective. The connections between neurons can be mapped as knowledge rules and labeled with confidence, possibility measures or certainty factors. The architecture and computational algorithm of the created network will implement the set of rules and their fuzzy combinations. The network which is initiated from fuzzy rules can be trained using original dataset to revise or update the extracted knowledge or rules. During the learning process, only the parameters of the rules (e.g. certainty factor or importance weights of the rules) are updated while the rules themselves are preserved. These hybrid systems are a powerful set of tools for solving problems in pattern recognition, data processing and prediction. They offer an effective approach for handling large amounts of dynamic, non-linear and noisy data, especially when the underlying physical relationships are not fully understood.

2.6 Datasets

In order to design, implement and test different intelligent system based approaches, including the proposed OWFDT, several datasets will be introduced in the following parts to compare the classification results for different techniques.

Iris Data: A dataset with 150 random samples of flowers from three iris species named setosa, versicolor, and virginica collected by Anderson (1935). For each species there are

50 observations with four attributes named sepal length, sepal width, petal length, and petal width in cm (Black & Merz). This dataset was used by Tsang (Tsang, Wang, & Yeung, 2000) to verify the learning accuracy of the weighted fuzzy decision tree over BP algorithm. It will be introduced in chapter three to compare Optimized Weighted Fuzzy Decision Tree (OWFDT) with weighted fuzzy decision tree both in terms of accuracy and number of training epochs to show the merits of OWFDT.

Wisconsin Breast Cancer: In this dataset, each pattern has 30 features and an associated class label (benign or malignant). The total number of instances are 569, of which 357 instances are classified as 'benign' and 212 instances are classified as 'malignant' (Black & Merz). Although this dataset was widely used as a benchmark dataset, no research had been conducted by using fuzzy decision tree approach. In addition to that, the fuzzy production rules produced by OWFDT can be linked with the attributes to enhance the interpretability of relationship between the attributes and the actual class labels.

Brain Computer Interface: This dataset is from dataset I of the BCI Competition 2005. In this motor imagery experiment of the left small finger or the tongue, ECoG was recorded by using 8×8 ECoG platinum electrode grids, placed on the contralateral motor cortex. Data was sampled at 1000 Hz with trial length of three seconds. After extraction of features using Common Spatial Patterns (CSP) and their derivatives, the dimension of the data was managed to be reduced to 12 features with two different classes (Singh, Li, Hines, & Stocks, 2007). In order to make classifications to the possible ECoG signals, this dataset will be used to construct a fuzzy decision tree and produce some fuzzy production rules. The dataset consists of 278 training instances and 100 testing instances respectively.

Eye Bacteria Data: This dataset was collected through a joint collaboration between researchers from the University of Warwick and Doctors from Heartlands Hospital and Micropathology Ltd. All the eye bacteria strains were grown under a prescribed medical condition. An electronic nose, Cyranose 320, based on a 32-sensor array of conducting polymers which produce a unique response signature representative of the test smell, was used to detect different odor features of six different eye bacteria species. 180 sample readings were recorded to form this dataset. Each reading contains 32 features (Boilot, et al., 2002; Dutta, Hines, Gardner, & Boilot, 2002; Gardner, Boilot, & Hines, 2005). Boilot (Boilot, et al., 2002; Gardner, et al., 2005) also applied V-integer GA using PNN to reduce the dimension of the dataset and obtained an optimal subset. But classification accuracy using the selected features is still subject to improvement. In chapter 5, bootstrap aggregating to decision tree will be used to select the most important features and the learning results using OWFDT will be compared with results obtained by Boilot (Boilot, et al., 2002; Gardner, et al., 2005) to show the improvement of OWFDT.

Composite Material Data: The data was collected by Warwick Manufacturing Group (WMG) to detect the failure mechanisms in composite material. When the composite material is being used, some failures may occur, such as fiber breakage, matrix cracking or fiber/matrix debonding, accompanied by characteristic audible noises. These are captured as sound signals and analyzed as follows. Four frequencies for each signal were collected after filtering using a Fast Fourier Transform (FFT). 7 samples which are different in terms of the fiber/matrix orientation (0°, 15°, 30°, 45°, 60°, 75°, 90°) were used for the experiments which in turn leads to 7 subsets. Each subset may contain three or four classes of failure mechanisms dominating in the corresponding samples and the number of instances ranges from 534 to 7052 (X. Li, et al., 2008a, 2008b). Although there are some researches which focus on utilizing frequencies as descriptors to differentiate different failure mechanisms, no study using hybrid intelligent computing techniques has been conducted. Chapter 4 will provide a detailed analysis of these datasets and the construction of a classifier using OWFDT. The introduction of fuzzy rules also produces a new perspective to classify different failure mechanisms which are also quite fuzzy in nature.

2.7 Summary

The current chapter has briefly reviewed some intelligent DM techniques including NNs, FL, DT and some hybrid techniques including ANFIS, EFuNN, Fuzzy ARTMAP, GNMM etc. The technique proposed in the next chapter had successfully combined FL, NN and DT together and a benchmarking study in chapter VI has also proved its efficiency over some hybrid techniques introduced in this chapter such as ANFIS, EFuNN in terms of its accuracy and the number of rules.

References

Babuska, R. (1998). Fuzzy Modelling for Control. *Kluwer Academic Publishers, The Netherlands*.

Bezdek, J. C., Ehrlich, R., & Full, W. (1984). FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences, 10*(2-3), 191-203.

Black, C. L., & Merz, C. J. UCI repository of machine learning databases,University of California, Department of Information and Computer Science. http://www.ics.uci.edu/mlearn/MLRepository.html

Boilot, P., Hines, E. L., Gardner, J. W., Pitt, R., John, S., Mitchell, J., et al. (2002). Classification of bacteria responsible for ENT and eye infections using the Cyranose system. *Sensors Journal, IEEE, 2*(3), 247-253.

Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H., & Rosen, D. B. (1992). Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *Neural Networks, IEEE Transactions on, 3*(5), 698-713.

Carpenter, G. A., Grossberg, S., & Rosen, D. B. (1991). Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks, 4*(6), 759-771.

Chow, T. W. S. (2007). *Neural Networks and Computing: Learning Algorithms and Applications*: World Scientific Publishing Co., Inc.

del-Hoyo, R., Martin-del-Brio, B., Medrano, N., & Fernandez-Navajas, J. (2009). Computational intelligence tools for next generation quality of service management. *Neurocomput., 72*(16-18), 3631-3639.

Downs, J., Harrison, R. F., Kennedy, R. L., & Cross, S. S. (1996). Application of the fuzzy ARTMAP neural network model to medical pattern classification tasks. *Artificial Intelligence in Medicine, 8*(4), 403-428.

Du, K.-L., & Swamy, M. N. S. (2006). *Neural Networks in a Softcomputing Framework*: Springer-Verlag New York, Inc.

Dutta, R., Hines, E., Gardner, J., & Boilot, P. (2002). Bacteria classification using Cyranose 320 electronic nose. *BioMedical Engineering OnLine, 1*(1), 4.

Gardner, J. W., Boilot, P., & Hines, E. L. (2005). Enhancing electronic nose performance by sensor selection using a new integer-based genetic algorithm approach. *Sensors and Actuators B: Chemical, 106*(1), 114-121.

Georgiopoulos, M., Huang, J., & Heileman, G. L. (1994). Properties of learning in ARTMAP. *Neural Networks, 7*(3), 495-506.

Godin, N., Huguet, S., & Gaertner, R. (2005). Integration of the Kohonen's self-organizing map and k-means algorithm for the segmentation of the AE data collected during tensile tests on cross-ply composites. *NDT&E International, 38*, 299-309.

Hagan, M. T., Demuth, H. B., & Beale, M. (1996). *Neural network design*: PWS Publishing Co.

Ham, F. M., & Kostanic, I. (2000). *Principles of Neurocomputing for Science and Engineering*: McGraw-Hill Higher Education.

Haykin, S. (1999). *Neural networks: a comprehensive foundation*. New York: Macmillan.

Höppner, F., Klawonn, F., Kruse, R., & Runkler, T. (1999). *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition*. Sussex: John Wiley & Sons Ltd.

Ian, C., & Jacek, M. Z. (2000). *Knowledge-based neurocomputing*: MIT Press.

Jang, J.-S. R., & Sun, C.-T. (1997). *Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence*: Prentice-Hall, Inc.

Jang, J. S. R. (1993). ANFIS: adaptive-network-based fuzzy inference system. *Systems, Man and Cybernetics, IEEE Transactions on, 23*(3), 665-685.

Karray, F. O., & Silva, C. W. D. (2005). *Soft Computing and Intelligent Systems Design: Theory, Tools and Applications*: Addison-Wesley.

Kartalopoulos, S. V. (1997). *Understanding Neural Networks and Fuzzy Logic: Basic Concepts and Applications*: Wiley-IEEE Press.

Kasabov, N. (1998). *Evolving Fuzzy Neural Networks - Algorithms, Applications and Biological Motivation.* Paper presented at the Methodologies for conception, design and application of soft computing.

Kasabov, N. (2001). Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 31*(6), 902-918.

Kasabov, N. (2007). *Evolving Connectionist Systems:The Knowledge Engineering Approach*. London: Springer.

Kasabov, N. (2008). Evolving Intelligence in Humans and Machines: Integrative Evolving Connectionist Systems Approach. *Computational Intelligence Magazine, IEEE, 3*(3), 23-37.

Kasabov, N. K. (2002). *Evolving Connectionist Systems: Methods and Applications in Bioinformatics, Brain Study and Intelligent Machines*: Springer-Verlag New York, Inc.

Kohonen T. (1990). Self-organized network. *Proceeding of IEEE*, pp 59-69.

Lampinen, T., Laurikkala, M., Koivisto, H., & Honkanen, T. (2005). Profiling Network Applications with Fuzzy C-means and Self-Organizing Maps *Classification and Clustering for Knowledge Discovery* (Vol. 4, pp. 15-27).

Li, H., Philip, C. C. L., & Huang, H.-P. (2000). *Fuzzy Neural Intelligent Systems: Mathematical Foundation and the Applications in Engineering*: CRC Press, Inc.

Li, X., Ramirez, C., Hines, E. L., Leeson, M. S., Purnell, P., & Pharaoh, M. (2008a). Discriminating Fiber-Reinforced Plastic Signatures Related to Specific Failure Mechanisms: Using Self-Organizing Maps and Fuzzy C-Means. In E. L. Hines, M. S. Leeson, M. M. Ramón, M. Pardo, E. Llobet, D. D. Iliescu & J. Yang (Eds.), *Intelligent Systems: Techniques and Applications*: Publisher: Shaker publishing.

Li, X., Ramirez, C., Hines, E. L., Leeson, M. S., Purnell, P., & Pharaoh, M. (2008b). *Pattern recognition of fiber-reinforced plastic failure mechanism using computational intelligence techniques.* Paper presented at the Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on.

Lin, C.-T., & Lee, C. S. G. (1996). *Neural fuzzy systems: a neuro-fuzzy synergism to intelligent systems*: Prentice-Hall, Inc.

Liu, P., & Li, H. (2004). *Fuzzy Neural Network Theory and Application*: World Scientific Press.

Mannan, B., Roy, J., & Ray, A. K. (1998). Fuzzy ARTMAP supervised classification of multi-spectral remotely-sensed images. *International Journal of Remote Sensing, 19*(4), 767 - 774.

Martinez, W. L., & Martinez, A. R. (2007). *Computational Statistics Handbook with MATLAB, Second Edition (Chapman Hall/CRC/Computer Science & Data Analysis)*: Chapman Hall/CRC.

Nauck, D., Klawonn, F., & Kruse, R. (1997). *Foundations of Neuro-Fuzzy Systems*: John Wiley & Sons, Inc.

Singh, H., Li, X., Hines, E. L., & Stocks, N. (2007). Classification and feature extraction strategies for multi channel multi trial BCI data. *International Journal of Bioelectromagnetism, 9*(4), 233–236.

Soyguder, S., & Alli, H. (2009). An expert system for the humidity and temperature control in HVAC systems using ANFIS and optimization with Fuzzy Modeling Approach. *Energy and Buildings, 41*(8), 814-822.

Tan, P.-N., Steinbach, M., & Kumar, V. (2005). *Introduction to Data Mining, (First Edition)*: Addison-Wesley Longman Publishing Co., Inc.

Tan, S. C., Rao, M. V. C., & Lim, C. P. (2008). Fuzzy ARTMAP dynamic decay adjustment: An improved fuzzy ARTMAP model with a conflict resolving facility. [doi: DOI: 10.1016/j.asoc.2007.03.006]. *Applied Soft Computing, 8*(1), 543-554.

Theodoridis, S., & Koutroumbas, K. (1999). *Pattern Recognition*. London: Academic Press.

Tsang, E. C. C., Wang, X. Z., & Yeung, D. S. (2000). Improving learning accuracy of fuzzy decision trees by hybrid neural networks. *Fuzzy Systems, IEEE Transactions on, 8*(5), 601-614.

Tsoukalas, L. H., & Uhrig, R. E. (1996). *Fuzzy and Neural Approaches in Engineering*: John Wiley & Sons, Inc.

Umanol, M., Okamoto, H., Hatono, I., Tamura, H., Kawachi, F., Umedzu, S., et al. (1994). *Fuzzy decision trees by fuzzy ID3 algorithm and its application to diagnosis systems.* Paper presented at the Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the Third IEEE Conference on.

Yang, J. (2009). *Intelligent Data Mining using Artificial Neural Networks and Genetic Algorithms: Techniques and Applications.* University of Warwick.

Chapter III: Improving Classification Rate Using Optimized Weighted Fuzzy Decision Trees

Chapter two had introduced some theoretical background to NN, FL and DT. This chapter will introduce the Optimized Weighted Fuzzy Decision Tree which is a classifier based on FL, ANN and DT. This chapter proposes an optimized WFDT, incorporating the introduction of Fuzzy C-Means to fuzzify the input instances but keep the expected labels crisp. This leads to a different output layer activation function and weight connection in the NN structure obtained by mapping the WFDT to NN. The chapter also introduces a momentum term in the learning process to train the weight connections to avoid oscillation or divergence. A new reasoning mechanism has also been proposed to combine the constructed tree with those weights which had been optimized in the learning process. These improvements of the Optimized Weighted Fuzzy Decision Tree (OWFDT) will be verified using two benchmark datasets and one dataset collected from our projects in this chapter. Three other datasets will also be applied to OWFDT in the following three chapters to demonstrate those improvements.

3.1 Introduction

Decision tree learning is one of the most widely used and accepted techniques for inductive inference. After partitioning the feature space, a tree is formed of nodes and branches. If a node does not have any children, it is classified as a leaf node. Every leaf node is associated with a label or value that characterizes the given data (Breiman, Friedman, Stone, & Olshen, 1984).

Many learning algorithms have been proposed for decision trees in the past. The most popular one was developed by Quinlan and is known as Interactive Dichotomizer 3 (ID3) (Quinlan, 1986). During the past several decades, many efforts have been made to combine fuzzy set theory with decision tree technique (Janikow, 1998; Levashenko, Zaitseva, & Puuronen, 2007; Yuan & Shaw, 1995; Zhiheng, Gedeon, & Nikravesh, 2008). Fuzzy set provides a theoretical background for fuzzy representation and fuzzy modeling of language related uncertainties. The combination of fuzzy sets with decision tree enables the latter to handle the uncertainty and enhances its approximate reasoning capability without sacrificing comprehensibility and ease of application (R.Weber, 1992; X. Wang, Chen, Qian, & Ye, 2000).

3.2 Fuzzy Decision Tree Algorithm

The fuzzy decision tree differs from the traditional crisp decision tree in two aspects: it uses splitting criteria based on fuzzy restrictions and uses a different inference procedure.

3.2.1 ID3 Algorithm Introduction

The ID3 algorithm is based on information entropy theory. It examines all the features and chooses the one which has the greatest information gain or the greatest entropy decrease. Entropy is defined as $\sum_i(-p_i\, log_2 p_i)$ where p_i is the possibility of occurrence for class i in the whole dataset.

Suppose we have L labeled patterns which fall into C_K clusters belonging to C_K classes. The ID3 algorithm for constructing a decision tree is outlined below:

- Compute the information content at the current node;
- Select the feature which results in the maximum decrease of entropy as the feature to be split at this node;
- Build the leaves of the current node using the selected feature;
- Repeat step 1 to 3 until all the leaves contain a pure class thus the tree’s entropy is zero.

In Fuzzy ID3 algorithm, input attributes are automatically discretized in linguistic terms, based on the distribution of pattern points in the feature space. Input feature values are described in terms of some combination of overlapping membership values in the linguistic property sets such as the most widely used low (L), medium (M) and high (H). An n-dimensional pattern $I = [F_1, F_2, \dots F_n]$ will be represented by a $3n$-dimensional pattern after being fuzzified:

$$\begin{aligned} I &= [\mu_{low}(F_1), \mu_{med}(F_1), \mu_{high}(F_1), \mu_{low}(F_2), \mu_{med}(F_2), \mu_{high}(F_2), \dots, \\ &\qquad \mu_{low}(F_n), \mu_{med}(F_n), \mu_{high}(F_n)] \qquad (1) \\ &= \left[F_{1,l}, F_{1,m}, F_{1,h}, F_{2,l}, F_{2,m}, F_{2,h}, \dots, F_{n,l}, F_{n,m}, F_{n,h}\right] \end{aligned}$$

where μ is the membership functions of the corresponding linguistic terms low, medium and high along every feature.

The fuzzy ID3 algorithm is very similar to ID3, except that rather than selecting the attribute based on the information gain (computed based on the probability of unfuzzified data) it computes the information gain based on the probability of the membership values in the dataset. The Fuzzy ID3 algorithm is extended so as to be applicable to a fuzzy set of data (a dataset with membership grades) and generate a FDT using fuzzy sets defined for all attributes. A fuzzy decision tree consists of nodes for testing attributes, edges for branching by membership values of fuzzy sets and leaves for deciding class names with certainties.

3.2.2 Construction of a Fuzzy Decision Tree Based on Fuzzy ID3

Suppose we have a data set D with m instances and each instance has n features which can be denoted as $I = [F_1, F_2, \dots F_n]$ and I has been classified as a certain class in the class set $C = [1, 2, \dots C_K]$. The input features can be fuzzified as $[F_{1,l}, F_{1,m}, F_{1,h}, F_{2,l}, F_{2,m}, F_{2,h}, \dots, F_{n,l}, F_{n,m}, F_{n,h}]$. Let D^{C_k} be a fuzzy subset in D which just includes a pure class C_k and let $|D|$ be the sum of the membership values in a fuzzy set of data D. The fuzzy ID3 algorithm to generate a fuzzy decision tree can be summarized as follows (Umanol, et al., 1994):

- Generate the root node that has a set of all data, i.e., a fuzzy set of all data with the membership value 1.
- If a node t with a fuzzy set of data D satisfies the following conditions:

  (a) the membership value proportion of a sub dataset which has a pure class C_k is greater than or equal to a threshold θ_r , that is,

$$\frac{|D^{C_k}|}{|D|} \geq \theta_r \tag{2}$$

  (b) the number of instances in a dataset is less than a threshold θ_n , that is,

$$\{D\}_{length} < \theta_n \tag{3}$$

  (c) there are no attributes for further classification,

Then it is a leaf node and will be assigned to a class name (more detailed method is described in step 3).

- If the node t does not satisfy the conditions in previous step, it is not a leaf node and the child nodes are generated as follows:
    a) For any feature $F_i(i = 1,2, \dots , n)$ in the dataset, calculate the information gain $G(F_i, D)$, as formulated in equations 4 to 8, and select the test attributeF_{max} , that maximizes them.
    b) Divide dataset D into three fuzzy subsets D_l, D_m, D_h according to F_{max} , where the Membership value of the data in $D_j, (j = l, m, h)$ is updated to the product of the membership value in D and the prospective membership value of $F_{max\,,j}\,(j = l, m, h)$ of the feature F_{max} in *D*.
    c) Generate new nodes t_1, t_2, t_3, for fuzzy subsets D_l, D_m, D_h, and label the fuzzy sets $F_{max\,,j}\,(j = l, m, h)$ to edges that connect between the nodes t_i and root *t*.
    d) Replace D by $D_i(i = l, m, h)$ and repeat from 2 recursively.

The information gain $G(F_i, D)$ for the attribute F_i in the dataset D is defined by:

$$G(F_i, D) = I(D) - E(F_i, D) \tag{4}$$

where

$$I(D) = -\sum_{k=1}^{C_k} (p_k * log_2 p_k) \tag{5}$$

$$E(F_i, D) = \sum_j (p_{i,j} * I\left(D_{F_{i,j}}\right))(j = l, m, h) \tag{6}$$

$$p_k = \frac{|D^{C_k}|}{|D|} \tag{7}$$

$$p_{i,j} = \frac{\left|D_{F_{i,j}}\right|}{\sum_j \left|D_{F_{i,j}}\right|} (j = l, m, h) \tag{8}$$

As for assigning the class name to the leaf node, Fuzzy ID3 proposes two typical methods as follows:

- The node is assigned to the class name whose membership value holds the largest proportion of the whole membership vales in the dataset associated with this leaf node;
- The node is assigned to all class names with prospective possibilities which are calculated as the proportion of the prospective class membership value to the whole membership vales in the dataset associated with this leaf node.

If the Wisconsin Breast Cancer dataset (Black & Merz) was applied to the procedures listed as above, the feature which has the biggest information gain in the tree root will be the sixth feature (F_6) in this dataset. According to (F_6), the original dataset D can be divided into three fuzzy subsets $D_{6,l}, D_{6,m}, D_{6,h}$. Three child nodes were then created for the root node, and three subsets $D_{6,l}, D_{6,m}, D_{6,h}$ were assigned to three child nodes. This process was repeated until equation 2 or 3 was satisfied. Figure 3-1 shows a part of a fuzzy decision tree constructed using the Wisconsin Breast Cancer dataset. This dataset consists of 569 samples. Each of these samples is denoted by 30 attributes and is classified as either 'benign' or 'malignant'. The FDT was constructed using half of the samples as a training dataset.

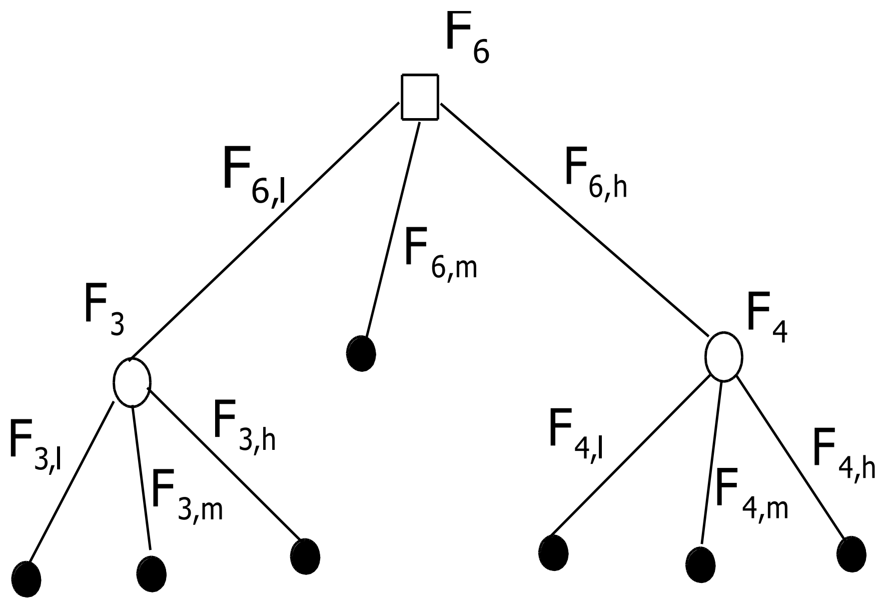


Figure 3-1: the architecture of a Fuzzy Decision Tree

3.2.3 Fuzzy Decision Tree Inference Mechanism

In addition to the heuristic algorithm for generating the FDT, another aspect associated with FDT induction is the reasoning mechanism. After generating the FDT, we need a mechanism to predict the classification of novel examples or to test the classification of training examples.

A FDT is composed of a set of internal nodes which denote the attributes used to construct the FDT and a set of leaf nodes which represents the degree of certainty of the node belonging to each class. The connection from root to leaf is called a branch or a path. The connection between two adjacent nodes (parent node and child node) in one path is called a segment of the path. All segments in one path are considered to have equal importance to the classes labeled at the leaf.

Suppose a FDT grows m leaf nodes, which leads to m paths. Each $path_i, (i = 1,2,\dots m)$ consists of several segments denoted by $segm_1^i, segm_2^i, \dots segm_{n_i}^i$. The certainty degree of that leaf node belonging to each class is denoted as a vector: $[\alpha_{i,1}, \alpha_{i,2}, \dots, \alpha_{i,C_K}]$. For example, leaf node 1 in Figure 3-1 is a path which consists of two segments $segm_1^1$ $(F_{6,l})$ and $segm_2^1$ $(F_{3,l})$. The numerical value associated with every segment is a membership value of that attribute on that linguistic term (Tsang, Wang, & Yeung, 2000). An instance is classified by starting from the root node to reach every leaf along the corresponding path. Let e be the new example to be tested on the tree. For every $path_i, (i = 1,2,\dots m)$ the membership value on every segment can be denoted as $F_1^i, F_2^i, \dots F_{n_i}^i$. The mechanism for classifying the class of the example e is summarized as follows:

- For each leaf node (path) $i (1 \le i \le m)$, compute the possibilities of the input sample e falling into each class using:

$$p_i^e = \left(\prod_{n_i} F_{n_i}^i \right) * [\alpha_{i,C_1}, \alpha_{i,C_2}, \dots, \alpha_{i,C_K}] \tag{9}$$

- Aggregate the possibilities of each leaf node (path) for the sample e:

$$C^e = \sum_{i=1}^{m} p_i^e$$

- The inferred result C^e is regarded as a fuzzy vector $[c_1^e, c_2^e, \dots c_k^e]$ where c_k^e is the value which indicates to what degree the example e belongs to class $k, (k = 1,2, \dots C_K)$. When a crisp inferred result is needed, the maximum value c_{max}^e $(1 \leq max \leq C_K)$ can be regarded as the final result, which infers the instance to be class $\max(1 \leq max \leq C_K)$.

It should be noted that the inference mechanism can be converted into a set of fuzzy production rules (FPR), which will be discussed in the following section, section 3.3.

3.3 Optimized Weighted Fuzzy Decision Tree

The weights in the fuzzy decision tree refer to those parameters in each path. The most commonly used parameters are GW and LW, which are explained as follows (Tsang, et al., 2000; X. Z. Wang, Yeung, & Tsang, 2001):

- The degree of certainty that the leaf node belongs to a certain class which is called the Global Weight (GW).

In the FDT, every leaf node is assigned a possibility of belonging to all the existing classes by aggregating the possibilities of each leaf node (path) for every sample. The inference mechanism of the FDT will aggregate all the leaf nodes together to determine the possibility of its instance's belongingness to the classes. In a weighted FDT, since there is generally more than one leaf node having the same classification, after assigning a GW to the leaf node (path), we introduce a different degree of importance to each node contributing to the final classification results. In (X. Wang, et al., 2000), only a certain number of leaf nodes have global weight connections due to fuzzification of the output. However in our proposed technique, every leaf node has a weight connection to the final output, as demonstrated in Figure 3-2, since we did not fuzzify the expected output.

- The degree of importance of each segment in a path which leads to the classification of that leaf node which is called Local Weight (LW).

In the FDT inference mechanism, the weights of those segments in one path from the root to a leaf are considered to be equal. While in a weighted FDT, an LW is initiated and assigned to every segment in one path to demonstrate the relative degree of importance of that segment contributing to the leaf's classification.

After a leaf node (path) is converted to fuzzy production rule (FPR), which is similar to the inference mechanism in ID3, a segment in the path corresponds to a proposition in a FPR. The introduction of the concept of LW indicates that diverse propositions in one FPR should have a different degree of importance contributing to its consequent (classification). Actually, these weights also indicate the importance of all the attributes and their corresponding linguistic terms.

We modify the weighted fuzzy production rules proposed by (X. Z. Wang, et al., 2001; Yeung, Wang, & Tsang, 1999) in four ways, mainly input fuzzification, modified weighted FPRs, the approach mapping weighted fuzzy decision tree to neural networks and an improved learning process for the constructed neural networks:

3.3.1 Fuzzify Input

When the attributes are categorical, the fuzzyfication is quite straightforward. Each possible value can be regarded as a fuzzy subset in which the membership value is either zero or one. For numerical attributes, we introduced the Fuzzy C-Means (FCM) clustering algorithm to cluster the attribute values into three clusters representing three linguistic terms. The choice of the number of clusters was set to be three since it was guided by the notion that a value can typically be thought of as being low, medium, or high. We generated the memberships based on a triangular membership function. Three cluster centers and the maximum and minimum values along each feature will be used as parameters in the function. Let $F_{j,max}$ and $F_{j,min}$ be the maximum and minimum values and $F_{j,high}, F_{j,med}$ and $F_{j,low}$ be the three sorted cluster centers along feature F_j considering N training patterns $F_j^1, F_j^2, \dots, F_j^n$. The membership values of an input pattern along the jth feature, corresponding to the three-dimensional linguistic term, are formulated as Equations 10, 11 and 12 (Mitra, Konwar, & Pal, 2002).

$$\mu_{low}(F_j) = \begin{cases} 0 & if\ F_j \geq F_{j,med} \\ \dfrac{F_{j,med} - F_j}{F_{j,med} - F_{j,low}} & if\ F_{j,low} \leq F_j < F_{j,med} \\ 1 & if\ F_{j,min} \leq F_j < F_{j,low} \end{cases} \tag{10}$$

$$\mu_{med}(F_j) = \begin{cases} \dfrac{F_j - F_{j,min}}{F_{j,low} - F_{j,min}} & if\ F_{j,min} \leq F_j < F_{j,low} \\ 1 & if\ F_{j,low} \leq F_j < F_{j,med} \\ \dfrac{F_{j,high} - F_j}{F_{j,high} - F_{j,med}} & if\ F_{j,med} \leq F_j < F_{j,high} \\ 0 & if\ F_j \geq F_{j,high} \end{cases} \tag{11}$$

$$\mu_{high}(F_j) = \begin{cases} 0 & if\ F_j < F_{j,med} \\ \dfrac{F_{j,high} - F_j}{F_{j,high} - F_{j,med}} & if\ F_{j,med} \le F_j < F_{j,high} \\ 1 & if\ F_{j,high} \le F_j \le F_{j,max} \end{cases} \tag{12}$$

In our dataset and simulations, each instance belongs to a single class, so unlike the usual weighted fuzzy decision trees, we did not fuzzify the class labels, which will result in different mapping structure from the weighted fuzzy decision tree and reasoning mechanism as discussed in the following sections.

3.3.2 Modified Weighted Fuzzy Production Rules

Usually, a rule consists of propositions in the antecedent and a conclusion in the consequent part. In (Tsang, et al., 2000; Yeung, et al., 1999) a generic form of fuzzy production rule was proposed by introducing LW and GW. LW indicates the importance of each proposition and GW indicates the importance of the entire rule. For instance, a generic type of FPR can be denoted as:

$$R^i\text{: If } V_1 \text{ is } A_1^{(i)} \text{ and } V_2 \text{ is } A_2^{(i)} \dots \text{and } V_{n_i} \text{ is } A_{n_i}^{(i)} \tag{13}$$

Then

$$U \text{ is } B^i, LW_1^{(i)}, LW_2^{(i)}, \dots, LW_{n_i}^{(i)}, GW^{(i)} \tag{14}$$

The propositions in the rule specify the attribute values as:

Proposition 1: V_1 is $C_1^{(i)}$; Proposition 2 : V_2 is $C_2^{(i)}$, …, Proposition n_i: V_{n_i} is $C_{n_i}^{(i)}$

where $V_1, V_2, \dots, V_{n_i}$ are attributes and $D_1^{(i)}, D_2^{(i)}, \dots, D_{n_i}^{(i)}$ are observed fuzzy values of these attributes. B^i is the output of the fuzzy rule, LW_i is the weight assigned to each attribute and $GW^{(i)}$ is the weight for this rule contributing to the final conclusion.

For each rule, the similarity between the proposition $A_j^{(i)}$ and the observed attribute-value $D_j^{(i)}$ is defined as the membership value which indicates to what degree the example belongs to the corresponding linguistic term. This similarity is denoted by$SM_j^{(i)}$. So the overall similarity of the rule can be defined as:

$$SM^{(i)} = Min_{(1 \le j \le n_i)}(LW_i * SM_j^{(i)}) \tag{15}$$

Suppose there are C_K classes denoted by $Class_1, Class_2, \dots, Class_{C_K}$. The inferred result is regarded as a fuzzy vector $[x_1, x_2, \dots, x_{C_K}]$ where $x_k(k = 1,2, \dots, C_K)$ is the value which indicates to what degree the observed object belongs to $Class_k(k = 1,2, \dots, C_K)$. The degree is determined by the following:

$$x^k = \sum_{\boldsymbol{B}^{(i)}=Class_k (k=1,2,\dots,C_K)} GW^{(i)} * SM^{(i)} \tag{16}$$

3.3.3 Mapping weighted fuzzy Decision Tree to Neural Networks

In order to gain better prediction results, the next step is to optimize both the local and global weights associated with the rules. Inspired by the NN principles, those rules can be mapped to the NN structure so that the weights can be updated through learning (Suarez & Lutsko, 1999; Tsang, et al., 2000). Figure 3-2 shows the structure of the converted NN and the relationship between the NN and the rules. This structure is based on a part of Wisconsin Breast Cancer fuzzy decision tree showed in Figure 3-1. The weights between the input and hidden layers were set to be the *LW*s and the weights between the hidden layer and the output layer were set to be *GW*s. The following is a specific description of the three layers in the network structure.

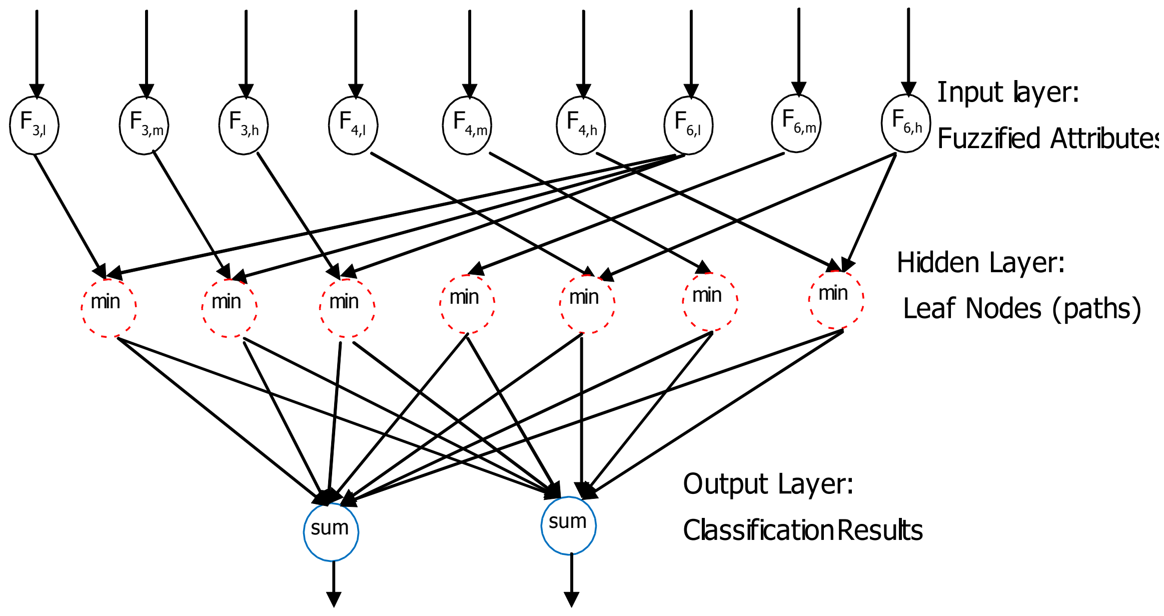


Figure 3-2: The Mapping Structure from FPR to Neural Network

Input Layer: Nodes in the input layer represent the propositions in the antecedent. The input of each node is regarded as the degree of similarity between the observed attribute value and the corresponding linguistic term (proposition) of the antecedent in a fuzzy rule. In our simulation, the input value was set to be the membership value for the prospective linguistic terms.

Hidden Layer: The nodes in this layer represent the leaf nodes in the fuzzy decision tree or the rules in the fuzzy production rules. Unlike the usual neural networks in which every node in the input layer has a weight associated with the hidden node, ONLY those nodes which appear in the antecedent part of the rule will have weights (LWs) assigned to connect to the associated nodes in the hidden layer. The activation function in this layer was set to:

$$H_j = Min(LW_{1,j} * x_1, LW_{2,j} * x_2, \dots, LW_{N_o,j} * x_{N_o}) \qquad (17)$$

Output Layer: The nodes in this layer denote the expected classes for each input. Unlike the conventional fuzzy production rules, we did not fuzzify the output labels, the connection weights existed between every node in the hidden layer and output layer as in the standard NN structure. These weights refer to the global weights (GW) which indicate the degree of importance of a certain rule contributing to the expected class. The output of each node represents the degree to which the input pattern belongs to the classes corresponding to the node. The activation function in this layer was set to:

$$O^k = \sum_j \left(GW_{j,k} * H_j\right), (k = 1,2, \dots, C_K) \qquad (18)$$

3.3.4 Neural Network Learning Process

We introduced the backpropagation algorithm to train the Neural Network, since it is the most popular and typical training algorithm in neural network (Yong, Xi-Zhong, & Qiang, 2003). The input layer $y_i^{(0)} (i = 1,2, \dots, N_0)$ (N_0 is the number of neurons in the input layer) is initialized to the membership values of the associated linguistic terms.

According to the weighted production rules, the hidden layer $y_j^{(1)}$ and the output layer $y_k^{(2)}$ can be calculated respectively as:

$$y_j^{(1)} = \bigwedge_{i=1}^{N_0} \left(LW_{i,j} * y_i^{(0)}\right) j = 1,2, \dots, N_1 \tag{19}$$

$$y_k^{(2)} = \sum_{j=1}^{N_1} \left(GW_{j,k} * y_i^{(1)}\right) k = 1,2, \dots, N_2 \tag{20}$$

where N_1 and N_2 are the number of neurons in the hidden layer and output layer respectively. We define the differentiable mean-square-error function as the error cost function of the OWFDT to be:

$$E = \frac{1}{2N} \sum_{n=1}^{N} \sum_{k=1}^{N_2} (y_k^{2,n} - d_k^{2,n})^2 \tag{21}$$

Where N is the total number of training instances, $d_k^{2,n}$ is the desired output for this n^{th} input pattern while $y_k^{2,n}$ is the actual output. The error E is the function with respect to LW and GW, which needs to be updated and optimized. According to the gradient descent method, the updating rules for weights should be written as:

$$LW_{ij} = LW_{ij} - \alpha \frac{\partial E}{\partial LW_{ij}} \ ; \tag{22}$$

$$GW_{jk} = GW_{jk} - \beta \frac{\partial E}{\partial GW_{jk}} \ ; \tag{23}$$

where α and β are learning rates for the hidden layer and output layer respectively. In this gradient descent algorithm, small learning rates often lead to a very low convergence speed, while large learning parameters will inevitably result in oscillations or even divergence. To avoid this issue, we naturally introduced the momentum term to our training process in which the change of the weights of the current iteration is made dependent partly on the weight changes of the previous iteration. These updated weights lead to a uniform decrease of error function. This can be achieved by adding a momentum term to equation 22 and 23, which leads to a modified weight update equation (Karray & Silva, 2005):

$$LW_{ij}^{t} = LW_{ij}^{t} - \alpha \frac{\partial E^{t}}{\partial LW_{ij}^{t}} + \gamma \Delta LW_{ij}^{t-1} ; \qquad (24)$$

$$GW_{jk}^{t} = GW_{jk}^{t} - \beta \frac{\partial E^{t}}{\partial GW_{jk}^{t}} + \gamma \Delta GW_{jk}^{t-1} ;$$

where *t is* the iteration number. The two partial derivatives in Equation 24 can be expressed as Equations 25 and 26 which can be derived using differentiation to equation 21.

3.3.5 Optimized Weighted Fuzzy Decision Tree Reasoning Mechanism

To improve the classification accuracy, we propose a new reasoning mechanism which combines the global weights and the constructed fuzzy decision tree. First, we redefine the certainty factor of each leaf node. Recall from the mapping process of FDT that each node will classify the input instance to different classes with a different certainty. This was determined when the FDT was constructed.

$$\frac{\partial E^t}{\partial GW_{ij}^t} = \begin{cases} \left(y_k^{2,t} - d_k^{2,t}\right) * y_j^{(1)} & \text{if } GW_{jk} * y_j^{(1)} \geq \bigvee_{q \neq j} (GW_{qj} * y_q^{(1)}) \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

$$\frac{\partial E^t}{\partial LW_{ij}^t} = \begin{cases} \sum_{k=1}^{N_2} \left(y_k^{2,t} - d_k^{2,t}\right) GW_{jk}^t * y_j^{(0,t)} & \text{if } GW_{jk} * y_j^{(1)} \geq \bigvee_{q \neq j} (GW_{qj} * y_q^{(1)}) \\ & \text{and } LW_{jk} * y_j^{(0)} \leq \bigwedge_{p \neq i} (LW_{pj} * y_p^{(0)}) \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

Take leaf node m as an example, the certainty factor of that leaf node can be calculated as:

$$\alpha_m = \left\{ \frac{\left|D_m^{C_1}\right|}{|D_m|}, \frac{\left|D_m^{C_2}\right|}{|D_m|}, \dots, \frac{\left|D_m^{C_K}\right|}{|D_m|} \right\} \tag{27}$$

As a hierarchical structure, a FDT shares the membership values along the paths leading to each leaf node from the root node. In Figure 3-1, $path_2$ has two segments $segm_{6,low}$ and $segm_{3,med}$, so the membership degree of $path_2$ can be calculated as:

$$\mu_{path_2}^i = \mu_{low}\left(F_6^i\right) * \mu_{med}\left(F_3^i\right) = \left[F_{6,l}^i, F_{3,m}^i\right] \tag{28}$$

For an arbitrary path m and input pattern i , the membership degree of $path_2$ can be denoted as:

$$\mu_{path_m}^i = \prod_j \mu_{\{low\,,medium\,,high\}}\left(F_j^i\right) \tag{29}$$

Figure 3-1 shows the basic structure of the FDT for the breast cancer dataset, in which seven leaf nodes are combined to produce a final conclusion. In our proposed reasoning mechanism, the certainty factor and the membership value of that path (leaf node) as well as the global weights are incorporated together to infer the classification result. For a random input pattern i , the possibility of falling into class $k(k = 1,2, \dots, C_K)$ at $path_2$ is given by:

$$\mu^i_{path_m} * \alpha_{m,k} * GW_{mk} \tag{30}$$

As a result, the prediction of all the leaf nodes for a particular class $k(k = 1,2, \dots, C_K)$ is combined to produce the possibility $y^i_k(k = 1,2, \dots, C_K)$ for the i^{th} pattern:

$$y^i_k = \sum_{m=1}^{M} \mu^i_{path_m} * \alpha_{m,k} * GW_{mk} \tag{31}$$

In Figure 3-1, the predicted possibilities for 'Benign' (y_1) and 'Malignant' (y_2) can be calculated respectively as:

$$y^i_1 = \sum_{m=1}^{M} \mu^i_{path_m} * \alpha_{m,1} * GW_{m1} \tag{32}$$

$$y^i_2 = \sum_{m=1}^{M} \mu^i_{path_m} * \alpha_{m,2} * GW_{m2} \tag{33}$$

If a crisp classification result is required, the class with the highest possibility can be selected as the expected class. For the breast cancer dataset, to classify a given instance either as 'Benign' or 'Malignant', the class corresponding to the maximum prediction possibility will be selected as:

$$O^i = argmax\{y_1^i, y_2^i\} \quad (34)$$

3.4 Simulation Results

In order to evaluate the performance of the proposed technique, we applied it to several benchmark datasets which were introduced in chapter two.

Table 3-1: Summary of the Datasets Employed

Database	Domain	Classes	Attributes	Samples
Iris	Botanic	3	4	150
Wisconsin Breast Cancer	Medical	2	30	569
Eye Bacteria	Medical	6	32	180
Brain Computer Interface	Medical	2	12	378
Composite Material	Material	3-5	4	534-7052

Table 3-1 summarizes all the datasets. The Iris dataset and Wisconsin dataset are widely used as benchmark datasets to test the performance of the intelligent systems techniques. The three other datasets were also introduced to evaluate the performance of OWFDT.

For the BCI dataset, where a training dataset (278 instances) and a test data set (100 instances) are present, the test dataset was classified based on the constructed and optimized OWFDT using the training dataset. For the other datasets, we randomly chose

50% of the dataset to grow, update and optimize the weights of the OWFDT and the other 50% was used as a test dataset to test the accuracy of the OWFDT.

Table 3-2: Testing accuracy of the employed datasets (in percentage)

Dataset	Fuzzy ID3	WFDT	OWFDT
Iris	90.67	93.33	96.00
Wisconsin Breast Cancer	92.98	93.31	96.49
Brain Computer Interface	89.00	90.00	92.00

Table 3-3: Training epochs of WFDT and OWFDT

Dataset	Fuzzy ID3	WFDT	OWFDT
Iris	n/a	186	50
Wisconsin Breast Cancer	n/a	479	143
Brain Computer Interface	n/a	354	208

Table 3-2 summarizes the mean test accuracies of the datasets employed. The classification accuracy was calculated as: $accuracy = n_r/n * 100$, where n is the total number of test instances and n_r is the number of instances which had been correctly classified. This simulation was repeated five times for each dataset. The accuracy was the average of the five. The table shows that the proposed optimization methods can improve the WFDT's accuracy for these datasets. The improvement is generally 2 to 3 percent higher than straight WFDT and typically 3 to 6 percent higher than Fuzzy ID3 in terms of accuracy.

Table 3-3 shows the average epochs OWFDT used to reach the minimum mean square error. The number of epochs was also calculated as the average of five simulations. It can be seen that OWFDT need less average epochs than WFDT in order to optimize the weights between the layers, so it demonstrates the efficiency of OWFDT over the weighted FDT while maintaining its effectiveness.

3.5 Conclusion

The incorporation of the attribute weights and the leaf nodes (paths) weights has enhanced the representational power and improved the accuracy of the WFDT. After mapping and converting the weighted fuzzy decision tree to a NN, these weights can be updated towards a convergent position. The learning performance and testing accuracy are improved in terms of the following aspects in our technique:

- The fuzzification of the input instances is more efficient and effective if the Fuzzy C-Means technique is introduced to determine the possible cluster centers, which can be used to fuzzify the data in terms of each linguistic term.
- Incorporating a momentum term to the gradient descent algorithm to train the NN can help the learning process to avoid local minima. In addition, it can avoid oscillations or divergence due to an inappropriately large learning rate or a low convergence rate due to a relatively small learning rate.
- The new reasoning mechanism proposed in our OWFDT has combined the leaf nodes weights with the leaf nodes certainty to infer the expected classification

results. This not only overcomes drawbacks in terms of not considering the leaf nodes' degree of importance in inferring the results of the traditional FDT inference mechanism but also enhances WFDT's reasoning mechanism by combining each leaf node's degree of importance with leaf nodes (path) weights to infer an unknown test instance.

The proposed OWFDT technique has shown its ability to improve the prediction accuracy of the Fuzzy ID3 and enhancement of WFDT's ability to precisely infer an unseen instance through the weighted production rules.

In the next three chapters, OWFDT will be introduced to be applied to three datasets to evaluate its performance over traditional FDT and weighted FDT. In chapter six, a benchmarking study will be conducted to compare its effectiveness, in terms of its accuracy and the number of rules generated by OWFDT, with some other hybrid techniques such as ANFIS, EFuNN etc.

Reference

Black, C. L., & Merz, C. J. UCI repository of machine learning databases,University of California, Department of Information and Computer Science. http://www.ics.uci.edu/mlearn/MLRepository.html

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and Regression Trees*: Chapman and Hall/CRC.

Janikow, C. Z. (1998). Fuzzy decision trees: issues and methods. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 28*(1), 1-14.

Karray, F. O., & Silva, C. W. D. (2005). *Soft Computing and Intelligent Systems Design: Theory, Tools and Applications*: Addison-Wesley.

Levashenko, V., Zaitseva, E., & Puuronen, S. (2007). *Fuzzy Classifier Based on Fuzzy Decision Tree.* Paper presented at the EUROCON, 2007. The International Conference on Computer as a Tool.

Mitra, S., Konwar, K. M., & Pal, S. K. (2002). Fuzzy decision tree, linguistic rules and fuzzy knowledge-based network: generation and evaluation. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 32*(4), 328-339.

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning, 1*(1), 81-106.

R.Weber. (1992). *Fuzzy-ID3: a class of methods for automatic knowledge acquisition.* Paper presented at the 2nd Internat. Conf. on Fuzzy Logic and Neural Networks, Iizuka, Japan.

Suarez, A., & Lutsko, J. F. (1999). Globally optimal fuzzy decision trees for classification and regression. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 21*(12), 1297-1311.

Tsang, E. C. C., Wang, X. Z., & Yeung, D. S. (2000). Improving learning accuracy of fuzzy decision trees by hybrid neural networks. *Fuzzy Systems, IEEE Transactions on, 8*(5), 601-614.

Umanol, M., Okamoto, H., Hatono, I., Tamura, H., Kawachi, F., Umedzu, S., et al. (1994). *Fuzzy decision trees by fuzzy ID3 algorithm and its application to diagnosis systems.* Paper presented at the Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the Third IEEE Conference on.

Wang, X., Chen, B., Qian, G., & Ye, F. (2000). On the optimization of fuzzy decision trees. *Fuzzy Sets and Systems, 112*(1), 117-125.

Wang, X. Z., Yeung, D. S., & Tsang, E. C. C. (2001). A comparative study on heuristic algorithms for generating fuzzy decision trees. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 31*(2), 215-226.

Yeung, D. S., Wang, X. Z., & Tsang, E. C. C. (1999). *Learning weighted fuzzy rules from examples with mixed attributes by fuzzy decision trees.* Paper presented at the Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on.

Yong, L., Xi-Zhong, W., & Qiang, H. (2003). *Using BP-network to construct fuzzy decision tree with composite attributes.* Paper presented at the Machine Learning and Cybernetics, 2003 International Conference on.

Yuan, Y., & Shaw, M. J. (1995). Induction of fuzzy decision trees. *Fuzzy Sets Syst., 69*(2), 125-139.

Zhiheng, H., Gedeon, T. D., & Nikravesh, M. (2008). Pattern Trees Induction: A New Machine Learning Method. *Fuzzy Systems, IEEE Transactions on, 16*(4), 958-970.

Chapter IV: Pattern Recognition of Fiber-Reinforced Plastic Failure Mechanism: Using Intelligent Systems Techniques

Chapter three introduced the Optimized Weighted Fuzzy Decision Tree (OWFDT) which is theoretically based on FL, ANN and DT and verified its efficiency and effectiveness using several benchmark datasets. In this chapter OWFDT will be used to design a classifier to predict the possible failure mechanisms in composite materials. Before designing the classifier, seven datasets were collected by sampling acoustic emission events signals in an attempt to find out the possible failure mechanisms in composite material. SOM and FCM will be used to cluster the datasets into different groups representing different failure mechanisms.

4.1 Introduction

The identification of the type of discontinuities or failure mechanisms within fiber-reinforced plastic (FRP) structures normally requires the use of local nondestructive testing (NDT) methods, among which, acoustic emission or acoustic event (AE) has been widely used due to its high sensitivity. AE is a source of elastic waves when external stimuli such as mechanical loading are imposed to composite material. AE is generated by the material itself and once generated, will be detected by AE sensors and acquired by the AE data acquisition system. The recorded AE data are in the form of signal parameters, such as amplitude, duration, signal strength and the number of acoustic events. These are the key parameters in the structural evaluation of the material. Normally, FRPs consist of more than one material. As a result, the failure mechanism will occur in one or a combination of materials. The various types of damage mechanism in FRP are matrix cracking, fiber breakage, fiber-matrix debonding, delamination, and fiber pullout (Ativitavas, Pothisiri, & Fowler, 2006).

Many research works have focused on observing AE signal parameters to identify failure mechanisms (Godin, Huguet, Gaertner, & Salmon, 2004; Huguet, Godin, Gaertner, Salmon, & Villard, 2002; Johnson, 2002). This is usually done by using some Intelligent Systems Techniques (ISTs) to perform pattern recognition of AE data relating to different failure mechanisms. The aim of the present study is to use intelligent systems to identify the unique characteristics of different failure mechanisms from AE data so that it can recognize the distinctive pattern of an unknown composite material failure mechanism.

For the current study, pattern recognition by means of ISTs is applied to a group of AE bursts rather than to individual sensor hits. This is because an individual hit, which may be one among thousands, is not likely to be representative of emission. In the work here we use two widely used approaches Self-Organizing Map (SOM) and Fuzzy C-means (FCM). These two techniques are used to classify AE bursts into different groups which represent different failure mechanisms. After that OWFDT is introduced to design the classifier to predict possible failure mechanism.

4.2 Test Apparatus and Procedure

In this section we first give a brief introduction to the AE equipment and the AE software system. After that a description of the test procedure is included in the second subsection.

4.2.1 Acoustic emission equipment

Due to the requirement to have a flexible and feasible process for manufacturing the test coupons in the current research, ‘hand lay up’ was the most suitable solution, because by using ‘hand lay up’ it is possible to control fibre volume fraction, fibre orientation and curing parameters. The fiber glass volume fraction that is normally achieved by this manufacturing process is 57%. All the Glass/Polyester plates were manufactured in-house in order to have a closer control of the parameters that influence the mechanical properties and ultimately the acoustic response of the materials. These parameters included fiber orientation and the dimensions of the coupons which will be used for the test. The coupon dimensions for the current research were 20 mm x 200 mm x 1.8 mm with three layers of unidirectional glass tape. Each plate was cut into different orientations in 15° steps against the direction of the load, which lead to seven different species (0°, 15°, 30°, 45°, 60°, 75° and 90°).

A broadband WD piezoelectric sensor manufactured by Physical Acoustic Corporation was used to capture the stress waves (AE events). The acoustic emission software was the TRA (transient recorder package) from Physical Acoustics Corporation 2001 (Pollock, 1989). The hardware was a sound card AEDSP-32 (Acoustic Emissions Digital Signal Processor) with a digital signal processor by Texas Instruments (TMS320C40). It has a maximum sampling speed of 8 MHz and 16 bit resolution, between 20 and 100 dB. After capturing the event, the piezoelectric sensor transforms the mechanical deformation of the material surface into an electrical signal. This signal is transferred to the sound card, which processes it and displays it as a wave signal as shown in Figure 4-2. The software used with this sound card can display on screen the transient waveform for each event. Each event can then be individually exported to an ASCII file, which is then processed by a program written in the "R" programming language (open source code language). With this program it is possible to perform and visualize a Fast Fourier Transformation (FFT) of each event and select the most dominant frequencies, which are to be used as the descriptor of the event. The distribution of these dominant frequencies as the test progresses and the changes of orientation will show the trends of the events occurring in the test specimen.

From the literature review (de Groot, Wijnen, & Janssen, 1995; Eduard, Hines, Gardner, & Franco, 1999; El Kadi, 2006; Godin, et al., 2004), it appears that there are several different approaches to the task of event recognition analysis, from simple hit analysis to amplitude-based and frequency-based studies. But the amplitude recorded at a sensor depends upon the distance the sensor is from the source of the emission and the amplitude of an acoustic wave is attenuated as the wave passes through the material. As compared with the amplitude, the frequency can be recorded and analyzed in a more objective and accurate way. Based on this, the dominant frequency of the signals was considered to be the most relevant characteristic for the present study.

4.2.2 Test procedure

According to Qi (Qi, 2000), 90% of AE activities for glass fiber reinforced (GFR) composite materials are concentrated in the frequency range 10–550 kHz. One wide band sensor as shown in Figure 4-1 was attached to the specimen by means of a G aluminum clamp with a plastic screw. The surface of the sensor was covered with silicon grease in order to provide good acoustic coupling between the specimen and the sensor. The signal was detected by the sensor and enhanced by a 2/4/6-AST pre-amplifier.

Figure 4-1 shows the test configuration, with the sensor clamped to the test specimen and the INSTRON grips where used to apply the tensile force.

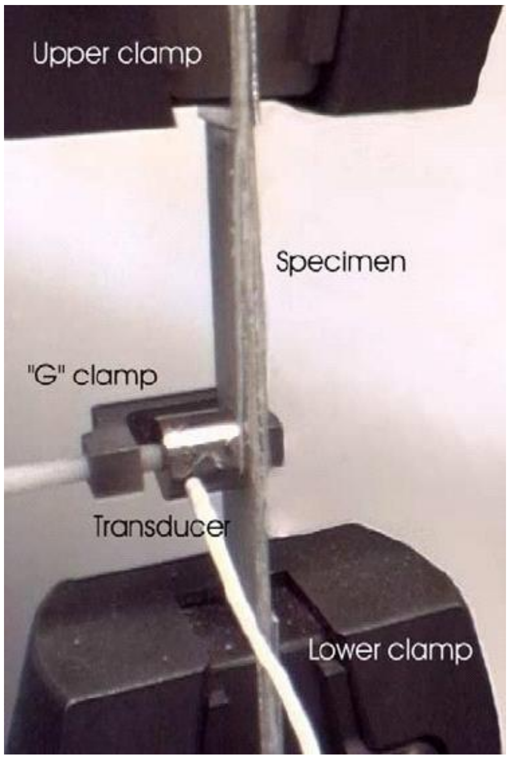


Figure 4-1: Test configuration, sensor coupled to the specimen using a G clamp.

4.3 Data Analysis

Initially, the MISTRAS software that records the acoustic events from each test was used (Pollock, 1989). It was observed that as each test progressed, the number of events increases significantly. Nevertheless, the data do not provide any information about the type of failure or the location of the failure. The TRA software, although using the same hardware, acquires the information in a different way; it records the complete waveform and stores the event. A Fourier transformation is then applied to define the frequencies related with these waveforms. According to Qi (Qi, 2000) these frequencies are useful when one is trying to ‘‘distinguish different AE signals from various possible failure modes in fiber reinforced composites’’. A waveform event contains several component frequencies, and it is not possible to determine them visually, therefore a FFT was introduced for each waveform to produce a power vs. frequency graph of a waveform (power spectrum). Higher powers contained at particular frequencies are then shown as peaks on this representation of the waveform. If no single frequency is dominant on the waveform, more than one frequency which account for the power will be extracted as the descriptors of the waveform.

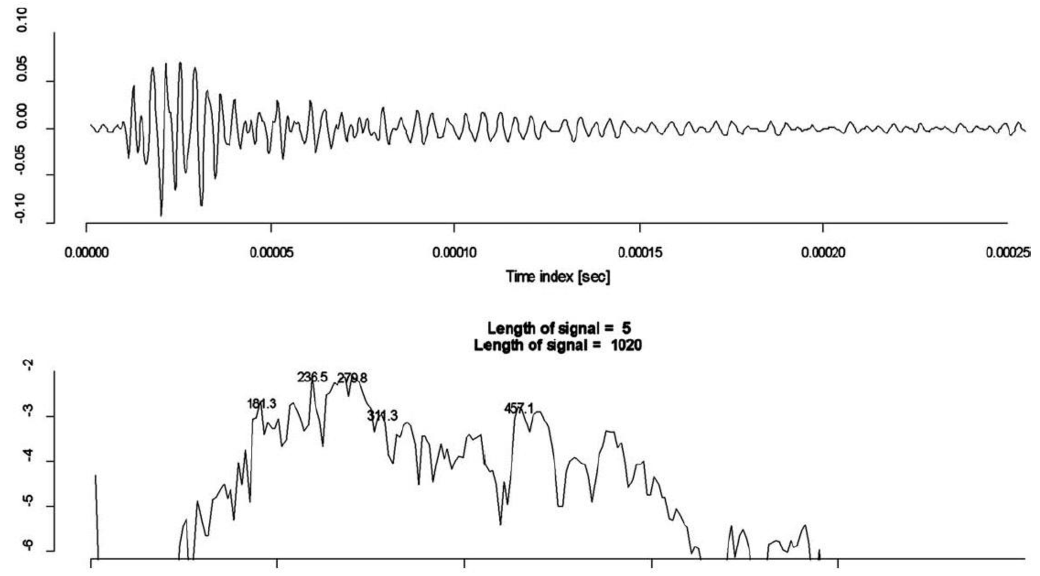


Figure 4-2: (a), (top) waveform of a transient acoustic event; (b) (bottom) FFT power spectrum

Although previous publications suggested that the frequency should be employed to differentiate between events, only one frequency is mentioned (Bohse, 2000; de Groot, et al., 1995; Godin, Huguet, & Gaertner, 2005). These studies had shown that frequency could be used as an identifier for different failure events in composite materials, nevertheless studies also showed that using only the first frequency as a descriptor is not enough to produce optimal results (Giordano, Calabrò, Esposito, Salucci, & Nicolais, 1999; Sause & Horn, 2010).

Tests conducted by Ramirez-Jimenez had found that although the 1st highest frequency may contribute about 50% of the total power of the waveform (Ramirez-Jimenez, et al., 2004), the remaining highest power frequencies also contribute a significant amount of the power. So even though the frequency with the highest power value can be used as an identifier for the events, however, this frequency was not enough to thoroughly describe the behavior of the composite material.

The program that generates the FFT identifies by default the five frequencies from each signal which have the highest power peaks as shown in Figure 4-2: (a) transient event waveform; and (b) FFT power spectrum. Sometimes the power difference between the highest power frequency and the subsequent lower power frequencies is so close to each other that it can be assumed that a single frequency cannot constitute the definition of that waveform. So it will be more practical to use the first two highest frequencies to characterize each acoustic event.

The FFT power spectrum of a typical AE event in Figure 4-2 shows several peaks, the highest one being 280 kHz and having a second higher peak at 236 kHz. The FFT power spectrum can be used as a “fingerprint” for each event and therefore may be used as a means of distinguishing between them. Because a single acoustic event may consist of several individual sources, and each source exhibits a specific frequency and no single frequency can dominate in the power spectrum, which can explain why a single waveform has to be denoted by different frequencies.

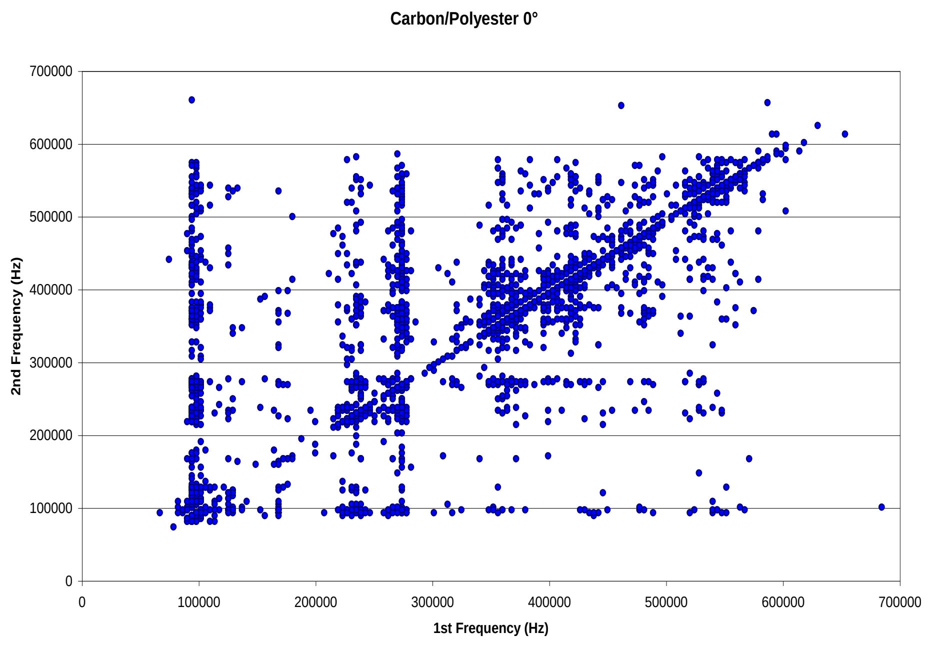


Figure 4-3: Relationship between the 1st and 2nd highest power frequencies

Figure 4-3 shows a plot of the 1st frequency against the 2nd frequency from one set of FFTs calculated from waveforms recorded on a tensile test of 0° Carbon/Polyester. It can be seen that there is some clustering of values where a relationship could be drawn, however according to Papadakis (Papadakis, 2004), a simple presentation of these frequencies is not enough to differentiate the events that originated each waveform.

In what follows, a Principal Component Analysis (PCA) is conducted to explore possible clustering and hence inform the classification of the AE data recorded from the tests. The aim of PCA exploration is to figure out whether or not simple clusters exist in the dataset and how the patterns cluster in the representation of the frequency space. Figure 4-4 shows the result of the PCA analysis using the 1st, 2nd, 3rd and 4th highest power frequencies from a tensile test of Glass/Polyester 0°.

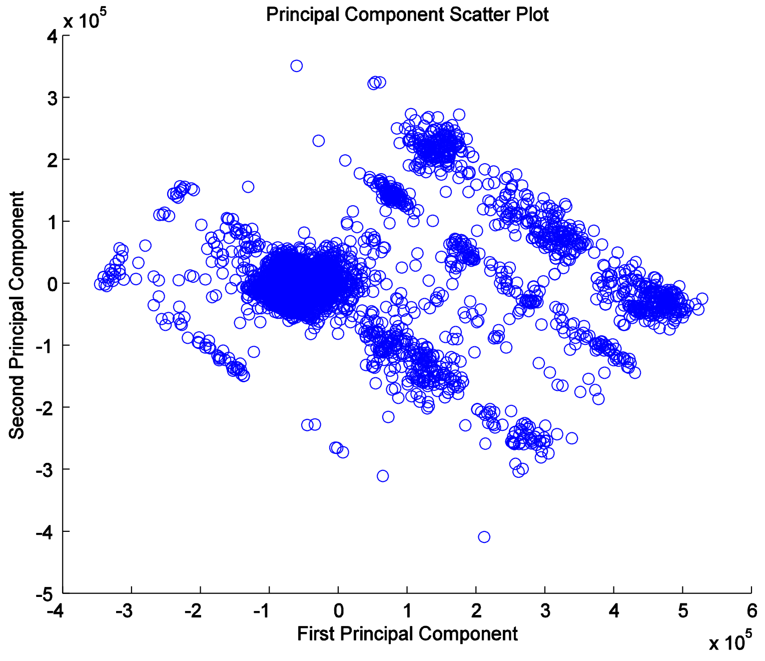


Figure 4-4: PCA for the frequencies from Glass/Polyester 0°

In the result of PCA in Figure 4-4, the first two principal components were kept which accounted for 91% of the variance, and there is no clear discrimination between the various clusters representing the frequency. But after a linear projection of multidimensional data onto different coordinates, the PCA analysis had eliminated those less significant components and reduced the dataset to those components which are responsible for the most significant contribution. So the information on the coordinates of data points in the new coordinates can be obtained and used to study the clustering of data. In the following study the two principal components will be used as the data input to SOM in an attempt to investigate the possible clusters.

4.4 Test Results

In the following sections two widely used IS techniques, mainly SOM and FCM are introduced in order to find any patterns in the dataset and link any patterns found with possible failure mechanism using two principal components information.

We applied SOM and FCM to 7 datasets from 7 different Glass/Polyester fiber orientations tests shown in Figure 4-5 as follows:

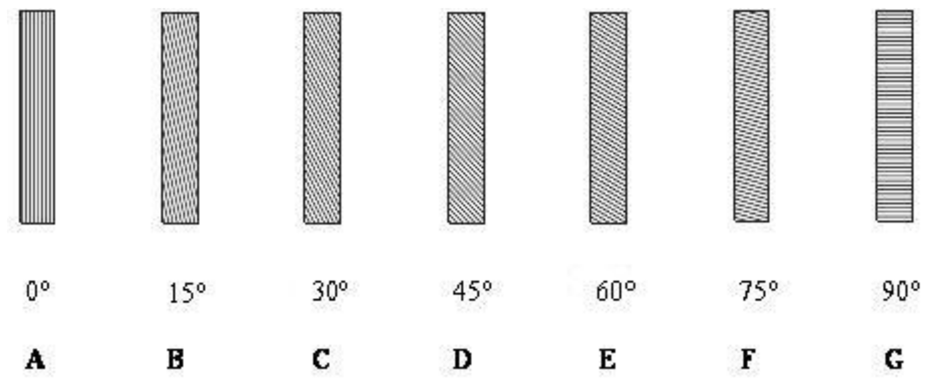


Figure 4-5: Diagram of fiber orientations

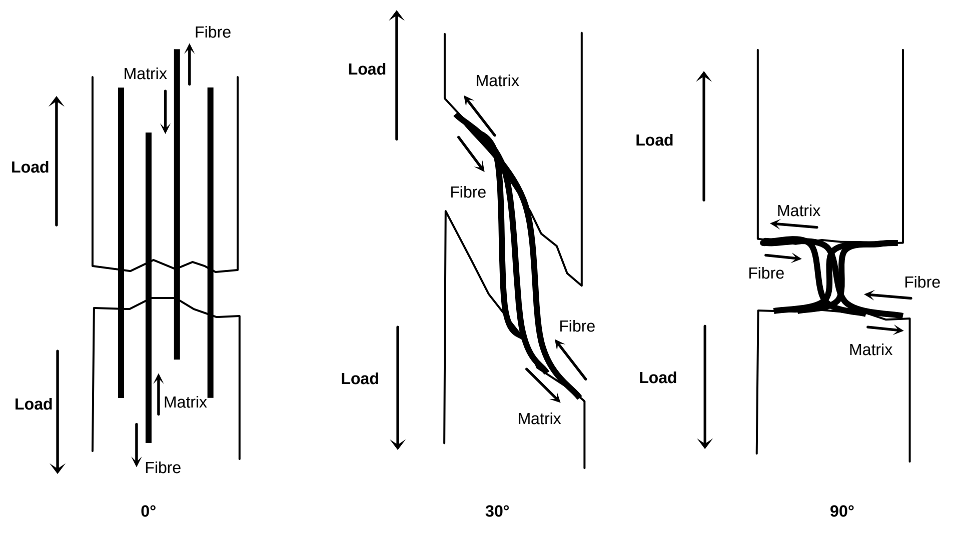


Figure 4-6: Behaviour of fibre pull out for three different configurations, 0°, 30° and 90° Glass/Polyester subjected to tensile load

Figure 4-6 shows the behavior of the fibers as the composite is subjected to tensile load by presenting three different and typical configurations as 0°, 30° and 90°.

The SOM and FCM clustering processes are usually unsupervised. They will result in the best solution after a number of iterations (Babuska, 1998; Kohonen, 2001; Lampinen, Laurikkala, Koivisto, & Honkanen, 2005).

The SOM was constructed to consist of 5 neurons since there are no more than 5 classes in all the seven tests, namely matrix cracking, fiber/matrix debonding, fiber/matrix pull out and fiber and matrix breakage. These classes were defined based on the clustering of the neurons. For some tests a single class may capture more than one neuron, therefore that class is denoted by the acoustic events represented by more than one neuron. The bar plot for each test will be shown next to the SOM output map where the clusters for each corresponding class have been drawn.

The FCM algorithm starts with an initial guess for the cluster centers, which are intended to mark the mean location of each cluster. The initial guess for these cluster centers is most likely to be incorrect. Additionally, FCM assigns every data point a membership grade for each cluster. By iteratively updating the cluster centers and the membership grades for each data point, FCM iteratively moves the cluster centers to the right location within this dataset. This process is based on minimizing an objective function that represents the distance from any given data point to a cluster center weighted by that data point's membership grade. In the following simulations two highest frequencies denoting every AE were used as input for FCM clustering.

We will now consider Glass/Ployester response at 0°, 15°, 30°, 45°, 60°, 75° and 90°.

4.4.1 0° Fiber orientation

The clustering process can be supervised or unsupervised as explained in chapter two. A supervised clustering process is presented with expected output, for example a fixed number of classes, whereas the unsupervised process will result in the best solution after several repetitions. Based on the process explained in chapter two, SOM and FCM were applied to the data from the 7 different fibre orientations as shown in Figure 4-5. Here we focus on the 0° degree test as the example.

Figure 4-7 shows the SOM output map with the 5 neurons positioned on their correspondent locations after 1,000 iterations. Notice that some of the neurons are closer to each other (i.e. neurons 4 and 5, neuron 1 and 2), these neurons are grouped into a single class by the SOM. Therefore the values that had been related to each neuron by the network will be considered to belong to that class. Once those neurons that are in the same vicinity are identified by the network as belonging to the same class, then a plot with the number of events corresponding to each neuron is created.

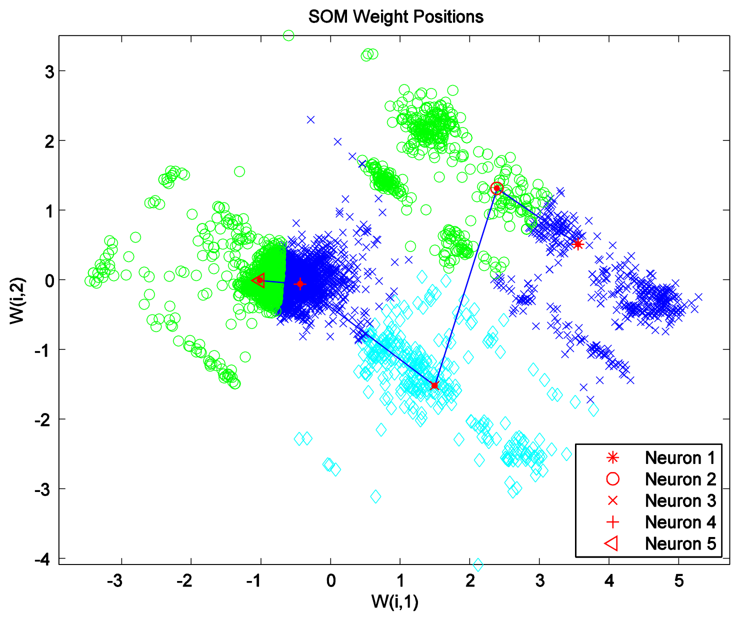


Figure 4-7: SOM output for 0° Glass/Polyester

Apart from that we have some useful visualized figures which enable us to interpret the results from a more straightforward perspective. Figure 4-8 shows the neighbor neurons distances within the SOM map, which indicates the distances between neighboring neurons. The blue pentagons represent the neurons and the line connects neighboring neurons. The color containing the lines indicates the distance between neurons; with dark colors representing larger distances and light colors representing smaller distances. As we can see, a group of light colored segments appears in the upper and bottom regions respectively, bounded by some darker segments. This grouping indicates that the data relating to the neuron 1 and 2 should be clustered into a single group and neuron 4 and 5 should be clustered into a single group as well. These groups can also be seen in Figure 4-7. The left region of Figure 4-7 contains several tightly clustered neurons. The corresponding weights are closer together and the positions are tighter in this region, which is indicated by the lighter colors in the neighbor distance Figure 4-8. The color in the middle region of Figure 4-8 is darker than those in the upper and lower regions, since the distance from neuron 2 to neuron 3 is larger. This color difference is understood to indicate that data points in this region are farther apart, which is confirmed in Figure 4-7.

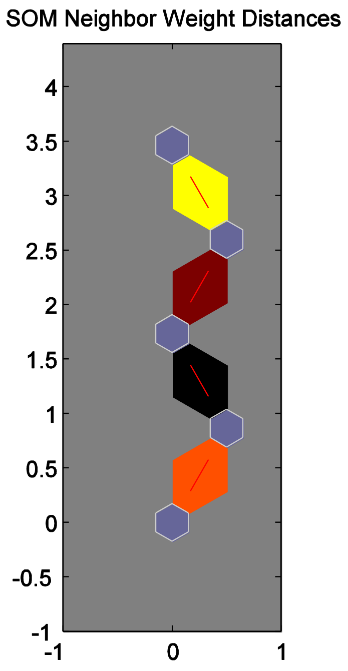


Figure 4-8: SOM Neighbor Weight Distances

In addition, we have also produced another figure which directly shows how many data points are associated with each neuron. It can be seen from Figure 4-9 that this dataset is concentrated a little more in the upper region; with neuron 4 denoting 4429 events and neuron 5 denoting 1292 events in the test.

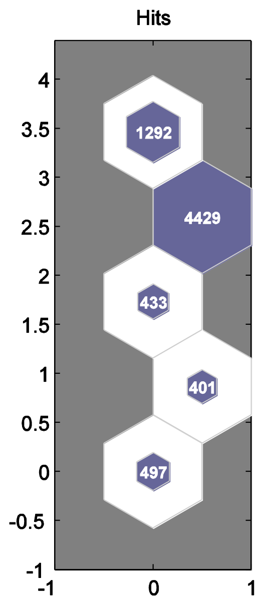


Figure 4-9: SOM sample hits for each neuron

The SOM result can also be justified using FCM (see chapter two). The FCM algorithm starts with an initial guess for the cluster centers, which is intended to mark the mean location of each cluster. The ‘initial guess’ for these cluster centers is subject to training and updating. Additionally, FCM assigns every data point a membership grade for each cluster. By iteratively updating the cluster centers and the membership grades for each data point, FCM iteratively moves the cluster centers to the right location within this dataset. This process is based on minimizing an objective function that represents the distance from any given data point to a cluster center weighted by that data point’s membership grade. As indicated in chapter two, the success of the clustering depends heavily on the choice of the prospective number of the classes. Fortunately, a number of clustering indices can be used as guidance as to how many clusters or classes are present in a data set although they are not the absolute indicators for deciding how many clusters there existed. Figure 4-10 shows 7 possible clustering indices for the 0° degree data. For many of the indices, mainly the Partition Coefficient (PC), Classification Entropy (CE), Partition Index (PI), Xie and Beni Index (XB) and Dunn Index (DI) (Balasko, Abonyi, & Feil, 2004), there appears to be a strong implication of 3 clusters, justifying the conclusion that the dataset should be classified as three clusters.

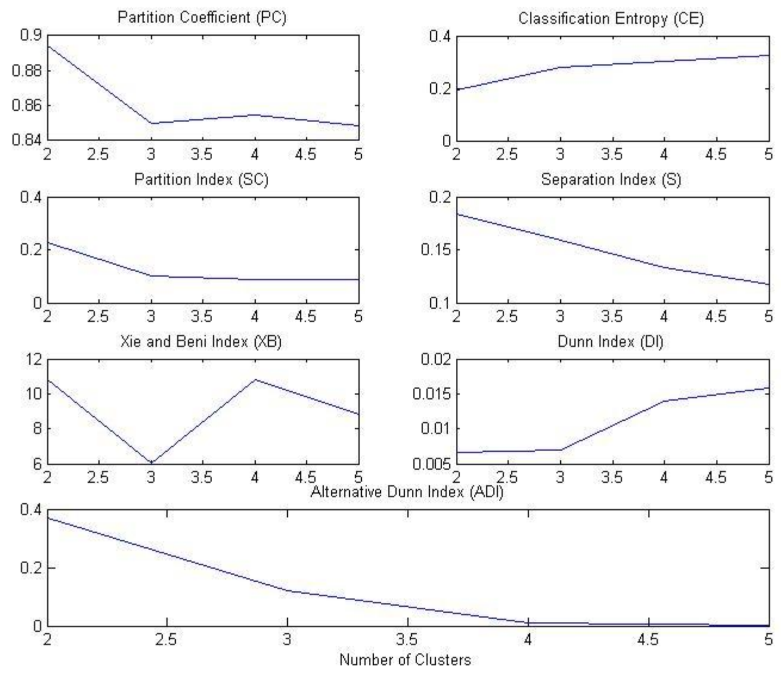


Figure 4-10: Clustering indexes showing the No. of classes versus the index

The FCM algorithm converges after 13 iterations. Figure 4-11 shows three clusters in the map which were all represented by a cluster center. The first cluster occurred at about 2.5 kHz, while the second and the third occurred at about 4.5 kHz and 5.5 kHz respectively. Each cluster consists of some events (with the first cluster having 5761 events, the second and the third cluster 457 and 834 events respectively) which amount to 7052 events in total.

In order to compare the consistency of these two algorithms, we plotted a vertical bar graph showing the percentage of events in each class for each algorithm. Figure 4-12 shows the number of events per class. The X axis accounts for the 5 different classes found during the training process whilst the Y axis has been normalized against the number of total events for that test. This is done so that it is easier to compare the relative population for each class per test across all the tests.

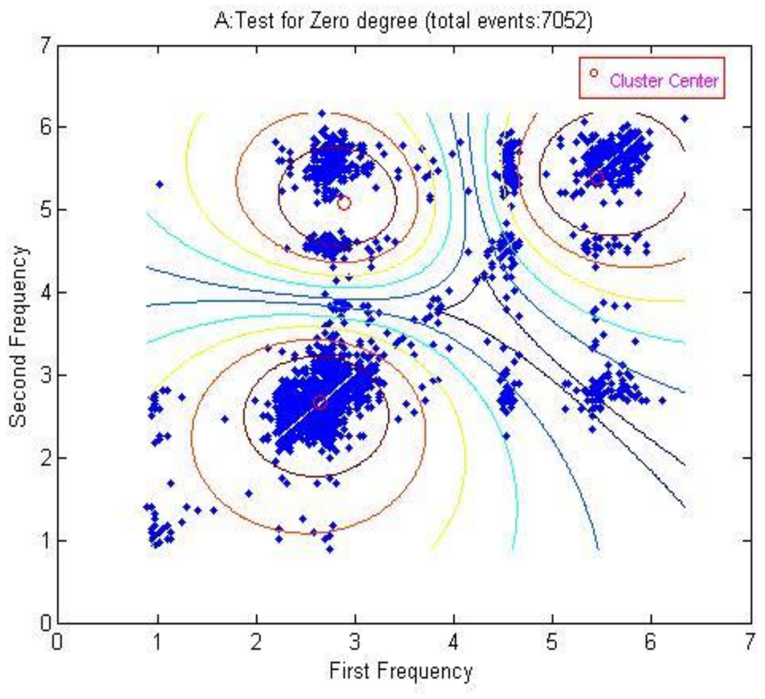


Figure 4-11: FCM output for 0° Glass/Polyester

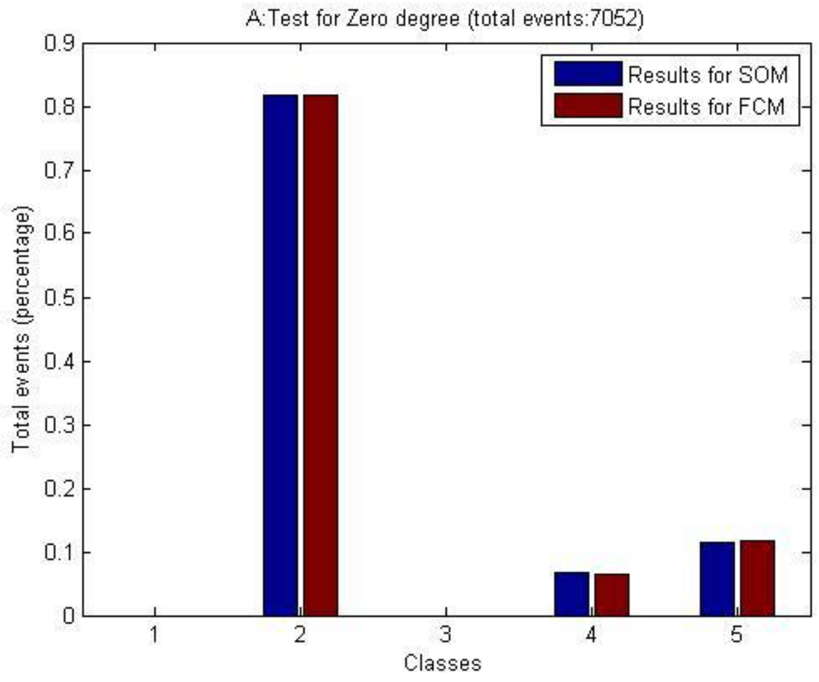


Figure 4-12: Normalized events per class for 0° Glass/Polyester

4.4.2 15° Fiber orientation

Figure 4-13 shows the SOM output map for 15° Glass/Polyester. The output neurons have been positioned by the network in different locations compared to those found for 0° Glass/Polyester, see Figure 4-7.

For this test a new class has appeared (Class 1) corresponding to neuron 5. Class 2 contains the values gathered using neurons 3 and 4 accounting for almost 60% of the total events of the test, see Figure 4-13.

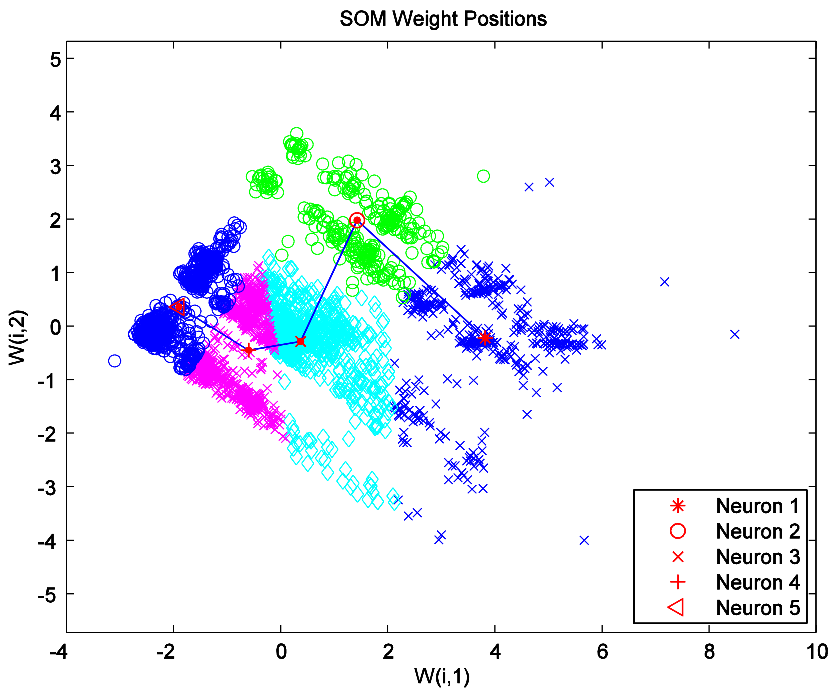


Figure 4-13: SOM output for 15° Glass/Polyester

Figure 4-14 shows the FCM output after it converges after 23 iterations. This was achieved after analyzing the same index listed in Figure 4-10. There are clearly four clusters in the map and each is represented by a cluster center. As in the case of the SOM, the cluster which consists of 3027 events dominates the classes while the other three consists of 806, 768 and 850 events respectively.

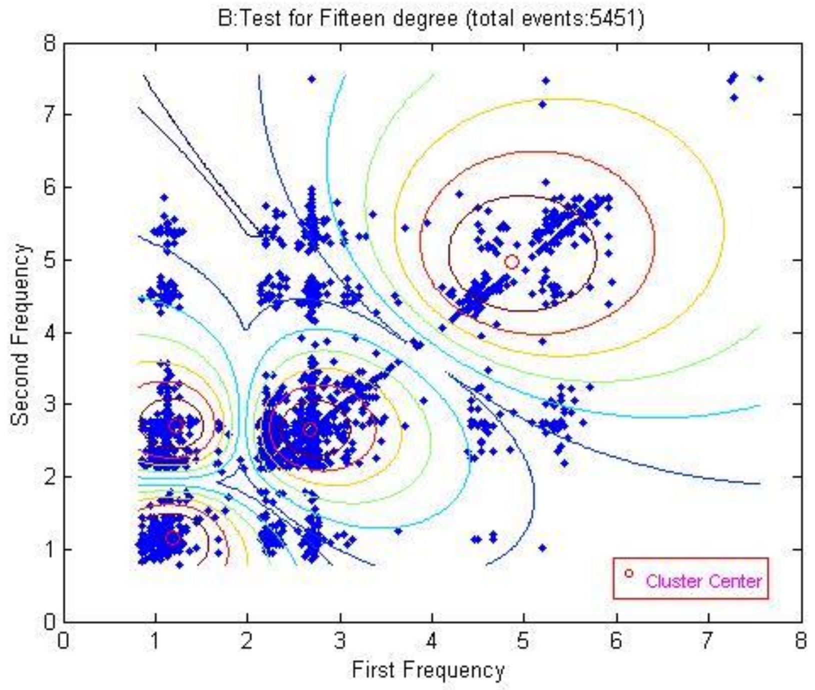


Figure 4-14: FCM output for 15° Glass/Polyester.

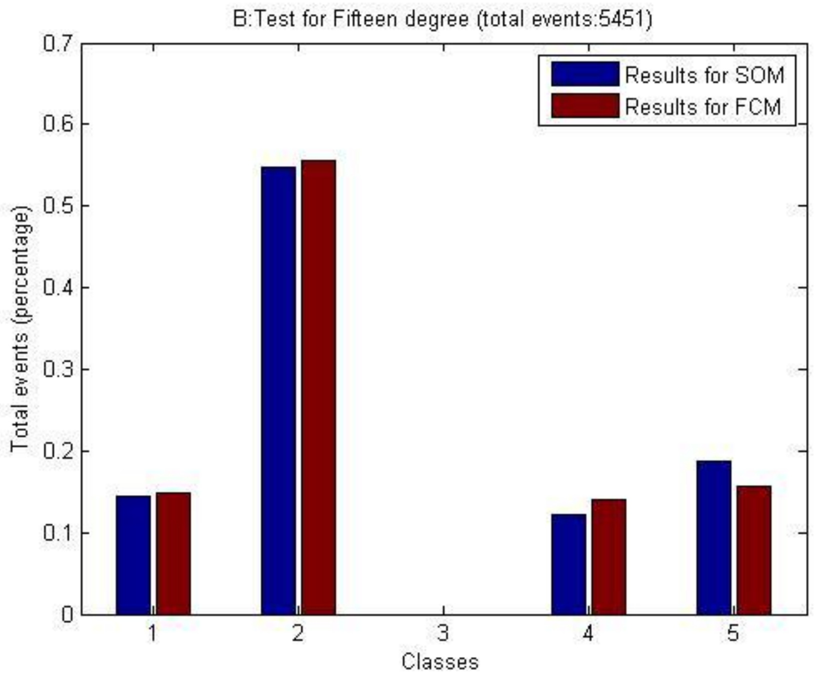


Figure 4-15: Normalized events per class for 15° Glass/Polyester

Figure 4-15 shows a bar graph of the percentage of events per class for 15° Glass/Polyester. Although there are some slight differences between FCM and SOM, again Class 2 accounts for the majority of the events while the other three classes (Class 1, Class 4 and Class 5) account for about 15% each for both algorithms.

4.4.3 30° Fiber orientation

Figure 4-16 and Figure 4-17 show the SOM and FCM output map for 30° Glass/Polyester. Four classes were found in both figures; in Figure 4-16, neuron 5 corresponded to Class 5, neurons 4 to Class 4, neurons 3 and 2 to Class 2 and neurons 1 to Class 1. FCM converges after 41 iterations. And both algorithms indicate that the second class dominates all the events. The distribution of events across the classes in Figure 4-18 for the 30° fiber orientation, showed a similar tendency to that for the 15° bar chart shown in Figure 4-15. Nevertheless the total number of events is significantly different for both tests; the 15° fiber orientation has almost 5 times more events (5451) than the 30° test (1251).

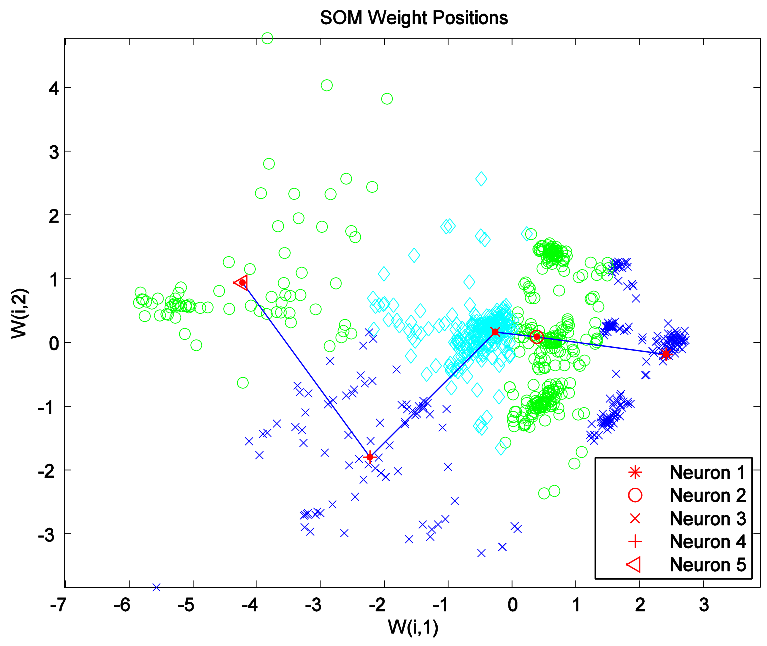


Figure 4-16: SOM output for 30° Glass/Polyester

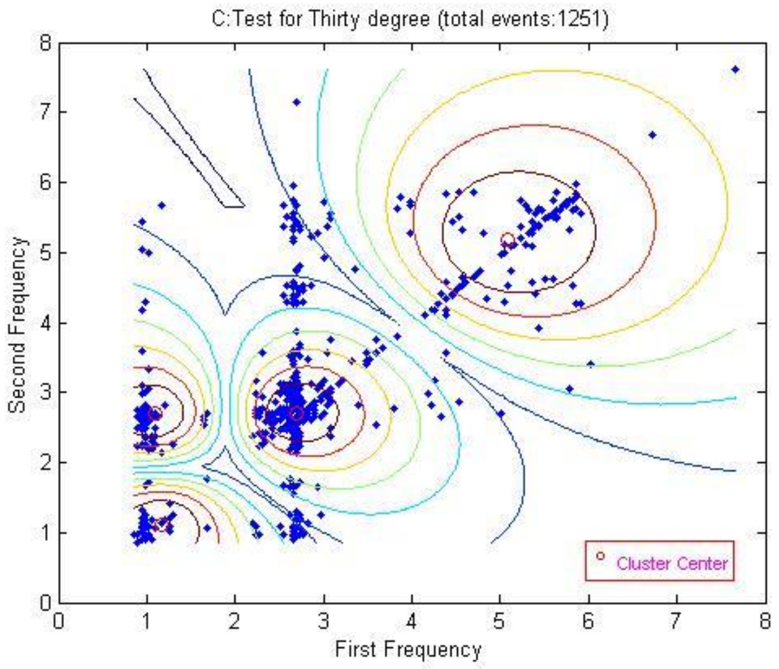


Figure 4-17: FCM output for 30° Glass/Polyester

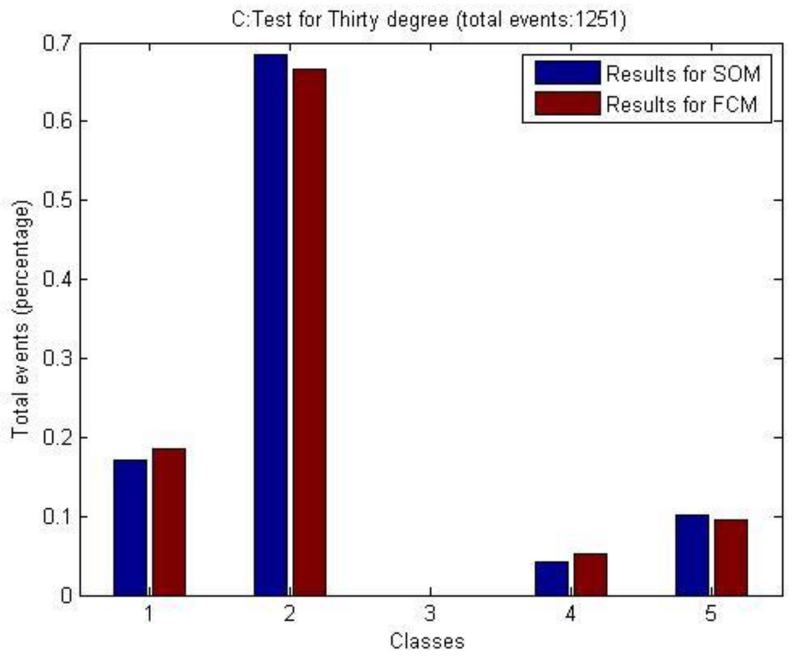


Figure 4-18: Normalized events per class for 30° Glass/Polyester

Figure 4-18 shows the normalized number of events corresponding to each class for the SOM and FCM. It can be seen that for both algorithms Class 3 has no events for 30° fibber orientation and class 2 contributes with almost 70% of the events. The other classes contributes less than 40% for the rest of the events.

4.4.4 45° Fiber orientation

One special fiber configuration is the 45° against the direction of the load; its SOM output and FCM output are shown in Figure 4-19 and Figure 4-20 respectively. 4 Classes are identified for the 45° test, which is a natural output following the previous fiber orientation test results (Figure 4-12, Figure 4-15, and Figure 4-18) as they rotate away from the direction of the load. The number of events for each class for the 45° fiber orientation test results showed a more even distribution than the previous ones, 20%

for Class 1, 36% for Class 2, 18% for Class 4 and 26% for Class 5 in the case of SOM and 20% for Class 1, 40% for Class 2, 18% for Class 4 and 22% for Class 5 in the case of FCM.

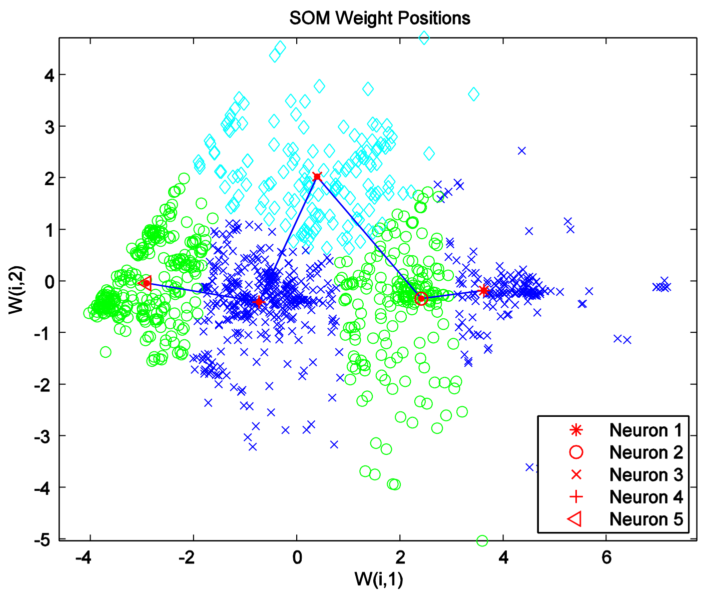


Figure 4-19: SOM output for 45° Glass/Polyester

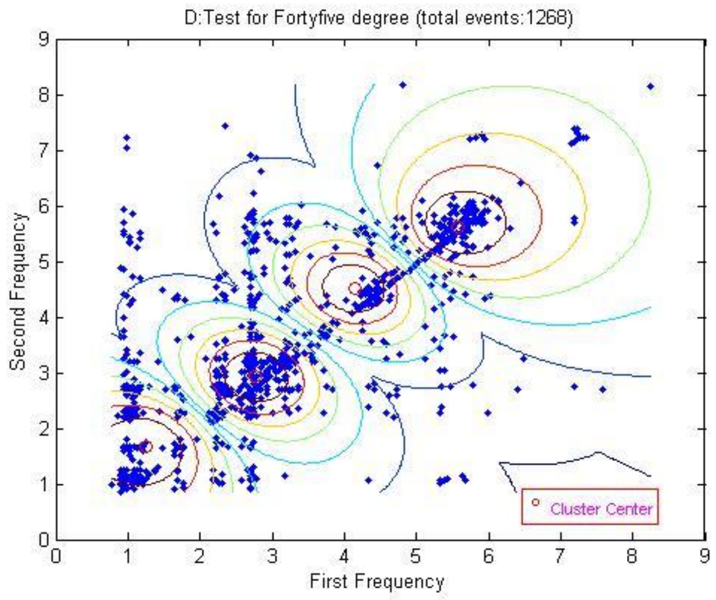


Figure 4-20: FCM output for 45° Glass/Polyester

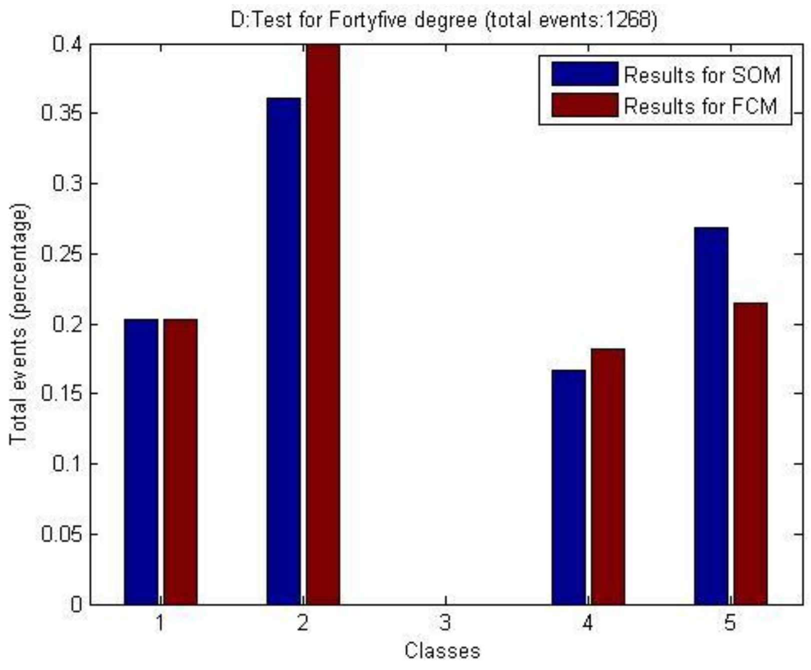


Figure 4-21: Normalized events per class for 45° Glass/Polyester

Figure 4-21 shows the classes that were constructed from the 45° Glass/Polyester SOM neuron map and FCM clustering respectively. The micro-mechanical events appearing during this test are the most evenly distributed in all the seven tests due to the orientation of the fibers.

Between 15° and 45° fiber orientations, all the classes were represented except for Class 3. For the 0° fiber orientated material Class 1 was also missing. Consideration of the phenomenon related to their micro-mechanical failure events will be described in the following sub-sections in this section.

4.4.5 60° Fiber orientation

From the test for 60°and up to the 90° Classes 4 and 5 were not represented and instead Class 3 appeared. As can be seen in Figure 4-22 the SOM output neurons and in Figure 4-23 the FCM clusters for 60° show a clustering of neurons in classes where Classes 4 and 5 are not present.

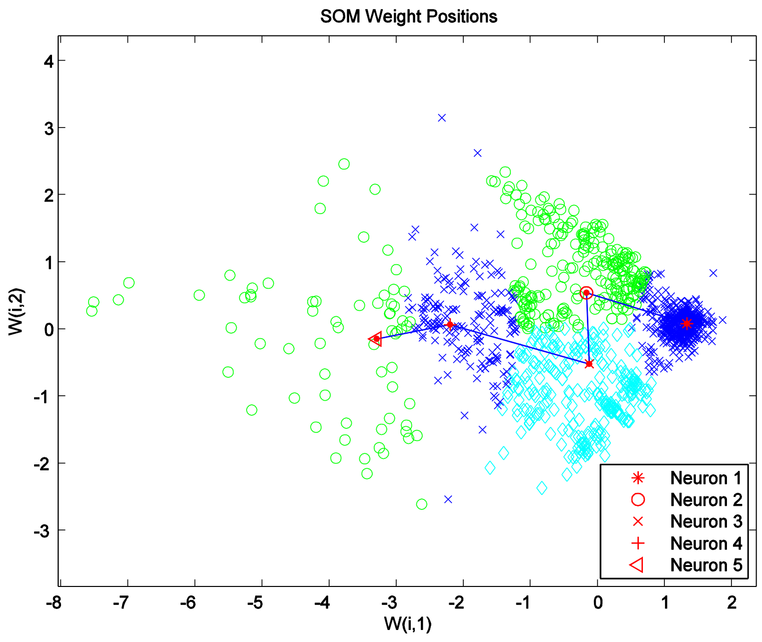


Figure 4-22: SOM output for 60° Glass/Polyester

In the case of the SOM algorithm, neurons 4 and 5 represent Class 1; neurons 3 and 2 represent Class 2 and neurons 1 is grouped into Class 3. In the case of FCM (Figure 4-23) it converges to three typical classes after 54 iterations. Starting from this fiber orientation and up until the 90° configuration Class 3 appeared and showed an increase in the number of events related to it as the fiber rotates away from the direction parallel to the load. On the other hand for this same offset of the fibers, Class 2 has gradually reduced in size as tests proceed from 60° to 90°.

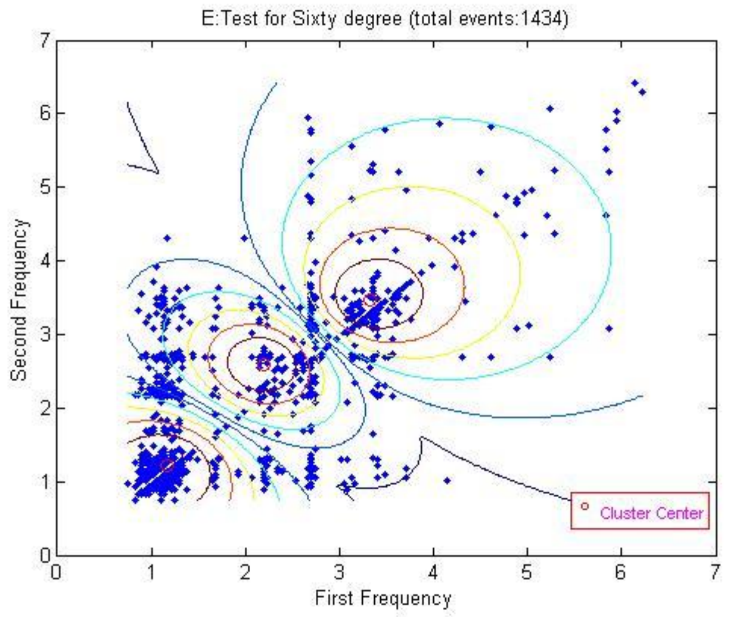


Figure 4-23: FCM output for 60° Glass/Polyester

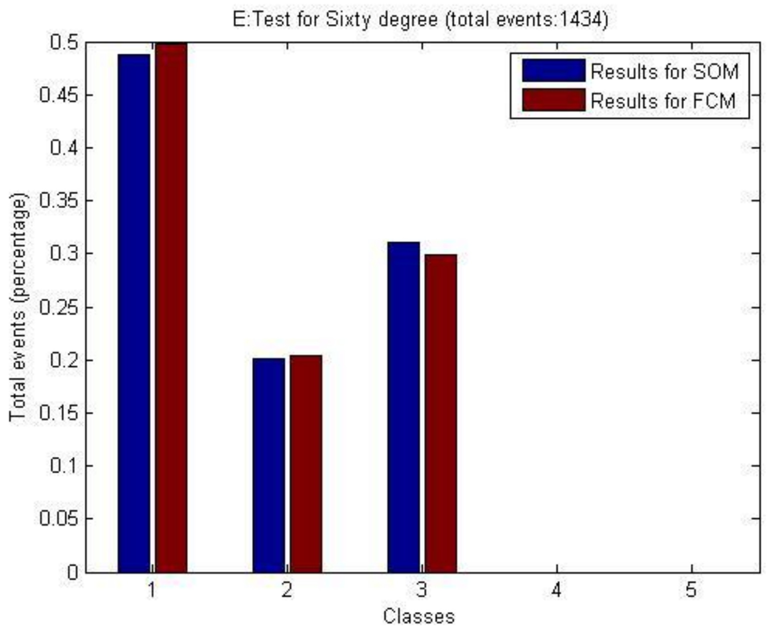


Figure 4-24: Normalized events per class for 60° Glass/Polyester

Figure 4-24 shows a normalized events bar chart for 60° fiber orientation of Glass/Polyester for both algorithms. It is noticeable that for this fiber configuration there are no events that could be related to either Class 4 or Class 5. Class 1 does not show any obvious pattern in terms of the change of its size as the fiber orientation turn around away from the direction parallel to the load.

4.4.6 75° Fiber orientation

Figure 4-25 shows the SOM output neurons for 75° Glass/Polyester along with the circled clusters obtained from these neurons. Figure 4-26 plots three cluster centers which were represented by cluster centers after FCM converged. As described in the case of 60° fiber orientation, Class 3 has appeared now and the number of events associated with it increases as the fiber orientation rotates away from the direction parallel to the load for the 60°, 75° and 90° tests. The number in Class 1 appears to increase in the first few configurations (between 0° and 60°) after this fiber configuration as we will see in the next orientation.

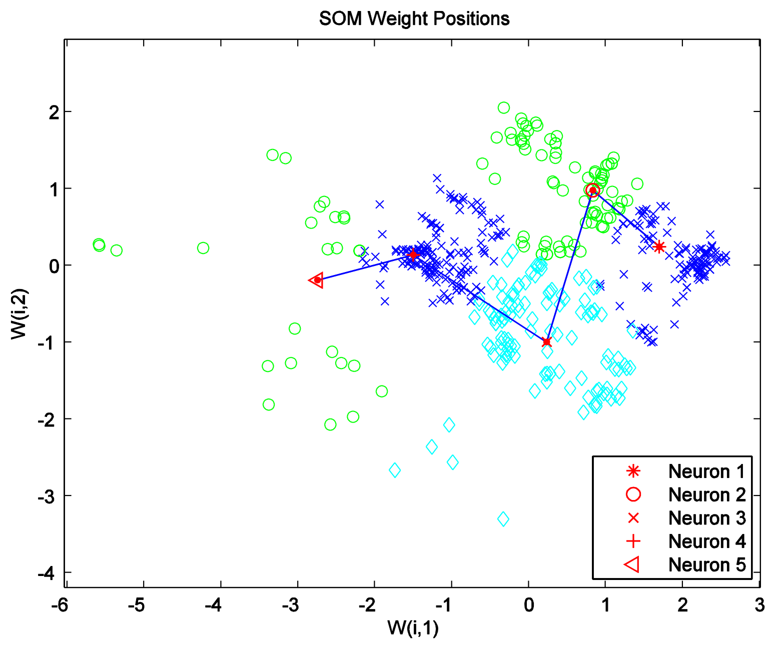


Figure 4-25: SOM output for 75° Glass/Polyester

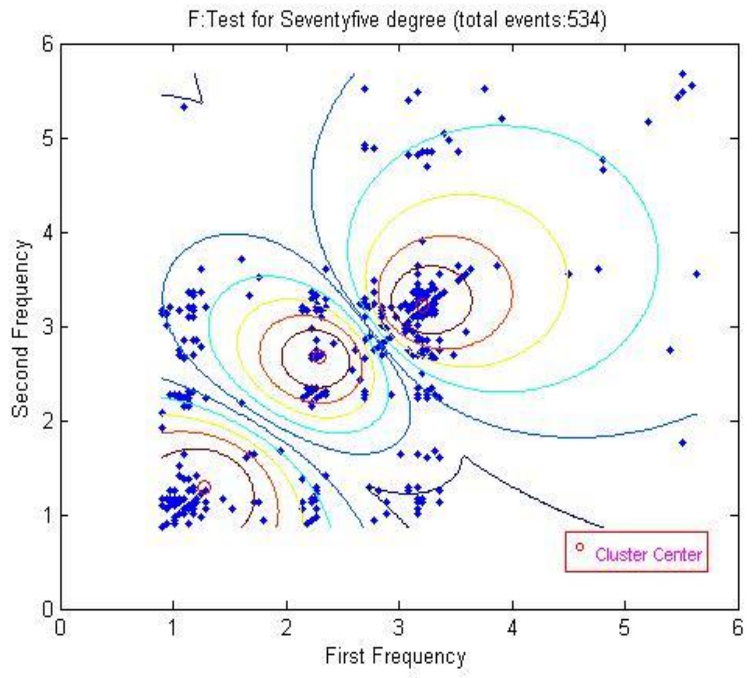


Figure 4-26: FCM output for 75° Glass/Polyester

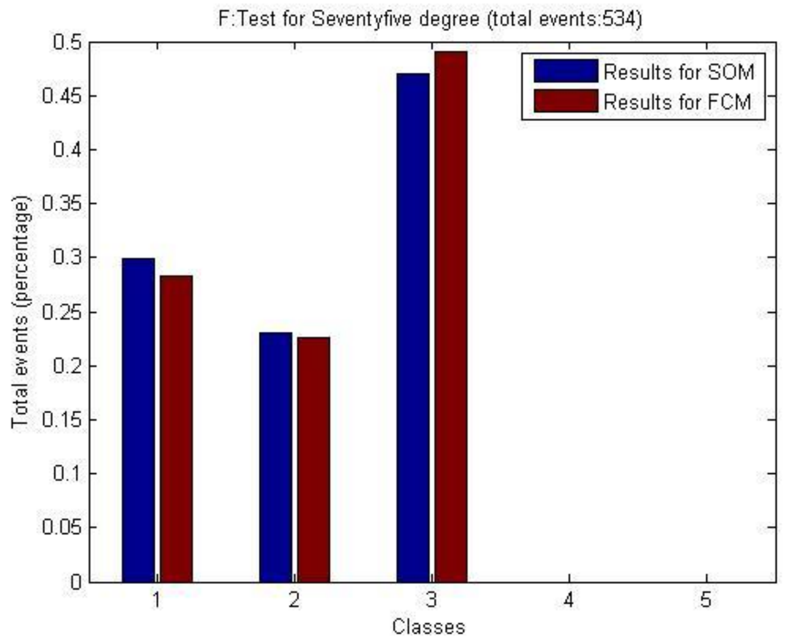


Figure 4-27: Normalized events per class for 75° Glass/polyester

Figure 4-27 shows the class normalized population for the 75° Glass/Polyester. Class 3 continues presenting an increase in the number of events compared to previous fiber configurations. Class 4 and Class 5 remain empty and the number in Class 1 continues to decrease as the fibers continuous to rotate away from the direction parallel to the load.

4.4.7 90° Fiber orientation

Figure 4-28 shows the SOM output neurons for the 90° fiber orientated Glass/Polyester. Class 1 is formed by neuron 5, Class 2 is populated by neurons 4 and 3, and finally Class 3 embraces neurons 2 and 1. FCM also converges to three clusters, after 30 iterations, and all the cluster centers are less than 4.5 kHz. This last configuration with the fiber in a direction perpendicular to the direction of the load (90°) is another special case in this present analysis. This is because in this configuration we will obtain an entire collection of events with characteristics different to those found in the test where the fiber are parallel (0°) to the direction of the load, as shown in Figure 4-12. The tests with steps in between will present a combination of events that gradually drift from those existing at 0°, at one end of the spectrum to 90°, the other end.

The classification of events across the entire set of seven tests suggests the evolutions of these different failure mechanisms that appear in more or less quantity over different fiber configurations.

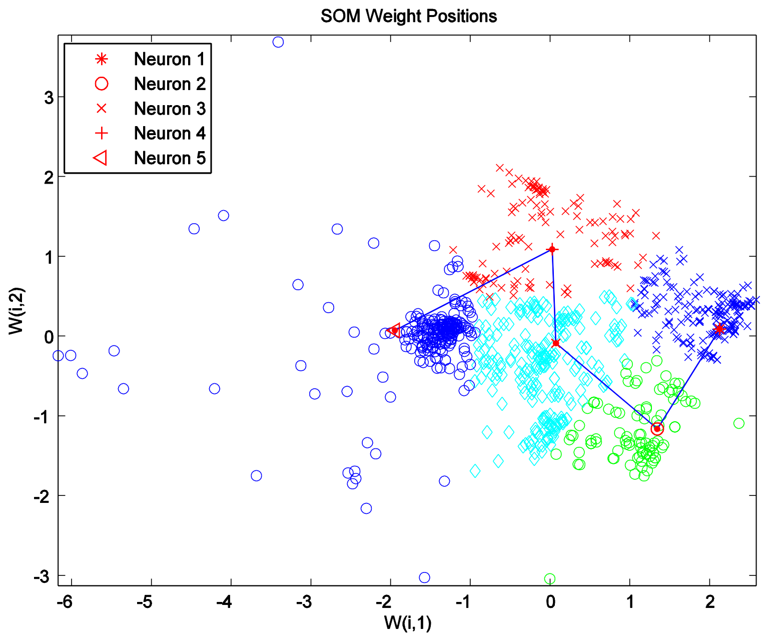


Figure 4-28: SOM output for 90° Glass/Polyester

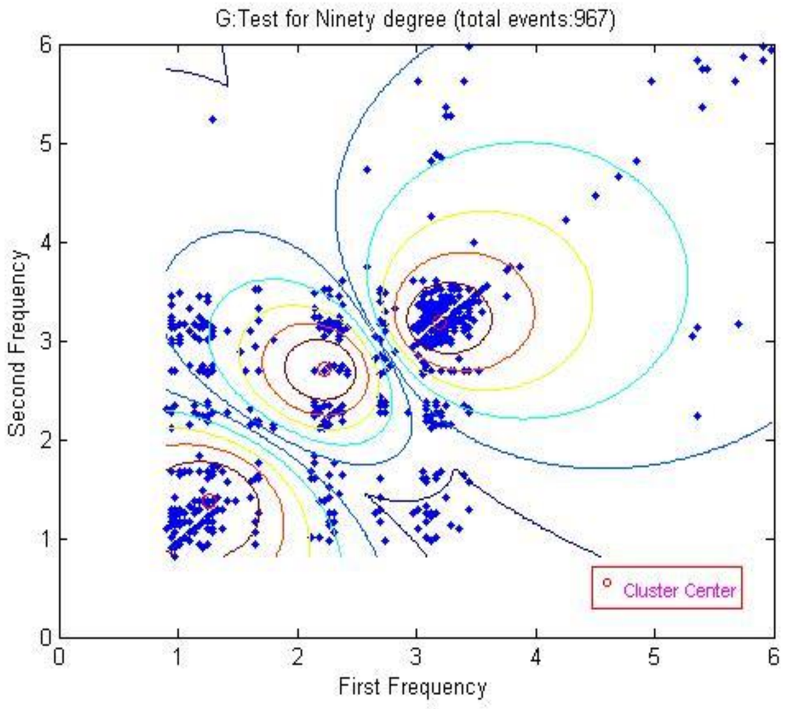


Figure 4-29: FCM output for 90° Glass/Polyester

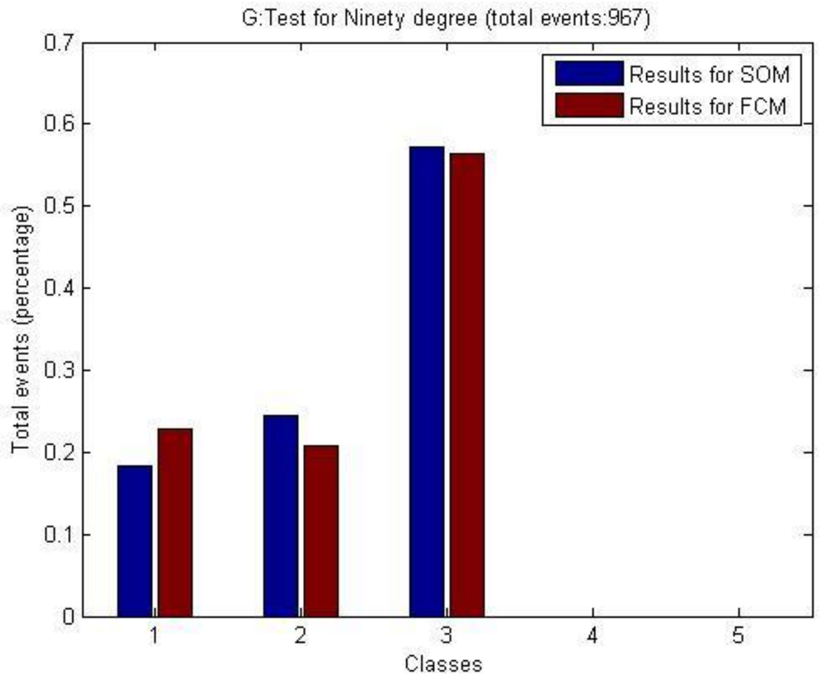


Figure 4-30: Normalized events per class for 90° Glass/Polyester

Bar graph Figure 4-30 shows the normalized event distribution for the classes found in both algorithms for 90° Glass/Polyester. Notice that Class 3 presents the largest contribution to the total number of events in the test with nearly 60%, which can be found in both the case of SOM and FCM.

Also, it can be seen in Figure 4-30 that Classes 4 and 5 are not present; Class 2 and Class 1 both contribute around 20% of the total number of events.

4.5 Discussion and Conclusions

The overall transition in both SOM and FCM of the relative contribution per class for each test can be seen in Figure 4-31. Class 2 shows a progressive decrease in its contribution to the total number of events. Class 3 on the other hand appears from the 60° fiber orientation and increases towards the transverse orientation.

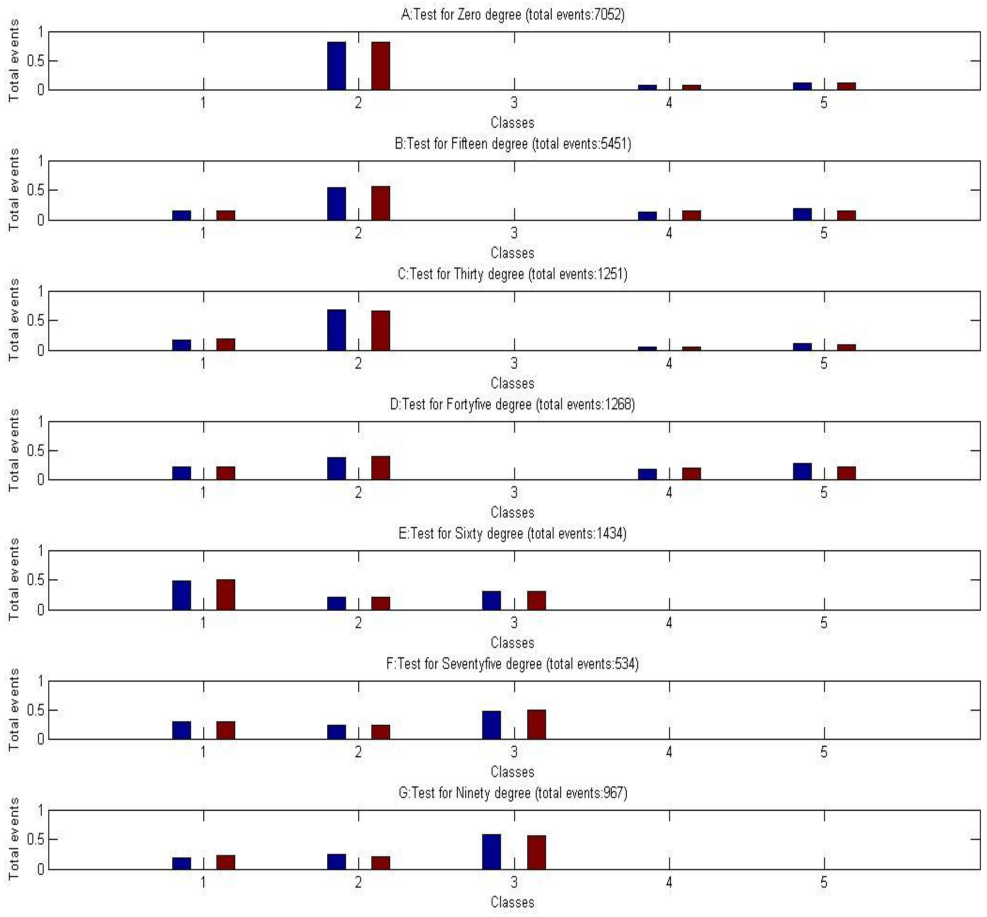


Figure 4-31: Comparison of results for the 7 tests

It can also be seen in Figure 4-31 that Classes 4 and 5 exist only for configurations below 45°; whereas they are not present for 60°, 75° and 90°. Class 1 appears in all the tests with a variable contribution to the total events except for the 0° configuration where it does not appear at all.

In addition to the bar chart which presents a straightforward comparison, we also made a comparison between the two methods for all the seven tests.

Table 4-1 shows the conclusions of these two techniques from which we can determine the consistency of these two results. Take the 0° test as an example, it indicates that 81.13% of the events had been classified as class 2, 6.14% of the events as class 4 and 12.73% as class 5 by SOM while FCM had labeled 81.69% of the events as class 2, 6.48% of the events as class 4 and 11.83% as class 5. In addition to that, another parameter 'similarity' was also introduced to evaluate the consistency of these two results. Similarity is defined as the ratio of the number of events which were classified as the same class by both SOM and FCM to the number of all the events. Take 0° test as an example again, we can conclude that 92.24% of all of the events (that is 6505 events) had been assigned the same classes. It should be noted that the results generated by SOM and FCM separately bear a high similarity (from 88.2% to 98.9%) and consistency which again justified our conclusion.

Table 4-1: Similarity Comparison of Two Results by FCM and SOM

Test	0°				15°			
Classes	2	4	5		1	2	4	5
SOM	81.13%	6.14%	12.73%		14.49%	54.65%	12.09%	18.77%
FCM	81.69%	6.48%	11.83%		14.09%	55.55%	14.77%	15.59%
Similarities	0.9224				0.8914			
Test	30°				45°			
Classes	1	2	4	5	1	2	4	5
SOM	18.07%	66.19%	4.40%	11.35%	17.19%	40.69%	23.34%	18.77%

FCM	18.55%	66.59%	5.28%	9.59%	18.22%	39.98%	20.35%	21.45%
Similarities	0.9888				0.9450			
Test	60°				75°			
Classes	1	2	3		1	2	3	
SOM	50.63%	20.71%	28.66%		27.90%	21.54%	50.56%	
FCM	49.72%	20.42%	29.85%		28.28%	22.66%	49.06%	
Similarities	0.9616				0.9172			
Test	90°							
Classes	1	2	3					
SOM	20.17%	25.96%	53.88%					
FCM	20.79%	22.75%	56.46%					
Similarities	0.8821							

Carlos Jimenez, who did these experiments and collected the dataset, also conducted an analysis from a material property perspective using scanning electron microscope (SEM) images in his thesis. Figure 4-32 and Figure 4-33 shows the SEM images of some of the experiments. After conducting research on the physical evidence seen on the SEM images for all the seven tests, he concluded that Class 1, Class 2 and Class 3 can be related to matrix cracking, fiber/matrix debonding and fiber/matrix pull out respectively, and Class 4 and 5 can be related to fiber and matrix breakage (Jimenez, 2007).

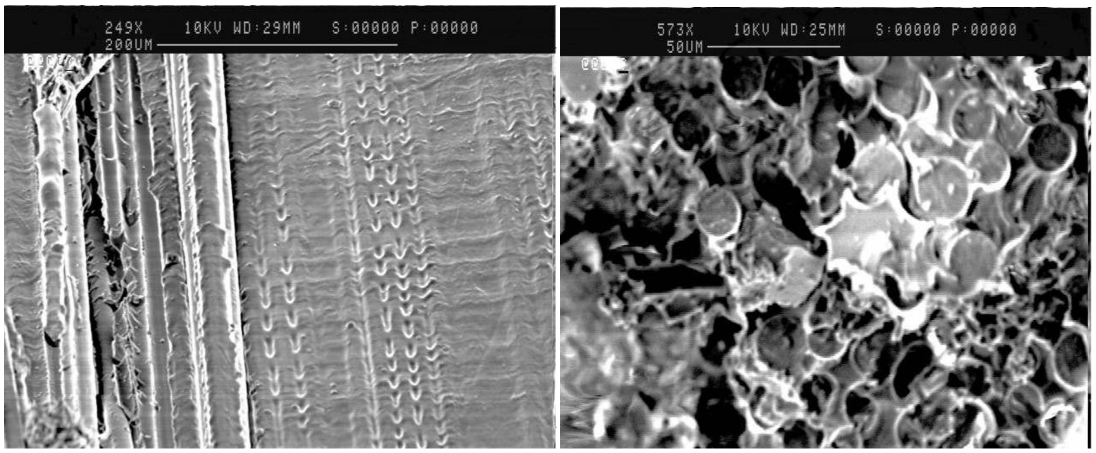

249X 10KV WD:29MM S:00000 P:00000
200UM
573X 10KV WD:25MM S:00000 P:00000
50UM

Figure 4-32: SEM image of the 0° Glass/Polyester test (left: from the upper face of, right: from side of the failure zone)(Jimenez, 2007)

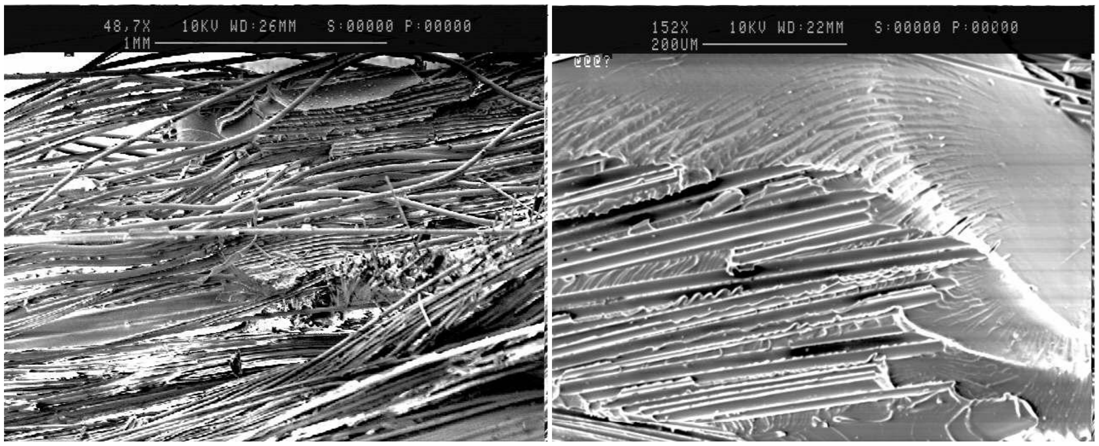

48.7X 10KV WD:26MM S:00000 P:00000
1MM
152X 10KV WD:22MM S:00000 P:00000
200UM

Figure 4-33: SEM image from the failure zone of Glass/Polyester test (left: 75°, right: 60°)(Jimenez, 2007)

In order to verify the correlation between classes and micro-mechanical failure events described in the previous seven tests, a test with 0°/90° Glass/Polyester was carried out. This type of configuration is known to present all the types of events described previously, fiber breakage, fiber/matrix debonding, fiber/matrix pull out and matrix cracking.

Figure 4-34 shows the SOM output neurons for the 0°/90° with each class denoted by each neuron. FCM converges after 46 iterations. Figure 4-35 shows all five clusters in the map which were all represented by a cluster center. From this figure we can see that the cluster which is around 2.5 kHz dominates all the classes, although each cluster comprises some events.

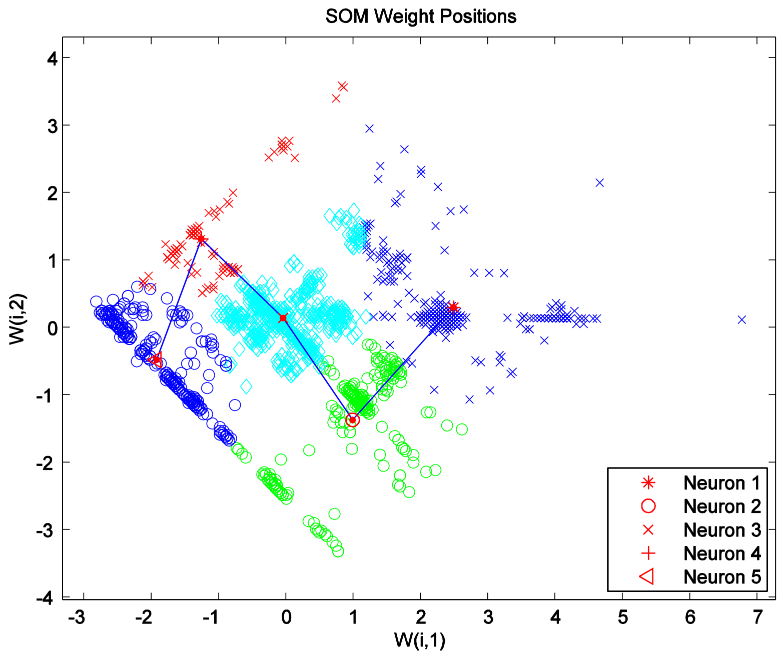


Figure 4-34: SOM output for 0/90° Glass/Polyester

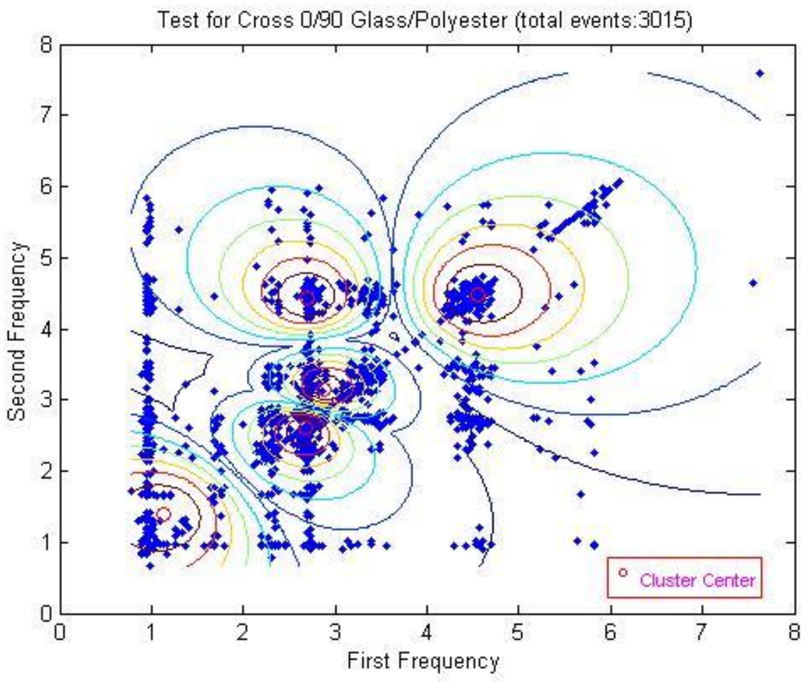


Figure 4-35: FCM output for 0/90° Glass/Polyester

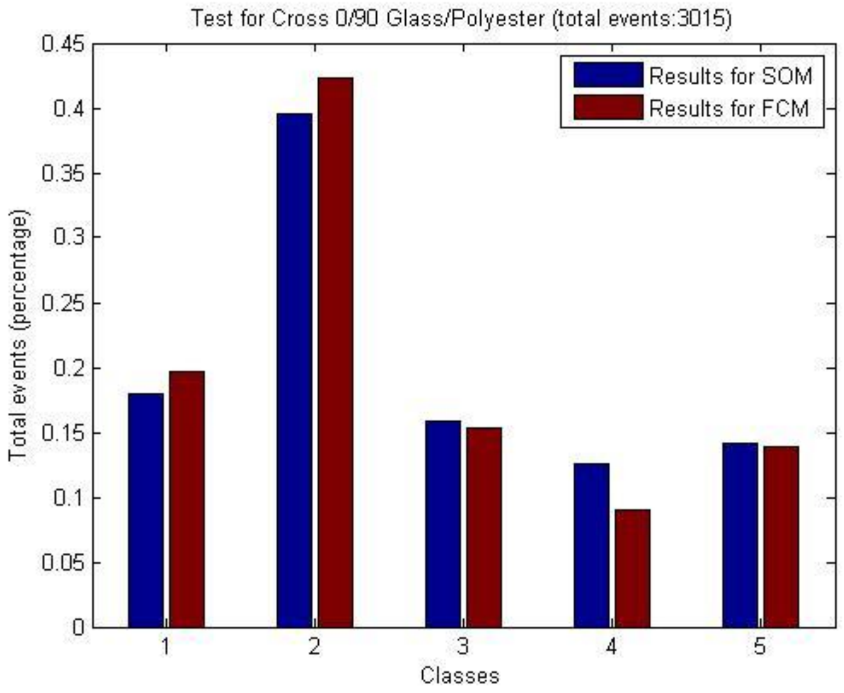


Figure 4-36: Normalized events per class for 0/90° Glass/Polyester

The bar chart (Figure 4-36) showed all 5 Classes which are just slightly different for SOM and FCM. This agrees with the hypothesis proposed in the present document since the 0°/90° configuration being a combination of the two configurations, it is expected to present micro-mechanical failure events (and their corresponding AE events) from the two fiber configurations.

Conclusions can be drawn as follows:

Frequency based analysis from AE technologies can be used to correlate micro-mechanical failure events in composite materials to their corresponding AE signature.

Two intelligent system techniques, SOM and FCM, have been used as reliable classification and clustering tools for AE events recorded from damage in composite materials.

From the plots obtained from the SOM and FCM classification of AE events, Class 1 appears in all seven tests except 0° orientation while Class 2 appear in all seven tests as well and dominates in 0° and 15° orientation. Class 4 and Class 5 didn't appear in 60°, 75° and 90° configurations, where, in 60° orientation, Class 1 dominates and in 75° and 90° orientation, Class 3 dominates.

4.6 Classifier design using Optimized Weighted Fuzzy Decision Tree

After the data was clustered, it was concluded that those acoustic events would fall into five categories. It would be helpful, if a classifier can be devised to detect what kind of failure it is after the signal is recorded. This classifier was designed using a FDT technique.

4.6.1 Fuzzy decision tree growing

Following the procedure of growing a tree in section 3.2.2, the following tree was constructed:

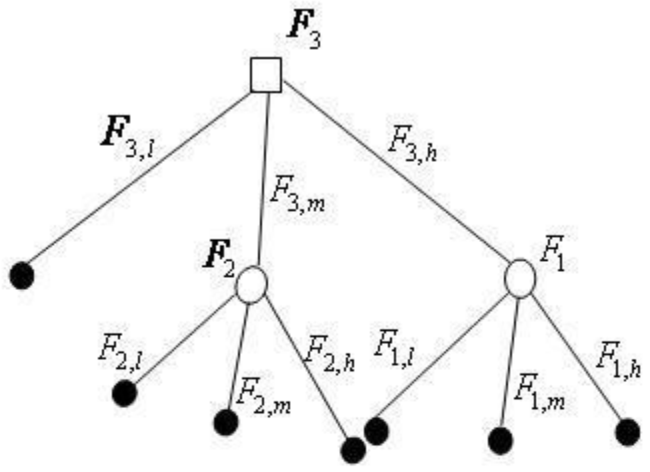


Figure 4-37: Architecture of FDT for Zero Degree Dataset

Figure 4-37 shows a fuzzy decision tree constructed using the 0° Fiber orientation test data. This dataset consists of 7052 samples and each of these samples is denoted by 4 features and classified to three classes. 50% of the samples were randomly selected to grow the FDT.

After the construction process, the FDT consists of 3 internal nodes and 7 leaf nodes. Each leaf node represents a rule and in the following reasoning mechanism, these 7 leaf nodes will be considered as 7 reasoning rules.

4.6.2 Optimized weighted fuzzy decision tree

4.6.2.1 Fuzzify input

The features of each instance (acoustic event) will be fuzzified into three membership values according to three linguistic terms. As introduced in chapter 3.3.1, FCM was used to cluster the feature values into three clusters. Three cluster centers and the maximum and minimum values along each feature will be used as parameters to fuzzify the features.

Let $F_{j,high}$, $F_{j,med}$ and $F_{j,low}$ be the three sorted cluster centers along every feature. The cluster centers along these four features are listed as:

$$\begin{bmatrix} F_{1,high} & F_{2,high} & F_{3,high} & F_{4,high} \\ F_{1,med} & F_{2,med} & F_{3,med} & F_{4,med} \\ F_{1,low} & F_{2,low} & F_{3,low} & F_{4,low} \end{bmatrix} =$$

$$\begin{bmatrix} 112.0972 & 5.3204\mathrm{e}+05 & 5.1030\mathrm{e}+05 & 4.9052\mathrm{e}+05 \\ 107.3586 & 2.8033\mathrm{e}+05 & 3.7376\mathrm{e}+05 & 4.8838\mathrm{e}+05 \\ 80.0960 & 2.6012\mathrm{e}+05 & 2.6230\mathrm{e}+05 & 2.5675\mathrm{e}+05 \end{bmatrix}$$

The fuzzified membership values of an input pattern are formulated as Equations 10, 11 and 12 in chapter three.

4.6.2.2 Mapping weighted fuzzy decision tree to neural networks and the learning process

In order to optimize both the local and global weights associated with the rules, these rules were mapped to a NN structure so that the weights can be updated and optimized.

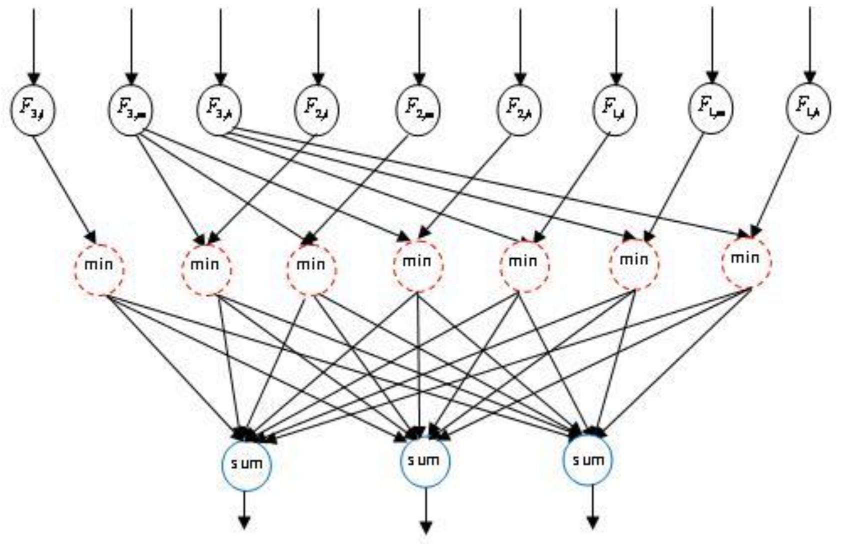


Figure 4-38: Neural Network Structure of Mapped FPR

Figure 4-38 shows the structure of the converted NN. This structure is based on the FDT showed in Figure 4-37. The input layer comes from the splitting features and the hidden layer denotes the leaf nodes or paths.

After the structure is established, the network is trained through the learning process illustrated in chapter 3.3.5.

4.6.3 Reasoning mechanism

First, the certainty factor of each leaf node was defined. Since each node will classify the input instance into different classes with a different certainty, the certainty of each node can be calculated as (taking node four in Figure 4-37 for example):

$$\alpha_4 = \left\{ \frac{\left|D_4^{C_1}\right|}{D_4}, \frac{\left|D_4^{C_2}\right|}{D_4}, \frac{\left|D_4^{C_3}\right|}{D_4} \right\}$$

The certainty factors associated with the 7 leaf nodes for the classes are:

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} \\ \alpha_{51} & \alpha_{52} & \alpha_{53} \\ \alpha_{61} & \alpha_{62} & \alpha_{63} \\ \alpha_{71} & \alpha_{72} & \alpha_{73} \end{bmatrix} = \begin{bmatrix} 0.0377 & 0.9605 & 0.0019 \\ 0.0009 & 0.9991 & 0 \\ 0.0015 & 0.9931 & 0.0054 \\ 0.1879 & 0.6190 & 0.1932 \\ 0.7283 & 0.2717 & 0 \\ 0.4774 & 0.4182 & 0.1044 \\ 0.0755 & 0.0445 & 0.8800 \end{bmatrix}$$

The constructed FDT shares the membership values along the paths leading to each leaf node from root node. In Figure 4-37, $path_4$ has two segments $segm_{3,med}$ and $segm_{2,high}$, so the membership degree of $path_4$ can be calculated as:

$$\mu_{path_4}^i = \mu_{med}\left(F_3^i\right) * \mu_{high}\left(F_2^i\right) = [F_{3,m}^i, F_{2,h}^i]$$

Figure 4-38 shows the basic structure of the FDT for the zero degree data, in which seven leaf nodes are combined to produce a final conclusion. In the reasoning mechanism of OWFDT, the certainty factor and the membership value of that path (leaf node) as well as the global weights are incorporated together to infer the classification result.

4.6.4 Simulation Results

Seven datasets from seven fiber orientation experiments used in section 4.4 were tested using this classifier. In each test, 50% of the dataset was randomly chosen to grow, update and optimize the weights of the WFDT and the other 50% was used as a test dataset to test the accuracy of the OWFDT. The simulation was repeated five times for every dataset. The accuracy was the average of the five. In order to compare its performance, the conventional FDT and Weighted FDT with no optimization were also introduced as benchmark classifiers.

Table 4-2 summarizes the mean test accuracies for the datasets employed. The classification accuracy was calculated as: $accuracy = \frac{n_r}{n} * 100\%$, where n is the total number of test instances and n_r is the number of instances which had been correctly classified.

Degree (event No., class No., Attribute No.)	Fuzzy ID3	WFDT	OWFDT
0^0 (7052,3,4)	94.46%	94.55%	95.12%
15^0 (5451,3,4)	90.53%	92.29%	93.61%
30^0 (1251,4,4)	89.47%	93.30%	95.22%
45^0 (1268,4,4)	83.98%	82.31%	84.83%

60^0 (1434,3,4)	87.01%	88.83%	90.50%
75^0 (534,3,4)	80.60%	92.86%	95.11%
80^0 (967,3,4)	86.31%	92.53%	95.49%

Table 4-2: Testing Accuracy for the material datasets

The table shows that the proposed optimization methods can improve the WFDT's accuracy for these datasets. The improvement is generally 2 to 3 percent higher than straight WFDT and typically 3 to 9 percent higher than Fuzzy ID3 in terms of accuracy.

4.7 Conclusion

In this chapter, we have successfully used some ISTs to analyze the data generated by the failure mechanism when composite materials are subject to load. The results have shown that there are different failure mechanisms in different applications. In order to detect those potential damages, a classifier can be designed to cluster and classify them. Our research has shown that OWFDT can be used as the technique to classify failure mechanisms with accuracy better than traditional FDT and weighted FDT.

Possible areas of future work include:

Following the methodology explained in the current chapter, there is scope for locating the damage source for large size components, i.e. plates or actual components.

Based on the current results there would be scope for analyzing the behavior of sandwich structures under different types of loads, tensile, compression and flexural.

A more complex and systematic classifier should be designed which incorporates recoding raw data and preprocessing them and ultimately predicting the failure mechanisms.

In the next chapter, OWFDT will be used to design a classifier to classify eye bacteria species for speed diagnosis of eye disease.

Reference

Ativitavas, N., Pothisiri, T., & Fowler, T. J. (2006). *Identification of fiber-reinforced plastic failure mechanisms from acoustic emission data using neural networks* (Vol. 40). London, ROYAUME-UNI: Sage.

Babuska, R. (1998). *Fuzzy Modeling for Control*: Kluwer Academic Publishers.

Balasko, B., Abonyi, J., & Feil, B. (2004). *Fuzzy Clustering and Data Analysis Toolbox*: Department of Process Engineering University of Veszprem.

Bohse, J. (2000). Acoustic emission characteristics of micro-failure processes in polymer blends and composites. *Composites Science and Technology, 60*(8), 1213-1226.

de Groot, P. J., Wijnen, P. A. M., & Janssen, R. B. F. (1995). Real-time frequency determination of acoustic emission for different fracture mechanisms in carbon/epoxy composites. *Composites Science and Technology, 55*(4), 405-412.

Eduard, L., Hines, E. L., Gardner, J. W., & Franco, S. (1999). Non-destructive banana ripeness determination using a neural network-based electronic nose. *Measurement Science and Technology, 10*(6), 538.

El Kadi, H. (2006). Modeling the mechanical behavior of fiber-reinforced polymeric composite materials using artificial neural networks--A review. *Composite Structures, 73*(1), 1-23.

Giordano, M., Calabrò, A., Esposito, C., Salucci, C., & Nicolais, L. (1999). Analysis of acoustic emission signals resulting from fiber breakage in single fiber composites. *Polymer Composites, 20*(6), 758-770.

Godin, N., Huguet, S., & Gaertner, R. (2005). Integration of the Kohonen's self-organising map and k-means algorithm for the segmentation of the AE data collected during tensile tests on cross-ply composites. *NDT & E International, 38*(4), 299-309.

Godin, N., Huguet, S., Gaertner, R., & Salmon, L. (2004). Clustering of acoustic emission signals collected during tensile tests on unidirectional glass/polyester composite using supervised and unsupervised classifiers. *NDT & E International, 37*(4), 253-264.

Huguet, S., Godin, N., Gaertner, R., Salmon, L., & Villard, D. (2002). Use of acoustic emission to identify damage modes in glass fibre reinforced polyester. *Composites Science and Technology, 62*(10-11), 1433-1444.

Jimenez, C. R. R. (2007). *Analysis of micro mechanical events in fiber reinforced composite materials by means of acoustic emissions.* University of Warwick.

Johnson, M. (2002). Waceform based clustering and classification of AE transients in composite laminates using principal component analysis. . *NDT & E Int, 35*(7), 367-376.

Kohonen, T. (2001). *Self-organizing Maps*. Berlin ; New York Springer.

Lampinen, T., Laurikkala, M., Koivisto, H., & Honkanen, T. (2005). Profiling Network Applications with Fuzzy C-means and Self-Organizing Maps (pp. 15-27).

Papadakis, N. (2004). *Relation comparison methodologies of the primary and secondary frequency components of acoustic events obtained from thermoplastic composites laminates under tensile stress*. Paper presented at the 11th European Conference on Composite Materials.

Pollock, A. (1989). *Acoustic emission inspection, Technical Report* Physical Acoustics Corporation.

Qi, G. (2000). Wavelet-based AE characterization of composite materials. *NDT & E International, 33*(3), 133-144.

Ramirez-Jimenez, C. R., Papadakis, N., Reynolds, N., Gan, T. H., Purnell, P., & Pharaoh, M. (2004). Identification of failure modes in glass/polypropylene composites by means of the primary frequency content of the acoustic emission event. *Composites Science and Technology, 64*(12), 1819-1827.

Sause, M., & Horn, S. (2010). Simulation of Acoustic Emission in Planar Carbon Fiber Reinforced Plastic Specimens. *Journal of Nondestructive Evaluation, 29*(2), 123-142.

Chapter V: Enhancing the Classification of Eye Bacteria Using Bagging to Multilayer Perceptron and Decision Tree

After designing the classifier using OWFDT to predict the possible failure mechanism in composite material, this chapter will design a classifier to classify the eye bacteria. In this chapter the aim is to classify different kinds of eye bacteria after the data were collected using the Electronic Nose. First the Multi-layer perceptron (MLP) and Decision Tree (DT) are introduced as the algorithms and the base classifiers. This is followed by the introduction of the bagging technique to both algorithms. It shows that the bagging technique can improve the accuracy of both MLP and DT. Finally a classifier is designed using OWFDT to predict the possible bacteria classes for the diagnosis of eye disease. The classifier is able to extract rules and improve the accuracy of standard FDT and weighted FDT.

5.1 Introduction

The eye is one of the main human organs which links to the inner body and is continuously exposed to a harsh outside environment where it is continually in contact with pathogenic airborne organisms. Although the eyelid may help to protect the eye, the warm, moist, enclosed environment between the conjunctiva and the eyelid also enables contaminating bacteria to establish an infection. The number of organisms responsible for infection of eye is relatively small, but they can proliferate rapidly and cause serious and irreversible damage to the eyes, which makes rapid diagnose essential (J. W. Gardner, Boilot, & Hines, 2005). Usually this kind of diagnosis is based on the study of symptoms, such as changes in bodily appearance, feel or functions etc (Dutta, Hines, Gardner, & Boilot, 2002). Since different diseases produce distinctive smells sometimes specific characteristic odors, smelling the bacteria becomes a significant part of diagnosis.

Fortunately, after 20 years of development, electronic noses (ENs) have been very successful in terms of health and safety issues and the task of diagnosing medical conditions through analyzing odors. An electronic nose is an instrument, which comprises of an array of electro chemical sensors with partial specificity and an appropriate pattern recognition system, capable of recognizing simple or complex odors (Julian W. Gardner & Bartlett, 1994). Many of the initial applications of ENs were concerned with the detection and classification of bacteria, which suggests EN can be used for medical diagnosis purposes in the detection of bacteria associated with eye diseases (Dutta, Hines, Gardner, Udrea, & Boilot, 2003; Llobet, Hines, Gardner, & Franco, 2004).

In this chapter, we focus on the use of the Cyranose 320 (Cyrano Sciences Inc.) for the detection of bacteria responsible for eye infections using pure laboratory cultures. The project represented a joint collaboration between researchers from the University of Warwick and Doctors from Heartlands Hospital and Micropathology Ltd. (a medical laboratory specializing in the detection of these pathogens) (Boilot, et al., 2002).

5.2 Experiment and Data Collection

5.2.1 Instrumentation

Although the number of organisms responsible for infection of the eye is relatively small, the damage caused may be irreversible and this has made rapid diagnosis of organisms causing the damage essential. Techniques such as a neural network based ENs have been used to detect and classify odorous volatile components and diagnose the nature of the infection as quickly as possible. The ENs, which are able to mimic the human sense of smell have been the subject of much research at the University of Warwick over the past 20 years or so (Dutta, et al., 2002).

The EN used here was Cyrano Sciences' Cyranose 320, currently used in diverse industries including petrochemical, chemical, medical, food, packaging etc. The diagnosis of disease often relies on invasive testing methods, subjecting patients to unpleasant procedures. A tool such as the Cyranose 320 will enable physicians and dentists to provide immediate, accurate diagnosis of chemical components and microorganisms in breath, wounds and bodily fluid.

The Cyranose 320 is a portable system which consists of 32 individual polymer sensors blended with carbon black composite, configured as an array. It works by exposing this array of polymer composite sensors to the chemical compounds in a vapor state. When the sensors come in contact with the vapor, the polymer expands like a sponge, changing the conductivity of the carbon pathways and causing a resistance change of the polymer composites. The change in resistance is measured as the sensor signal and captured as the digital pattern representing the test smell (J. W. Gardner, et al., 2005). And from that measurement, a distinct response signature for each vapor was recorded.

5.2.2 Experimental Materials

The eye bacteria experiments were conducted under typical lab conditions. The most common bacteria responsible for eye infection is conjunctivitis and organisms such as Staphylococcus aureus (sar), Haemophilus influenzae (hai), Streptococcus pneumonia (stp), Escherichia coli(eco), Pseudomonas aeruginosa (psa) and Moraxella catarrhalis (moc). All bacteria strains were grown on blood or lyse blood agar in standard petri dishes at 37° C in a humidified atmosphere of 5% CO_2 in air. After overnight culturing, the bacteria were suspended in a sterile saline solution (0.15 M NaCl) to concentrations of approximately $10^8, 10^5, 10^4$ colony forming units (cfu)/ml. A ten-fold dilution series of bacteria in saline was prepared and these three dilutions were sniffed using the EN. The numbers of viable bacteria present were confirmed by plating out a small aliquot of the diluted samples and counting the resultant colonies after overnight incubation (Dutta, et al., 2002).

5.3 Data Collection and Processing

When the sensors are exposed to vapors or aromatic volatile compounds the sensors swell, changing the conductivity of the carbon pathways and causing an increase in the resistance value that is monitored as the sensor signal. The resistance changes across the array are captured as a digital pattern that is representative of the test smell. The sensor technology yields a distinct response signature for each vapor regardless of its complexity; the overall response to a particular sample produces a 'smell print' specific to a stimulus. For each solution of our eye bacteria tests, the datalogger was introduced manually into a sterile glass vial containing a fixed volume of bacteria in suspension (4 ml) to collect samples. The operation was repeated ten times for each one of the three dilutions for each one of the six bacteria species, to give a total of 180 readings. These data were gathered over the course of a whole week. After that, the key features were extracted from the raw data files which were saved during the data collection. All the data was normalized using a fractional difference model:$dR = (R - R_0)/R_0$ where R is the response of the system to the sample gas and R_0 is the baseline reading with the reference gas being the ambient room air. The complete bacteria data set was then normalized by dividing each *dR* (values between [1.1, 18.2] $m\Omega$) by the maximum value for each sensor, in order to set the range of each sensor parameter to [0, 1].

Figure 5-1 shows the statistical values of the dataset. It can be seen that the maximum value for each sensor varies within a small range. This is because all the signals were produced by the same type of carbon black polymer composite resistors. But both the minimum values and the mean values presented a stronger variation, due to the fact that the EN sensors react differently to different odours. This feature helps in distinguishing odours using the EN data. It is noticeable that the standard deviations of sensors 8, 23, 24 and 32 are considerably larger than the average ones. These findings may indicate sensors that would appear in the optimal subset of sensors.

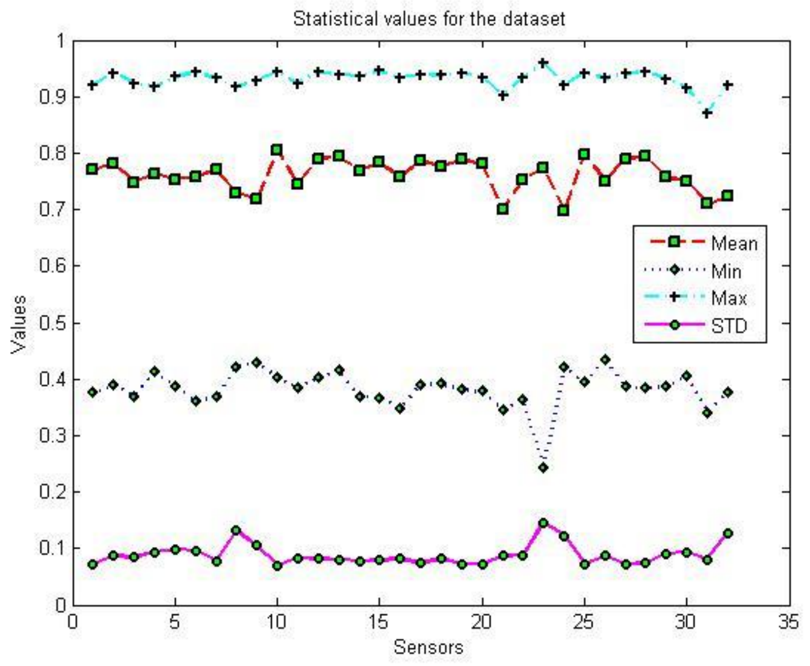


Figure 5-1: Statistical values for the dataset.

Principal Component Analysis (PCA) was conducted to visualize the data and explore the possible data clusters in order to analyze the patterns. The objective of the PCA exploration was to establish whether or not simple classes exist for the different bacteria species, and to see whether or not data clusters could be found before the pattern recognition stage. The PCA results are shown in Figure 5-2 in which three principal components were presented in a 3D space. These three components accounted for 97.37% of the variance and six categories appeared to be evident with little overlapping, so that the bacteria were completely separated in the principal component space. The PCA analysis also found that the variance of the first principal component is around 80% and that of the second principal component is around 16% and that of the third principal component is around 1.5%, showing a high variance and low cross-correlation between the sensors. It is worth noting that no conclusions can be drawn from this analysis and that PCA is only used for data visualization.

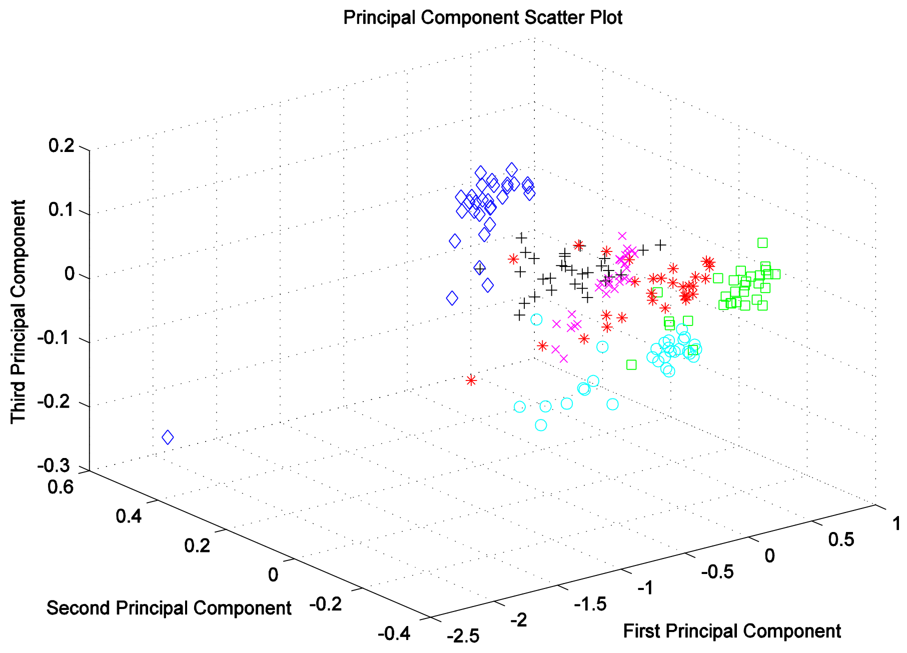


Figure 5-2: Principal component scatter plot with colored clusters

5.4 Introduction to the classification techniques

The goal of designing the pattern recognition classifiers when applied to EN data is typically to generate a class predictor for an unknown odor vector from a discrete set of previously learned patterns. The techniques we used here are artificial neural networks and Decision trees. But as a classifier, both ANN and DT endure unstable performance, since a small change in the learning set or the classification parameters may lead to a significant change of the results (Wendy & Angel, 2007). In order to combine classifiers and produce a stable performance, the bagging technique was introduced as a way of combining ensembles of classifiers.

5.4.1 General bootstrap methodology and bootstrap aggregation

The concept of the bootstrap methods was first proposed by Efron and Tibshirani (Efron & Tibshirani, 1993). There is no generic and consistent terminology in the literature indicating what techniques are considered to be a bootstrap method. Here, we use bootstrap to refer to Monte Carlo simulations that treat the original sample as the pseudo-population or as an estimate of the population(Mooney & Duval, 1993). Thus, we now resample from the original sample instead of sampling from the pseudo-population (Davison & Hinkley, 1997). In this section, we will discuss the general bootstrap methodology. The application of it to the MLP and decision tree will be discussed in section 5.5.

The bootstrap method is a method of resampling with replacement and generating the random samples from an underlying population. We obtain each bootstrap replica by randomly selecting *N* observations out of *N* with replacement, where *N* is the dataset size. Suppose we denote the original sample as $x = (x_1, \ldots, x_n)$. We can denote the new sample obtained from this method by $x^* = (x_1^*, \ldots, x_n^*)$. Since we sample with replacement from the original samples, there is a possibility that some points will appear more than once in x^* or not at all.

If $x = (x_1, x_2, x_3, x_4) = (a,\ b, c, d)$, the possible samples can be:

$$x^{*1} = (x_1^{*1}, x_3^{*1}, x_3^{*1}, x_2^{*1}) = (a, c, c, b)$$

or $$x^{*2} = (x_2^{*2}, x_4^{*2}, x_3^{*2}, x_1^{*2}) = (b, d, c, a)$$

Bagging, the acronym for "Bootstrap Aggregation", is a method for generating an aggregate classifier by using multiple versions of a classifier. Each version of the classifier is trained by using a set of bootstrap replica as the training dataset, which means each version of the training set is created by randomly selecting *n<=N* samples of a training dataset. And then each training dataset is used to train a classifier model, resulting in *n* classifiers. To classify a new observation, we find the class predicted by each of the bootstrap classifiers and assign the class label with the result that appears most often among the bootstrap classifiers or by the average of all the results of the classifiers (Jang, Sun, & Mizutani, 1997; Wendy & Angel, 2007).

Bagging works by reducing variance of an unbiased base classifier such as a Multilayer perceptron (MLP) or a decision tree (DT). In order to justify the use of bagging, the base classifier must be unstable; its configuration must vary significantly from one bootstrap replica to another.

5.5 Simulation Result

In this section we first propose the use of basic MLP and DT to classify Eye bacteria data. In each subsection, we introduce the bagging technique to enhance their performance.

5.5.1 Using MLP as base classifier

MLP is a forward network that consists of an input layer, one or more hidden layer and an output layer. The number of the input layer neurons is determined by the number of features of the input pattern. Here for the eye bacteria data, we have 32 sensors which subsequently generate a 32-dimension feature dataset. In addition to that, the number of hidden layers, the number of neurons in each hidden layer, the number of training epochs and the learning rate and the momentum are the parameters which may influence the network's performance. These parameters have to be carefully selected to guarantee the performance through cross validation. The most frequently employed training algorithm for MLP is Back Propagation (BP). Usually, BP is executed in two phases: forward phase and backward phase. The forward phase involves feeding an input pattern into the input layer and propagating the signal to the output layer to generate the predicted class which is then compared to the target output to compute the prediction error. In the backward phase, the error is backpropagated to adjust the weights between the hidden layer and the output layer and the weights between the hidden layer and the input layer. Two most popular training algorithms for feed-forward MLP are either back-propagation with gradient descent (GD) or Levenberg-Marquardt (LM). Here we choose both of these two as the back-propagation training algorithm.

For our eye bacteria experiment, the objective is to discriminate samples in terms of bacteria species. So we set the number of neurons in the input layer to be 32 and the number of neurons in the output layer to be 1. The whole dataset was divided into three subsets: training set, validation set and test set. After carrying out the test procedure, we found that a standard MLP cannot achieve a stable result which can show a good regression between the predicted response and the corresponding target.

In order to improve the accuracy, we introduced the bagging techniques to MLP. This technique consists of creating new training data sets to generate a base classifier by applying a learning algorithm. The final prediction is determined by the average predictions of each created classifier. In order to minimize the standard error, we decided to produce 100 replicas to establish 100 ensemble classifiers. The procedure was outlined below:

- Generate a bootstrap sample of size 180 by sampling from the original data set. 60% of the instances will be selected as the training dataset, 20% of the instances will be selected as the validation dataset, and the remaining 20% will be regarded as the test dataset.
- Construct and train a base MLP classifier with the bootstrap sample from step 1.
- Repeat step one and step two 100 times in order to yield 100 classifiers.
- Take the test dataset to each of the classifiers. Assign a classification result to the test instance using each of the 100 classifiers.

For each classifier, we randomly select 108 (60 percent of all the samples) instances as training dataset, 36 instances as validation dataset (20 percent of all the samples) and 36 instances (20 percent of all the samples) as test dataset. After repeating this process and training the classifiers 100 times, every instance will be assigned a certain number of class labels. The number of class labels will be the times the instance was selected in the test datasets. Thus we constructed a parameter called Mean Magnitude Error (MME) to evaluate the accuracy of bagging MLP technique. The MME was defined as:

$$MME = \frac{1}{n}\sqrt{\sum_{i=1}^{n}(O_i - T)^2}$$

where O_i is the predicted class label from a classifier and T is the targeted output for this instance, and n is the number of times this instance was selected by the ensemble classifiers. According to the definition of MME, a lower value of MME will suggest a better prediction result. Ideally an MME value of zero will indicate a perfect classification result. If the value of MME is less than a specified threshold, that instance can be regarded as being classified correctly (Braga, Oliveira, Ribeiro, & Meira, 2007). Figure 5-3 shows the MME values for all the instances generated by the ensemble.

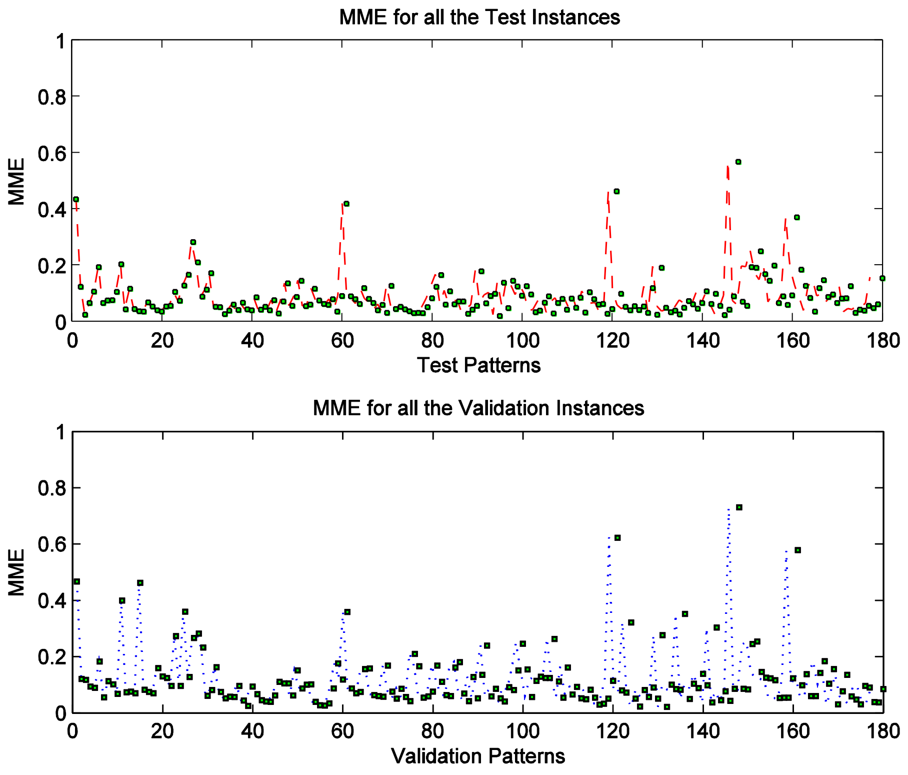


Figure 5-3: MME for the entire instance

Table 5-1 shows the relationship between the MME threshold and accuracy. It can be seen that the accuracy can be as high as 100% if the MME threshold was set to 0.6, an accuracy which cannot be achieved using basic MLP classifier. Figure 5-4 shows the linear regression relationship between the network outputs and the corresponding targets. The Regression values (the correlation coefficients between the network response and the target, with 1 meaning perfect correlation) have proved that the network outputs for both test data and validation data fit well with the targeted ones.

Table 5-1: MME vs. Accuracy

MME threshold	Accuracy for LM	Accuracy for GD
0.6	100.00%	98.89%
0.5	99.44%	98.33%
0.4	97.78%	95.00%
0.3	97.22%	87.22%
0.2	95.00%	65.00%

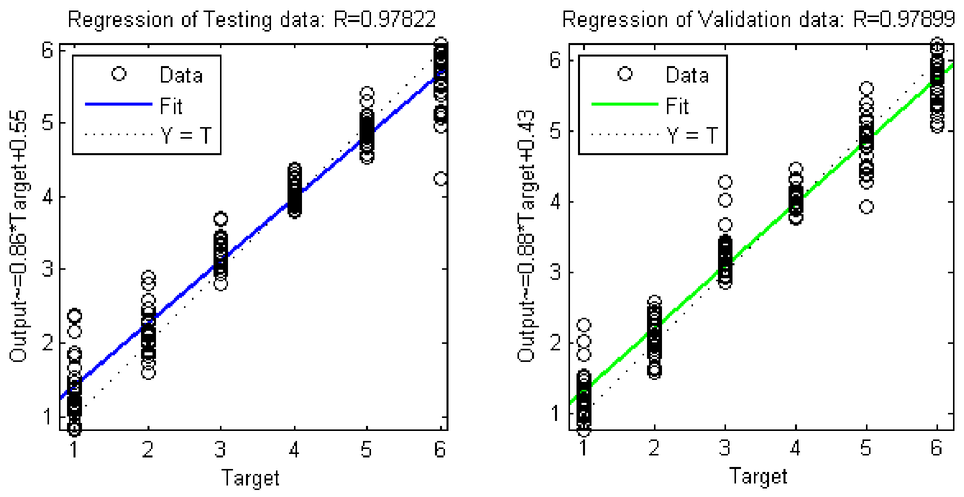


Figure 5-4: Regression of testing and validation data

5.5.2 Using Decision tree as the base classifier

The idea behind a classification tree is to split the high dimensional data into smaller and smaller partitions, such that the partitions become purer in terms of the class membership. It first grows an overly large tree using a criterion that generates optimal splits for the tree. Usually, these large trees fit the training data set very well, but they do not generalize well, so the rate at which we correctly classify new patterns is low (see chapter two).

The eye bacteria dataset was used to grow the tree. It consists of 13 leaf (terminal) nodes and 12 internal nodes. Each leaf node is a reasoning path or rule. Figure 5-5 shows a cluttered-looking tree using a series of rules such as "$X_{23} < 0.648033$" to classify each pattern into one of 13 terminal nodes. To determine the input pattern assignment for an observation, the decision tree starts at the top node and applies the rule. If the point satisfies the rule the tree takes the left path, and if not it takes the right path. Ultimately it reaches a terminal node that assigns an input pattern to one of the six categories.

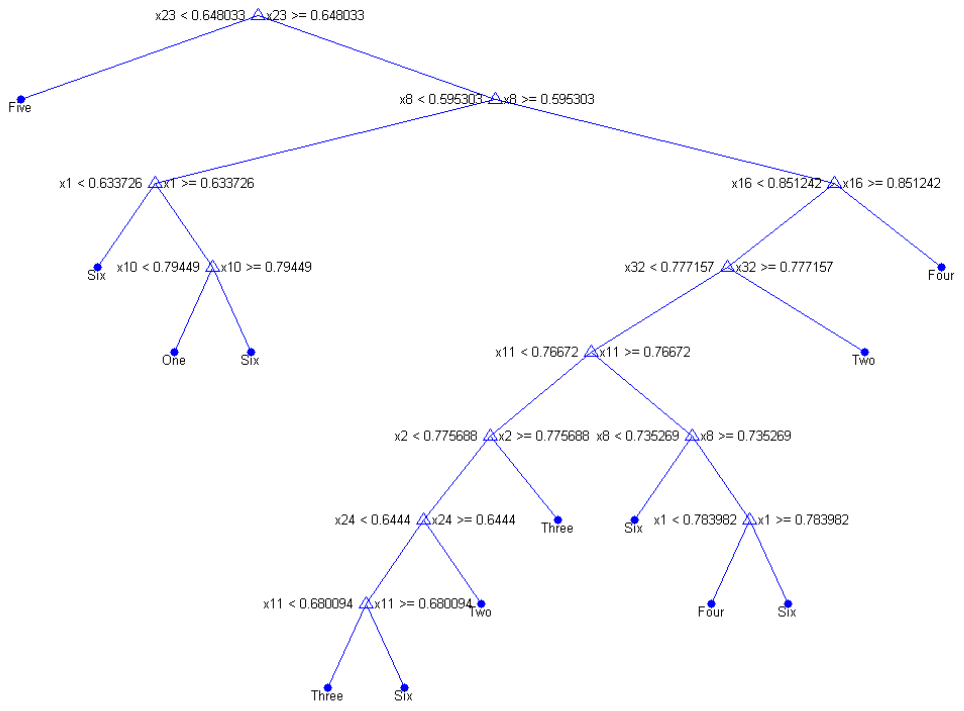


Figure 5-5: Classification tree structure before pruning

A large and complex tree over fits the training data and will not generalize well to new patterns. This will cause large generalization error. So the "true" error rate this large and complex classification tree would incur by using it to classify new data should be taken into account. But usually, a simpler tree that performs as well as or even better than a more complex tree can be found through a pruning process. The suggestion made by

Breiman, et al (Breiman, Friedman, Olshen, & Stone, 1984) is to find a nested sequence of subtrees by successively pruning branches of the overly large tree. The best tree from this sequence is chosen based on the misclassification rate estimated by cross-validation or an independent test sample. That means a simpler subset of that tree may give a smallest error, because some of the decision rules in the full tree hinder rather than help. Figure 5-6 shows the tree test result using a tenfold cross-validation in which a subset of 10% of the data was set aside for validation and the remaining 90% of the data was used for training. First the resubstitution error or training error (the proportion of the original observations that were misclassified by various subsets of the original tree) was computed. Then cross-validation was used to estimate the true error for trees of various sizes. Figure 5-6 indicates that the resubstitution error is overly optimistic. It decreases as the tree size grows, but the cross-validation results show that beyond a certain point, increasing the tree size increases the error rate. The tree with the smallest cross-validation error was chosen. While this may be satisfactory, Breiman, et al (Breiman, et al., 1984) suggested that the preferred way is to use a simpler tree if it is roughly as good as a more complex tree. This tree is even smaller than the tree with the smallest cross-validation error. The rule he proposed is to take the simplest tree that is within one standard error of the minimum. This is shown in Figure 5-6. It was obtained by computing a cutoff value that is equal to the minimum cost plus one standard error. It can be seen from Figure 5-6 that the tree with the smallest cross validation error, which is 0.20, has 10 leaf nodes. One standard error (0.0290) was added to the minimum cost to get the cutoff value of 0.2290. The best pruned is the smallest tree under this cutoff. The solid line shows the estimated cost for each tree size, the dashed line marks one standard error above the minimum, and the square marks the smallest tree under the dashed line. Figure 5-7 displays the pruned classification tree, which has 9 leaf nodes as indicated in Figure 5-6 as the best choice. It is the smallest tree under the cutoff value and has a cross validation error of 0.2056. And for this Eye bacteria data, the misclassification error cost is 20.56%.

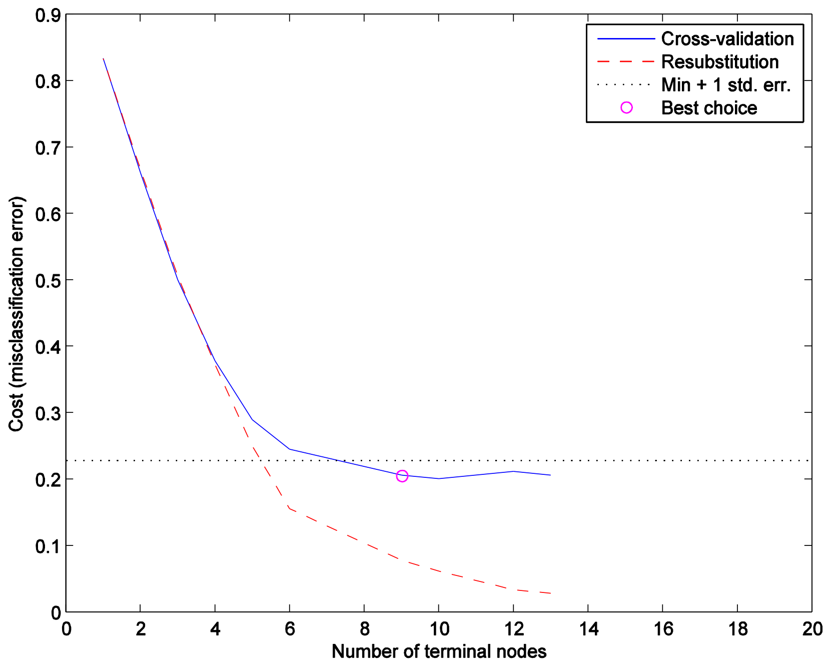


Figure 5-6: The estimated error for cross validation and resubstitution

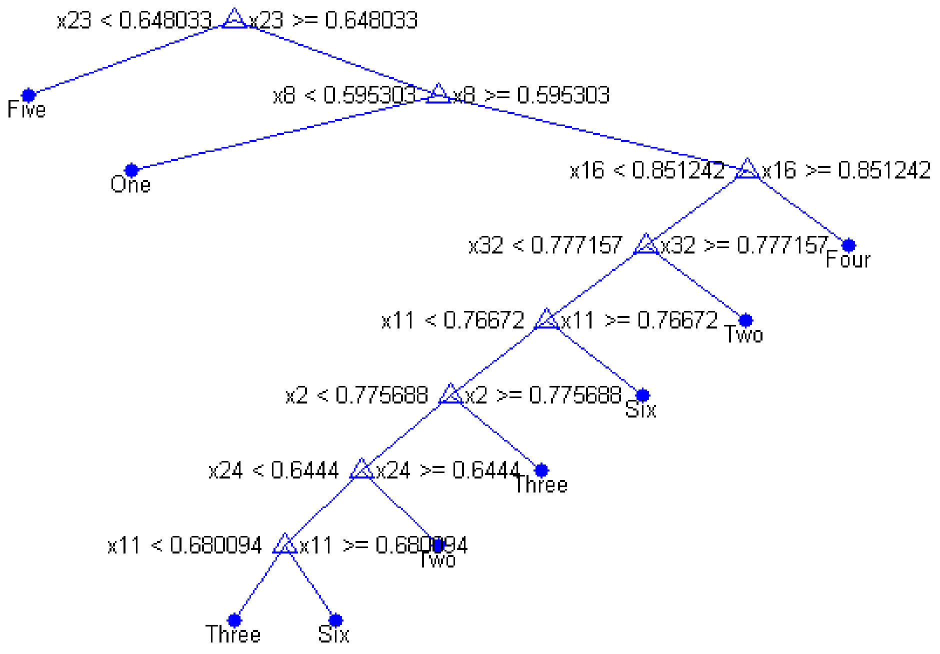


Figure 5-7: The optimal tree with nine terminal nodes after pruning

In order to decrease the misclassification error cost, the bagging technique was also introduced to the decision tree. It is statistically proven[1] that drawing *N* out of *N* observations with replacement omits on average 37% of observations for each decision tree (Tan, Steinbach, & Kumar, 2005). These are called "out-of-bag" observations. The out-of-bag observations can be served as test data set. They can be used to estimate the predictive power and feature importance. For each observation, the out-of-bag prediction was estimated by averaging over predictions from all trees in the ensemble for which this observation is out of bag. Then the computed prediction was compared against the true response for this observation. By comparing the out-of-bag predicted

[1] Each sample has a probability $1-(1-\frac{1}{N})^N$ of being selected in each bootstrap sample. If N is sufficiently large, this probability converges to$1-1/e \cong 0.632$.

responses against the true responses for all observations used for training, the average out-of-bag error was achieved. This out-of-bag average is an unbiased estimator of the true ensemble error. The solid line in Figure 5-8 shows the out-of-bag error curve for the ensemble classifier. It can be seen that the misclassification error cost has decreased significantly to 9.44%.

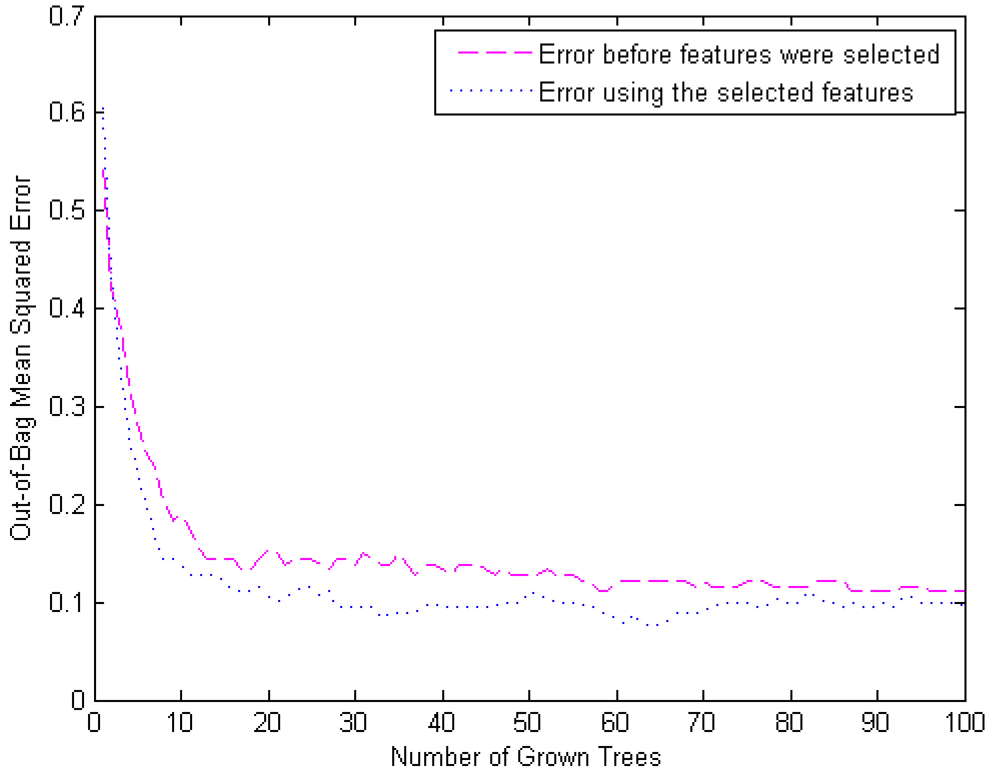


Figure 5-8: Misclassification error cost curve

5.6 Feature selection

Usually the prediction ability should depend more on important feature and less on unimportant features. Another attractive feature of bagged decision trees is its ability to select the important features by randomly permuting the value of this feature across all of the observations in the data set and measuring and estimating the increase of the out-of-bag mean squared error due to this permutation. The larger the increase, the more important the feature is. Thus, we do not need to supply test data for bagged ensembles because we can obtain reliable estimates of the predictive power and feature importance in the process of training, which is an attractive feature of bagging. Through selecting the most important features, the dimension of the training data can be reduced significantly and the computation cost can be lower. Figure 5-9 shows the out-of-bag feature importance for all the 32 features. Using an arbitrary cutoff at 0.6, the most important 8 features were selected. The lower dimensional data (the selected features) was then used to train the ensemble classifier following the same procedure. The dotted line in Figure 5-8 shows the out-of-bag error using the selected features. The performance is obviously at least as good as the one using all of the data.

The optimal subset selected by Bagging Decision tree (i.e. 2, 6, 8, 9, 11, 23, 24, 32) is different compared with results obtained by Boilot (Biilot, 2003; Boilot, et al., 2002; J. W. Gardner, et al., 2005) (i.e. 1, 8, 9, 11, 14, 15, 17, 23, 31, 32), who applied the so-called V-integer GA using PNN classification algorithm. They achieved an accuracy of average 82.0% with these ten features. Their experiment also showed that as the number of features to be selected becomes 8, the accuracy decreased to 78.6% as well. The application of our techniques in the following paragraph and section will show a better classification result using the selected 8 features.

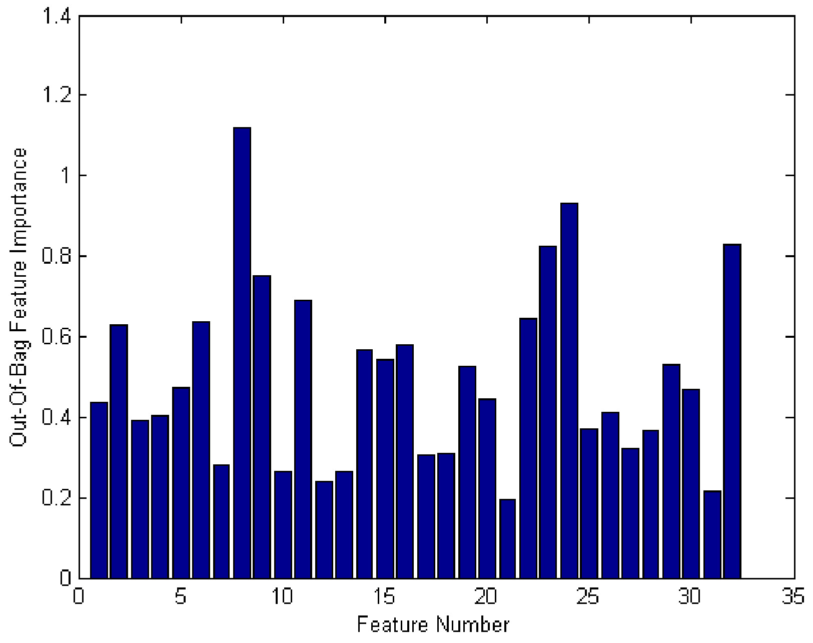


Figure 5-9: Feature importance histogram

After the features were selected, the Bagging MLP simulation process was repeated again using the selected features to test the efficiency. Table 5-2 shows the relationship between the MME threshold and accuracy after using the selected features to train MLP using two different algorithms. It can be seen that the accuracy can reach 100% if the MME threshold was set to 0.5. Figure 5-10 shows the linear regression relationship between the network outputs and the corresponding targets using selected features. It can be seen that the output tracks the target well for both test and validation set, both with a significantly high regression value.

Table 5-2: MME vs. Accuracy

MME threshold	Accuracy for LM	Accuracy for GD
0.6	100.00%	99.44%
0.5	100.00%	98.33%
0.4	98.33%	94.44%
0.3	96.67%	88.89%
0.2	94.44%	66.67%

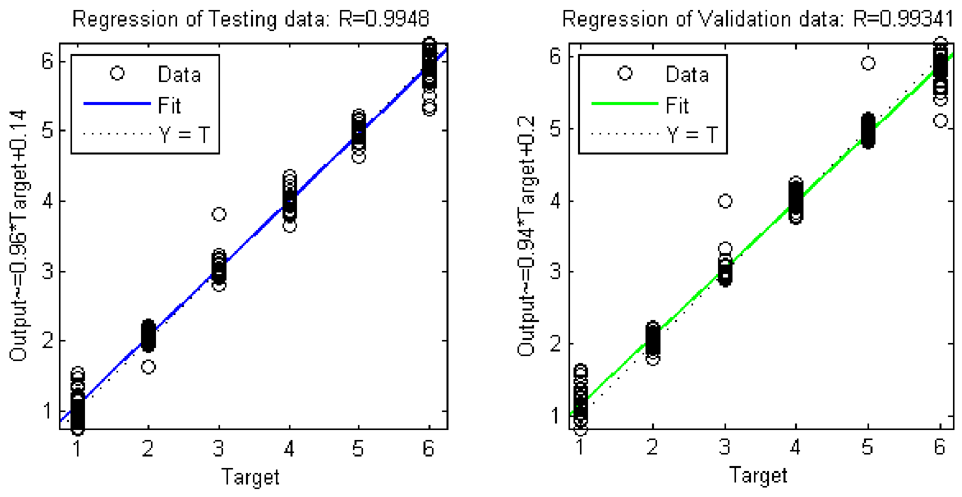


Figure 5-10: Regression of test and validation data using selected features

An ensemble of classifiers is a set of classifiers whose individual classifier decisions are combined in a way to classify a new instance. It performs well when the single classifier is unstable. In our example, the performance of the ensemble classifiers has significantly improved. Bagging to decision tree has decreased the error from above 20 percent to below 10 percent while bagging to MLP can achieve accuracy as high as 100 percent.

5.7 Classifier design using Optimized Weighted Fuzzy Decision Tree

In order to classify the eye bacteria data successfully, a classifier should be designed to make correct predictions of those possible bacteria. Following the procedure listed in Chapter 3.2.2, a fuzzy decision tree is designed using Eye Bacteria data. After the tree construction process, the FDT consists of 29 internal nodes and 41 external (leaf) nodes. Each leaf node is a reasoning path and is represented as a rule in the following reasoning mechanism. 41 leaf nodes will be considered as 41 reasoning rules.

5.7.1 Fuzzify input

The features of each instance of eye bacteria data will be fuzzified into three membership values according to three linguistic terms. As introduced in Chapter 3.3.1, FCM was used to cluster the feature values into three clusters. Three cluster centers and the maximum and minimum values along each feature will be used as parameters to fuzzify the features.

Let $F_{j,high}$, $F_{j,med}$ and $F_{j,low}$ be the three sorted cluster centers along each feature. The cluster centers along these eight features are listed as:

$$\begin{bmatrix} F_{1,high} & F_{2,high} & F_{3,high} & F_{4,high} & F_{5,high} & F_{6,high} & F_{7,high} & F_{8,high} \\ F_{1,med} & F_{2,med} & F_{3,med} & F_{4,med} & F_{5,high} & F_{6,high} & F_{7,high} & F_{8,high} \\ F_{1,low} & F_{2,low} & F_{3,low} & F_{4,low} & F_{5,high} & F_{6,high} & F_{7,high} & F_{8,high} \end{bmatrix} =$$

$$\begin{bmatrix} 0.8398 & 0.8549 & 0.8720 & 0.7910 & 0.8611 & 0.8551 & 0.8148 & 0.8529 \\ 0.7911 & 0.7911 & 0.7460 & 0.7812 & 0.7694 & 0.7397 & 0.7848 & 0.7782 \\ 0.6843 & 0.5645 & 0.5987 & 0.6496 & 0.4941 & 0.5518 & 0.6482 & 0.5675 \end{bmatrix}$$

The fuzzified membership values of an input pattern along the j^{th} feature, corresponding to the three-dimensional linguistic term are formulated as Equations 10, 11 and 12 in chapter three.

5.7.2 Reasoning mechanism

Following the procedure of mapping weighted fuzzy decision tree to neural networks in Chapter 3.3.5, the local and global weights can be updated and optimized through the learning process. After the weights have been finalized, the reasoning mechanism will be introduced to give certainty factors to each leaf node.

First the certainty factor of each node was defined. Since each node will classify the input instance to different classes with a different certainty, the certainty of each node can be calculated as Equation 27 in Chapter three (taking node one for example):

$$\alpha_1 = \{\frac{D_1^{C_1}}{D_1}, \frac{D_1^{C_2}}{D_1}, \frac{D_1^{C_3}}{D_1}, \frac{D_1^{C_4}}{D_1}, \frac{D_1^{C_5}}{D_1}, \frac{D_1^{C_6}}{D_1}\}$$

Certainty factors associated with all the 41 leaf nodes for the classes are listed below:

$$\begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} & \alpha_{1,6} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} & \alpha_{2,6} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} & \alpha_{3,6} \\ & & & . & & \\ & & & . & & \\ & & & . & & \\ \alpha_{39,1} & \alpha_{39,2} & \alpha_{39,3} & \alpha_{39,4} & \alpha_{39,5} & \alpha_{39,6} \\ \alpha_{40,1} & \alpha_{40,2} & \alpha_{40,3} & \alpha_{40,4} & \alpha_{40,5} & \alpha_{40,6} \\ \alpha_{41,1} & \alpha_{41,2} & \alpha_{41,3} & \alpha_{41,4} & \alpha_{41,5} & \alpha_{41,6} \end{bmatrix}$$

$$= \begin{bmatrix} 0.0535 & 0.0190 & 0.0193 & 0.0000 & 0.8674 & 0.0407 \\ 0.0000 & 0.8120 & 0.0023 & 0.0000 & 0.1856 & 0.0000 \\ 0.0000 & 0.9015 & 0.0046 & 0.0000 & 0.0939 & 0.0000 \\ & & & . & & \\ & & & . & & \\ & & & . & & \\ 0.0401 & 0.0209 & 0.0350 & 0.1595 & 0.0000 & 0.7446 \\ 0.0080 & 0.0313 & 0.2386 & 0.3738 & 0.0000 & 0.3483 \\ 0.0000 & 0.0199 & 0.2095 & 0.7068 & 0.0000 & 0.0639 \end{bmatrix}$$

The certainty factor of each leaf node indicates the certainty of classifying an instance into different classes. Taking leaf node 39 as an example, this leaf node has a certainty of 0.7446 to classify an instance into class six while it has a certainty of 0 of classifying this instance into class five. For each leaf node, the certainty factor and the membership value of that path (leaf node) as well as the global weights are incorporated together to infer the classification result as detailed in Chapter 3.3.5. To get the final result, all the forty-one leaf nodes are combined together to produce the prediction.

5.7.2 Simulation Results

After the construction process, the eye bacteria dataset was tested using this classifier. Half of the dataset was randomly chosen to grow, update and optimize the weights of the decision tree and the other 50% was used as a test dataset to test the accuracy of the tree. The training and test process will be repeated five times. The accuracy was the average of five. In order to compare its performance, the conventional Fuzzy Decision Tree (FDT) and Weighted Fuzzy Decision Tree (WFDT), which doesn't include the optimization process for weight updating, were also introduced as benchmark classifiers.

The growing process has found that if the features cannot be selected and dimensions cannot be reduced, the fuzzy decision tree would face the problem of the curse of dimensionality. So the dataset which incorporates the selected features will be used for the testing purpose. The classification accuracy was calculated as: $accuracy = \frac{n_r}{n} * 100\%$, where *n* is the total number of test instances and n_r is the number of instances which had been correctly classified. It has been found that the Optimized Weighted Fuzzy Decision tree can enhance the classification accuracy from 78.33% for Fuzzy decision tree to 85.56% while the weighted fuzzy decision tree reached an accuracy of 82.17%.

In addition to the improvement of the accuracy, OWFDT has also been able to reduce the iteration numbers used to train the mapped neural network, which has made the optimization of the weights more efficient. The simulation has found OWFDT took an average of 197 epochs to reach the optimal weights while weighted FDT needed 487 epochs to finish the training process.

5.8 Conclusion

In the simulation, we try to classify six kinds of bacteria which are responsible for eye infection. First the MLP and the decision tree were employed in order to discriminate between these six bacteria species. Although a single MLP can classify the data set, the accuracy was not high enough and the results were highly unstable and dependent on the network initialization. The decision tree constructed was able to discriminate different bacteria classes to some extend but with a high error cost. By introducing the bagging technique to MLP, we were not only able to enhance the accuracy but also stabilize the performance of both classifiers. By bagging the decision tree, we have successfully extracted several most important features and reduced the dimensionality of the original data. In addition to that, the error cost of the classification tree has also reduced significantly making the decision tree more reliable. By using the selected features to grow the fuzzy decision tree, the problem of curse of dimensionality had been solved. The optimized weighted fuzzy decision tree had also enhanced the accuracy of the conventional FDT as well as reduced the iteration epochs of weighted FDT.

Reference

Biilot, P. (2003). *Novell Intelligent data processing techniques for electronic nose: Feature Selection and Neuro-fuzzy Knowledge base.* University of Warwick.

Boilot, P., Hines, E. L., Gardner, J. W., Pitt, R., John, S., Mitchell, J., et al. (2002). Classification of bacteria responsible for ENT and eye infections using the Cyranose system. *Sensors Journal, IEEE, 2*(3), 247-253.

Braga, P. L., Oliveira, A. L. I., Ribeiro, G. H. T., & Meira, S. R. L. (2007). *Bagging Predictors for Estimation of Software Project Effort.* Paper presented at the Neural Networks, 2007. IJCNN 2007. International Joint Conference on.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees*: Wadsworth International Group.

Davison, A. C., & Hinkley, D. V. (1997). *Bootstrap Methods and Their Application (Cambridge Series in Statistical and Probabilistic Mathematics , No 1)*: Cambridge University Press.

Dutta, R., Hines, E., Gardner, J., & Boilot, P. (2002). Bacteria classification using Cyranose 320 electronic nose. *BioMedical Engineering OnLine, 1*(1), 4.

Dutta, R., Hines, E. L., Gardner, J. W., Udrea, D. D., & Boilot, P. (2003). Non-destructive egg freshness determination: an electronic nose based approach. *Measurement Science and Technology, 14*(2).

Efron, B., & Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. London: Chamman and Hall.

Gardner, J. W., & Bartlett, P. N. (1994). A brief history of electronic noses. *Sensors and Actuators B: Chemical, 18*(1-3), 210-211.

Gardner, J. W., Boilot, P., & Hines, E. L. (2005). Enhancing electronic nose performance by sensor selection using a new integer-based genetic algorithm approach. *Sensors and Actuators B: Chemical, 106*(1), 114-121.

Jang, J. S. R., Sun, C. T., & Mizutani, E. (1997). *Neuro-Fuzzy and Soft Computing-A Computational Approach to Learning and Machine Intelligence*: Prentice Hall.

Llobet, E., Hines, E. L., Gardner, J. W., & Franco, S. (2004). Non-destructive banana ripeness determination using a neural network-based electronic nose. *Measurement Science and Technology, 10*(6).

Mooney, C. Z., & Duval, R. D. (1993). *Bootstrapping: A Nonparametric Approach to Statistical Inference*: Sage Publications.

Tan, P.-N., Steinbach, M., & Kumar, V. (2005). *Introduction to Data Mining, (First Edition)*: Addison-Wesley Longman Publishing Co., Inc.

Wendy, L. M., & Angel, R. M. (2007). *Computational Statistics Handbook with MATLAB, Second Edition*: Chapman Hall/CRC.

Chapter VI: Classification of CLASH Dataset using OWFDT and other Intelligent Systems Techniques

In the previous chapters, the effectiveness and efficiency of OWFDT has been evaluated through a comparison of its classification results with fuzzy decision tree and weighted fuzzy decision tree. In chapter 3 these three techniques were compared using Iris data, Wisconsin Breast Cancer data and Brain Computer Interface data. Chapter 4 utilized composite material data and chapter 5 introduced eye bacteria data to evaluate OWFDT's performance. In this chapter, a benchmark study using the CLASH dataset will be conducted to compare and evaluate the effectiveness and efficiency of OWFDT as compared to some other widely used hybrid ISTs including ANFIS, EFuNN, GNMM, Fuzzy ARTMAP etc. in order to explore the relative merits of adopting OWFDT.

6.1 Introduction and motivation

Wave Overtopping (WO) is defined as the sea water which is flowing over the crest of a coastal structure land-inward. WO occurs when the highest run-up levels reach the crest of the structure and exceed the free board and pass over it. Accurate prediction tools for wave overtopping are necessary in order to improve the water management efficiency in coastal zones (Geeraerts, Troch, De Rouck, Verhaeghe, & Bouma, 2007).

There are two existing approaches to measuring and assessing WO at coastal structures. The first approach deals with the overtopping volume per overtopping wave. The second approach considers mean overtopping discharges over certain time intervals and per meter structure width. The second approach is widely accepted because of the uneven distribution of overtopping in time and in space caused by irregular wave action (Verhaeghe, De Rouck, & van der Meer, 2008).

6.2 Dataset

The database was collected for the European research project CLASH (Crest Level Assessment of Coastal Structure by full scale monitoring, neural network prediction and Hazard analysis on permissible wave overtopping) (Meer, W., Verhaeghe, & G.J.Steendam, 2005). This extensive database on wave overtopping consists of two phases. A first preliminary phase composed of overtopping information originating from before 2003, and consisted of about 6500 tests, which was released within CLASH in August 2003 as an intermediate result. In the second phase, from August 2003 to

December 2004, this preliminary database was enlarged and improved to a final database, consisting of 7107 overtopping tests, originating from 163 independent test series (Verhaeghe, Meer, & J.W., 2003; Verhaeghe, Meer, J.W., et al., 2003).

In the dataset, each test was denoted by a fixed number of parameters. These parameters had to be chosen in such a way that an as complete as possible overall view of the overtopping test was represented by these parameters. These parameters can be categorized into two groups: hydraulic parameters and structural parameters. The hydraulic parameters describe the wave characteristics and the measured overtopping, whereas the structural parameters describe the coastal structure. (van der Meer, Verhaeghe, & Steendam, 2009).

The extensive overtopping database provides the data for the development of intelligent systems techniques, but more information on the overtopping measurements than strictly needed for the development of the prediction method has been included in the database. Hadewych Verhaeghe examined the experimental model and eliminated those parameters which are considered to be irrelevant to the NN prediction model. After that, 16 parameters were selected to be used for the construction of neural network model (Verhaeghe, 2005). The 16 selected parameters, consisting of 5 hydraulic parameters, 11 structural parameters are used as input and output parameter.

Table 6-1 shows the 16 parameters as well as their function in the development of the prediction models.

Table 6-1: Parameters in the dataset and their function in the application

Nature	Parameters	Function
hydraulic	$H_{m0\ deep}\ [m]$	input
	$H_{m0\ toe}\ [m]$	input
	$T_{m-1,0\ toe}\ [s]$	input
	$\beta\ [°]$	input
	$q\ [m^3/s/m]$	output
structural	$h\ [m]$	input
	$h_t\,[m]$	input
	$B_t\,[m]$	input
	$\gamma_f\,[-\ ]$	input
	$cot\alpha_d\,[-\ ]$	input
	$cot\alpha_u\,[-\ ]$	input
	$R_c\,[m\ ]$	input
	$h_b\,[m\ ]$	input
	$B_h\,[m\ ]$	input
	$A_c\,[m\ ]$	input
	$G_c\,[m\ ]$	input

6.2.1 Output parameter

The mean overtopping discharge is the most common approach used to the design of coastal structures (TAW, 2002). The huge amount of available data regarding mean overtopping discharge measurements was an extra benefit of considering mean overtopping discharges as the output in this work (Lykke Andersen & Burcharth, 2009).

Within this dataset, mean overtopping discharge, denoted as q in $m^3/s/m$, logically serves as the output parameter for the prediction models.

According to its practical meaning, mean overtopping discharge may either be regarded as important or eligible for human activity, so the output was guided to two classes in order to design the classifiers. For this reason the output parameter q is replaced for the classifier by 2 possible values, referring to two 'classes' of overtopping:

- the value of q is replaced by '+1' if the overtopping discharge q is assessed to be significant and important to human activities;
- the value of q is replaced by '-1' if the overtopping discharge q is equal to zero or assessed to be negligible.

6.2.2 Preprocessing the data according to the Froude model scaling

The Froude scaling law was introduced to scale the dataset because the data were collected from different tests which were set up in different conditions. Usually, Froude scaling is applied to model flows. The Froude number is defined as the ratio of a characteristic velocity to a gravitational wave velocity. In fluid mechanics, the Froude number is used to determine the resistance of an object moving through water, and permits the comparison of objects of different sizes. The majority of hydraulic models in coastal engineering are scaled according to Froude model law (Ingram, Gao, Causon, Mingham, & Troch, 2009; van Gent, van den Boogaard, Pozueta, & Medina, 2007).

According to Hadewych Verhaeghe (Verhaeghe, 2005), the parameter $H_{m0\ toe}$ is one of the most important parameters regarding the overtopping phenomenon, and is in addition always positive. These characteristics have made it the best choice as scaling parameter. After scaling all tests with $H_{m0\ toe}$ as scaling parameter, all the test can be regarded as been scaled to a fictive situation in which the value of $H_{m0\ toe}$ = 1m. In this chapter, Hadewych Verhaeghe's scaling process was adopted to scale the dataset as listed in Table 6-2.

Table 6-2: parameter scaling according to Froude model law

Input	Output
$H_{m0\ deep} / H_{m0\ toe}$	$q / (H_{m0\ toe})^{1.5}$
$T_{m-1,0\ toe} / (H_{m0\ toe})^{0.5}$	
β	
$h / H_{m0\ toe}$	
$h_t / H_{m0\ toe}$	
$B_t / H_{m0\ toe}$	
γ_f	
$cot\alpha_d$	
$cot\alpha_u$	
$R_c / H_{m0\ toe}$	
$h_b / H_{m0\ toe}$	
$B_h / H_{m0\ toe}$	
$A_c / H_{m0\ toe}$	
$G_c / H_{m0\ toe}$	

Table 6-2 shows the parameter characteristics after scaling to $H_{m0\ toe}$ = 1m according to Froude model law.

Table 6-3: Statistical characteristics for the dataset

Parameter	Minimum	Maximum	Mean	Standard Deviation
$H_{m0\ deep}$	0.692	6.304	1.119	0.333
$T_{m-1,0\ toe}$	2.721	86.244	5.070	3.134
β	0	80.000	3.517	11.218
h	0.904	64.476	3.900	3.311
h_t	0.429	45.429	3.364	2.887
B_t	0	50.000	0.923	2.057
γ_f	0.330	1	0.697	0.274
$cot\alpha_d$	0	7.000	1.943	1.446
$cot\alpha_u$	-5.000	9.706	1.859	1.723
R_c	0	62.390	1.571	1.152
h_b	-2.294	8.411	0.053	0.446
B_h	0	38.462	0.694	2.160
A_c	0	62.390	1.470	1.168
G_c	0	39.000	0.989	1.537

Table 6-3 and Figure 6-1 show the statistical values for the dataset. The scaling process had successfully dismissed the distinction between small and large scale test. In addition to that, the scaled minima and maxima give a more comparable overall view of the tests. It also gives an idea of the minima and maxima which would have appeared if all tests were performed with wave heights of 1m at the toe of the structure, keeping all other parameters in proportion.

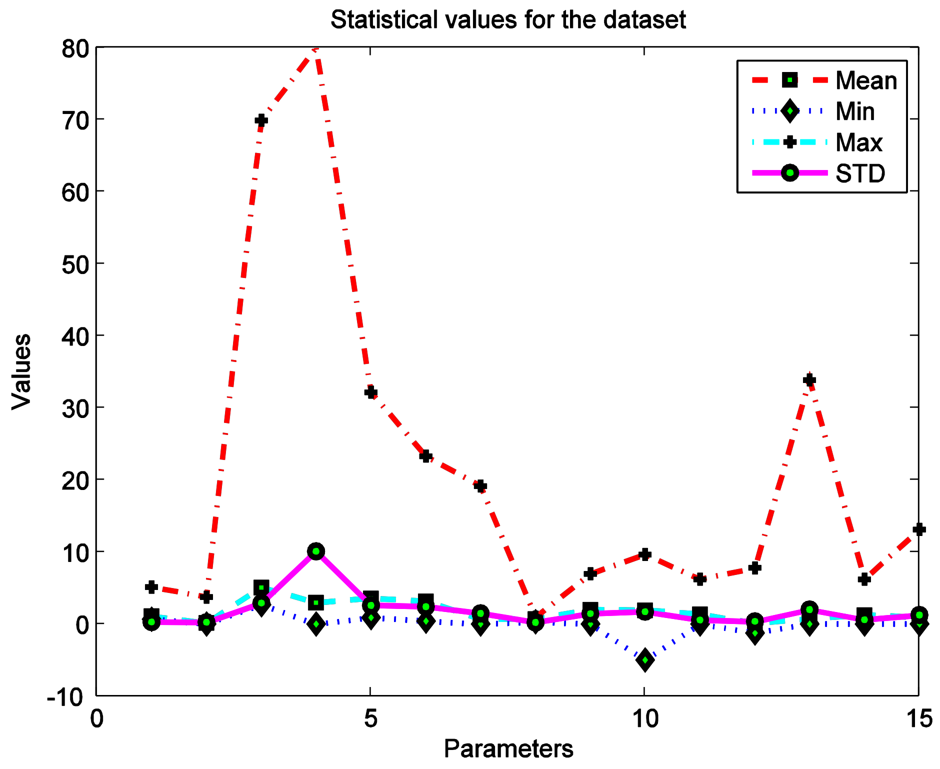


Figure 6-1: Statistical characteristics for the dataset

Principal Component Analysis (PCA) was used to visualize the data and explore the data clustering. Figure 6-2 shows the percentage of the first five principal components in CLASH data. It indicates that there is a clear break in the amount of variance accounted for by the first and second principal components. While the first component accounts for nearly 75% of the variance and the second component explains 9% of the variance, the first two principal components account for 85% of the variance. It is obviously reasonable to assert that these two principal components represent the dataset. But these two classes are also evident with some overlapping, which suggests that these two classes are somehow separable.

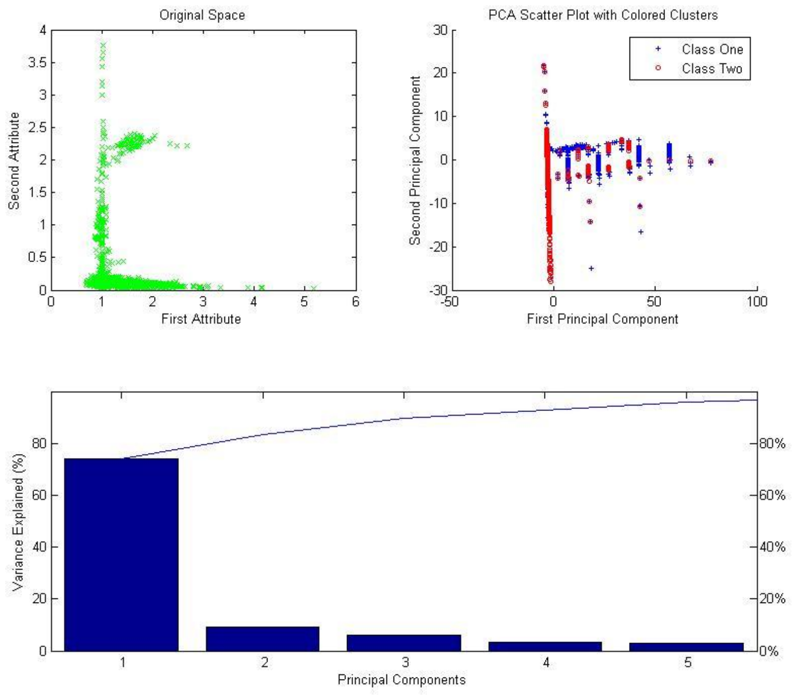


Figure 6-2: Principal component analysis for the dataset

6.3 Neural Network Simulation results

60 percent of the samples were used as training dataset, 20 percent of the samples were used as validation set and the left 20 percent were regarded as test dataset. After training an MLP model with the training dataset, the independent test set is used to

assess the performance of the constructed NN. The Root-Mean-Square Error (RMSE) of this independent test set, which is defined in equation 1, was introduced to assess its performance (the scaled parameters will be denoted with a left superscript s):

$$rmse = \sqrt{\frac{1}{N_{test}} \sum_{n=1}^{N_{test}} [({}^{s}q_{measured\ n}) - ({}^{s}q_{NN_n})]^2} \qquad (1)$$

where N_{test} indicates the number of test data, ${}^{s}q_{measured\ n}$ refers to the predicted output and ${}^{s}q_{NN_n}$ refers to the output desired by the NN. The values of ${}^{s}q_{measured\ n}$ and ${}^{s}q_{NN_n}$ are both expressed as the logarithm of the mean overtopping discharges which are in $m^3/s/m$.

The lower the value of RMSE, the better the overall prediction capability of the considered NN will be.

The MLP model consists of one input layer with 15 input neurons to accommodate 15 parameters (all the parameters have been scaled according to Froude, with the value of $H_{m0\ toe}$ as scaling parameter), one hidden layer with hidden neurons and an output layer with the preprocessed, scaled according to Froude, mean overtopping discharge, $log({}^{s}q)$, as the output parameter.

The number of hidden neurons is determined by training and testing several models, where the number of hidden neurons is varied. The optimal acceptable number of hidden neurons is determined by comparing the performance of the models with different number of hidden neurons.

It should be noticed that one extra hidden neuron leads to a significant increase of the network complexity. Figure 6-3 shows the value of RMSE for trained models with the number of hidden neurons varying from 15 up to 30. Since the input layer has 15 neurons, we set the minimum number of hidden neurons to be 15. In case of 15 input parameters and 1 output parameter, 16 additional parameters have to be determined during the training process for one extra hidden neuron, so the maximum number of neurons in hidden layer was set to be 30, although this number is still subject to testing. Simulations were run with the training sets and test sets chosen randomly and the initialization of the weights and biases set arbitrarily for each model. Consequently, the final result as well as the value of RMSE depends on these random choices, which explains the slight fluctuation of the curve in Figure 6-3.

Generally speaking, the higher the number of hidden neurons, the lower the value of RMSE will be. But there is no evidence of a solid relationship between the RMSE and the number of hidden neurons. As can be seen from Figure 6-3, there is no significant improvement for a number of hidden neurons larger than 22. As simplicity is preferred over needless complexity, the optimal number of hidden neurons is chosen to be 22.

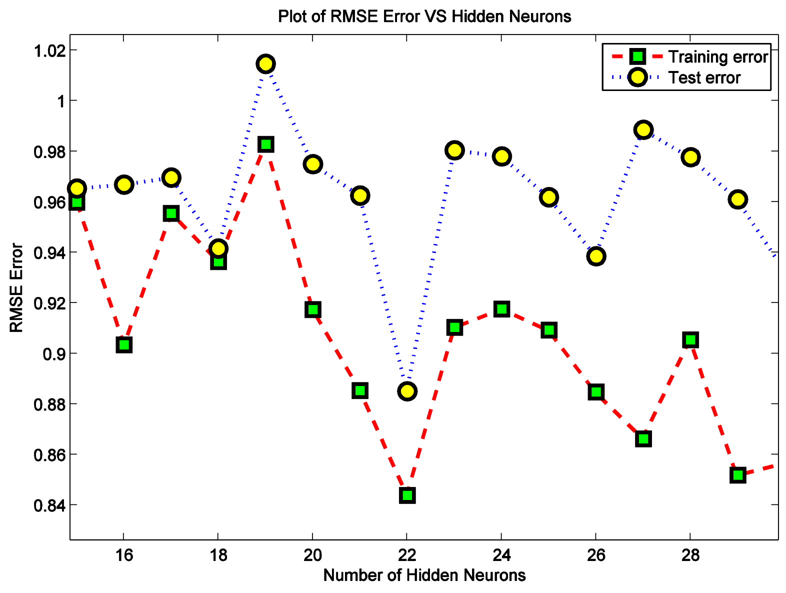


Figure 6-3: RMSE VS number of hidden neurons

The training process stopped when the validation error increased for six iterations, which occurred at iteration 25. Figure 6-4 shows a plot of the training errors, validation errors, and test errors. For the CLASH data, the result is reasonable because the final mean-square error is small and the test dataset error and the validation dataset error have similar characteristics. And in addition to that no significant overfitting has occurred by iteration 25 (where the best validation performance occurs).

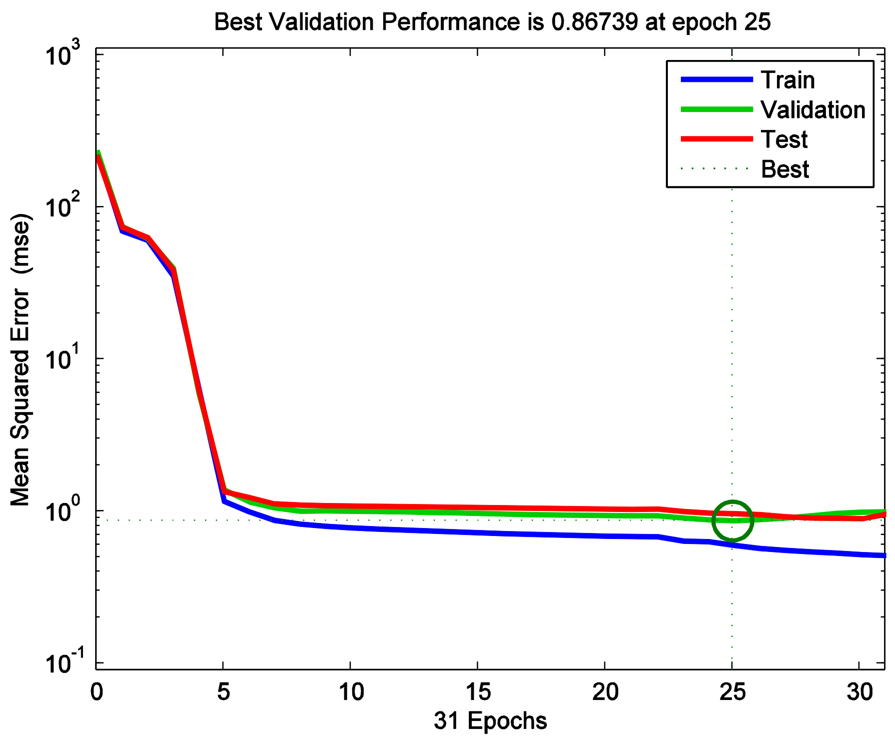


Figure 6-4: Training performance of neural networks

A linear regression between the network outputs and the corresponding targets was also performed in order to analyze the network response. Figure 6-5 shows the regression results. It indicates that the output tracks the targets very well for training, test, and validation datasets, and the regression value is above 0.90 for test dataset and over 0.93 for the entire dataset.

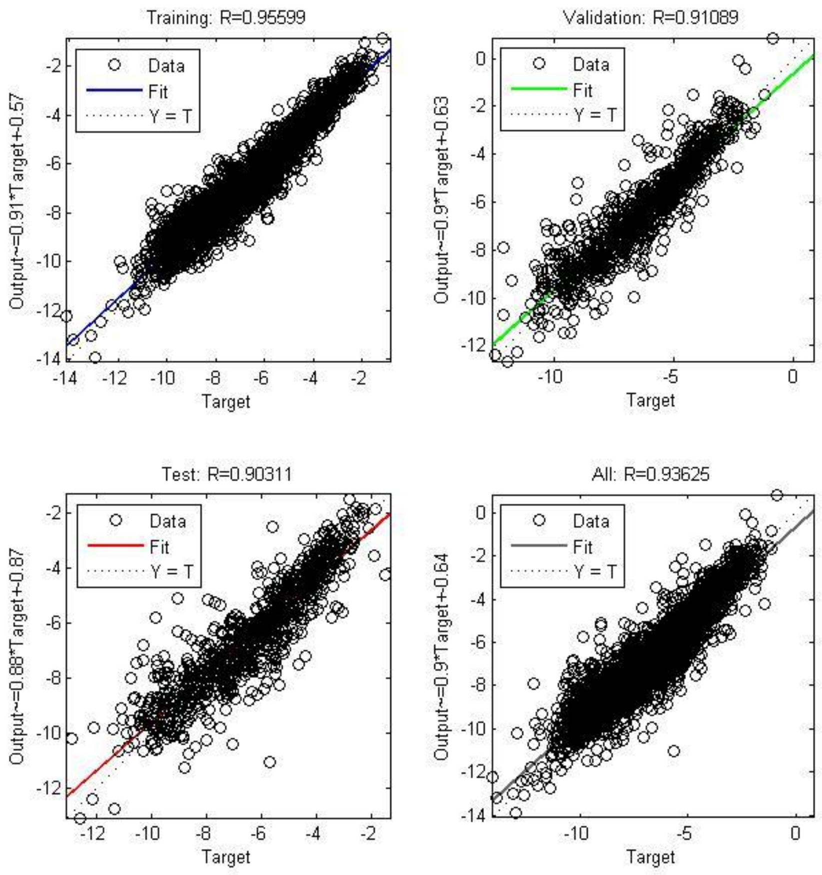


Figure 6-5: Regression results for 1) training, 2) validation, 3) test and 4) all the datasets

6.4 OWFDT results

As introduced in Chapter 3.3.1, each feature (attribute) of the data was fuzzyfied into three membership values according to three linguistic terms using Fuzzy C-Means (FCM) clustering algorithm. The FCM algorithm was used to cluster the data and find the three cluster centers. Three cluster centers and the maximum and minimum values along each feature are used to fuzzify the features using equations 10, 11, 12 in chapter three.

Let $F_{j,high}$, $F_{j,med}$ and $F_{j,low}$ be the three sorted cluster centers along each feature. The cluster centers along these fifteen features are listed as follows:

$$\begin{bmatrix} F_{1,high} & F_{2,high} & F_{3,high} & F_{4,high} & \cdots & F_{12,high} & F_{13,high} & F_{14,high} & F_{15,high} \\ F_{1,med} & F_{2,med} & F_{3,med} & F_{4,med} & \cdots & F_{12,med} & F_{13,med} & F_{14,med} & F_{15,med} \\ F_{1,low} & F_{2,low} & F_{3,low} & F_{4,low} & \cdots & F_{12,low} & F_{13,low} & F_{14,low} & F_{15,low} \end{bmatrix} =$$

$$\begin{bmatrix} 1.0109 & 0.1060 & 4.2900 & 0.3442 \ldots & 0.0185 & 0.4920 & 1.1510 & 0.4398; \\ 1.0399 & 0.1730 & 4.3587 & 0.8491 \ldots & 0.0304 & 0.5267 & 1.1612 & 0.7426 \\ 1.1348 & 0.1731 & 5.0886 & 37.4091 \ldots & 0.3762 & 1.1805 & 1.2920 & 0.9409 \end{bmatrix}$$

The constructed FDT consists of 21 internal nodes and 43 leaf nodes. Each leaf node represents a reasoning rule. Certainty factors associated with all the 43 leaf nodes for the classes are used to infer the classification results and listed below:

$$\begin{bmatrix} \alpha_{51} & \alpha_{5,2} \\ \alpha_{9,1} & \alpha_{9,2} \\ \alpha_{10,1} & \alpha_{10,2} \\ & . \\ & . \\ & . \\ \alpha_{62,1} & \alpha_{62,2} \\ \alpha_{63,1} & \alpha_{63,2} \\ \alpha_{64,1} & \alpha_{64,2} \end{bmatrix} = \begin{bmatrix} 0.9581 & 0.0419 \\ 0.0000 & 1.0000 \\ 1.0000 & 0.0000 \\ & . \\ & . \\ & . \\ 0.9252 & 0.0748 \\ 0.0000 & 1.0000 \\ 0.8777 & 0.1223 \end{bmatrix}$$

The certainty factor and the membership value of each leaf node as well as the global weights of that node are incorporated together to infer the classification result. To get the final result, all the forty-three leaf nodes are combined together to produce the prediction.

After the tree was constructed, the CLASH dataset was tested using OWFDT. Half of the dataset was randomly chosen to grow, update and optimize the weights of the decision tree and the other half was used as a testing dataset to test the accuracy of the tree.

The classification accuracy was calculated as: $accuracy = \frac{n_r}{n} * 100\%$, where n is the total number of test instances and n_r is the number of instances which had been correctly classified. After the testing process, it has been found that the OWFDT can achieve a classification accuracy of 79.28% using its 43 weighted rules which outpaced the accuracy of 72.37% of traditional FDT and 74.12% of weighted FDT.

6.5 Simulation results of other benchmarking techniques

6.5.1 ANFIS

ANFIS applies a combination of the least-squares method and the back propagation gradient descent method for training Fuzzy Inference System membership function parameters to emulate a given training dataset. It was also utilized to analyze the CLASH dataset as well.

When the CLASH data was processed using the ANFIS system with its default grid partitioning of the input space (Cornelius, 1999; Karray & Silva, 2004), the system fails to converge because of the problem of the *curse of dimensionality*, since ANFIS generates as many as 32768 (i.e. 2^{15}) rules when two MFs are used for each input and 14348907 (i.e. 3^{15}) rules when three MFs are used for each input, which are huge and unacceptable numbers of rule permutations. Meanwhile, ANFIS will end up with the problem of the curse of dimensionality and cannot converge as well when the FCM clustering algorithm was used to partition the input space. However, if the ANFIS structure is generated using subtractive clustering, which considerably reduces the number of rules, ANFIS managed to achieve a good classification results as will be shown in the next paragraph.

After 300 iterations, ANFIS reached a RMSE of 0.32 with a classification accuracy of 64.54%. Figure 6-6 shows the ANFIS structure where six membership functions for each input pattern and six rule nodes are displayed.

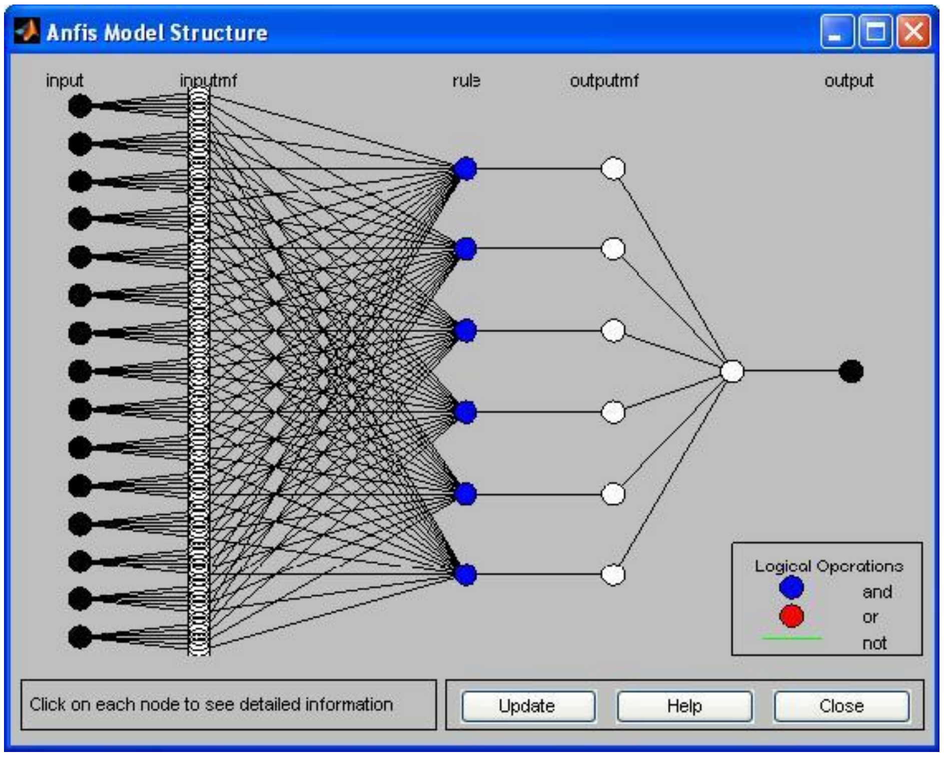


Figure 6-6: Overview of ANFIS structure

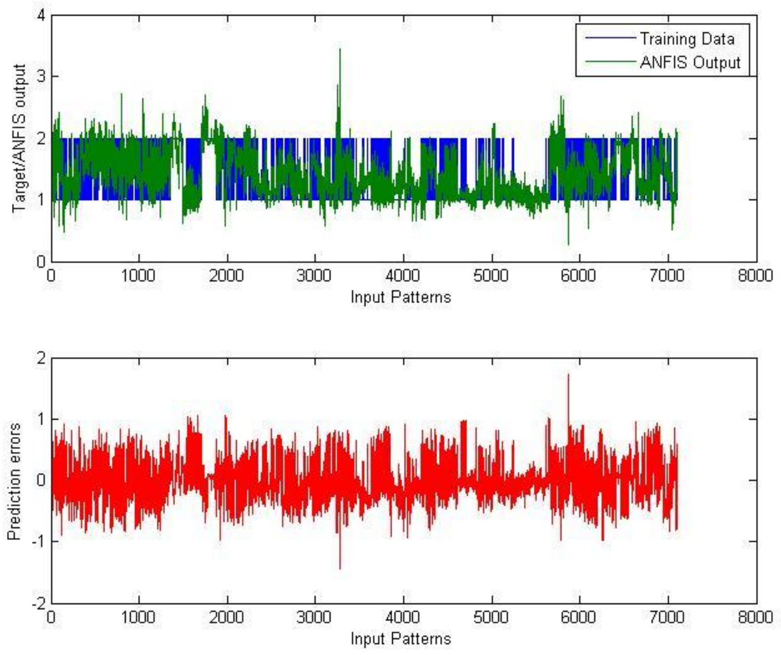
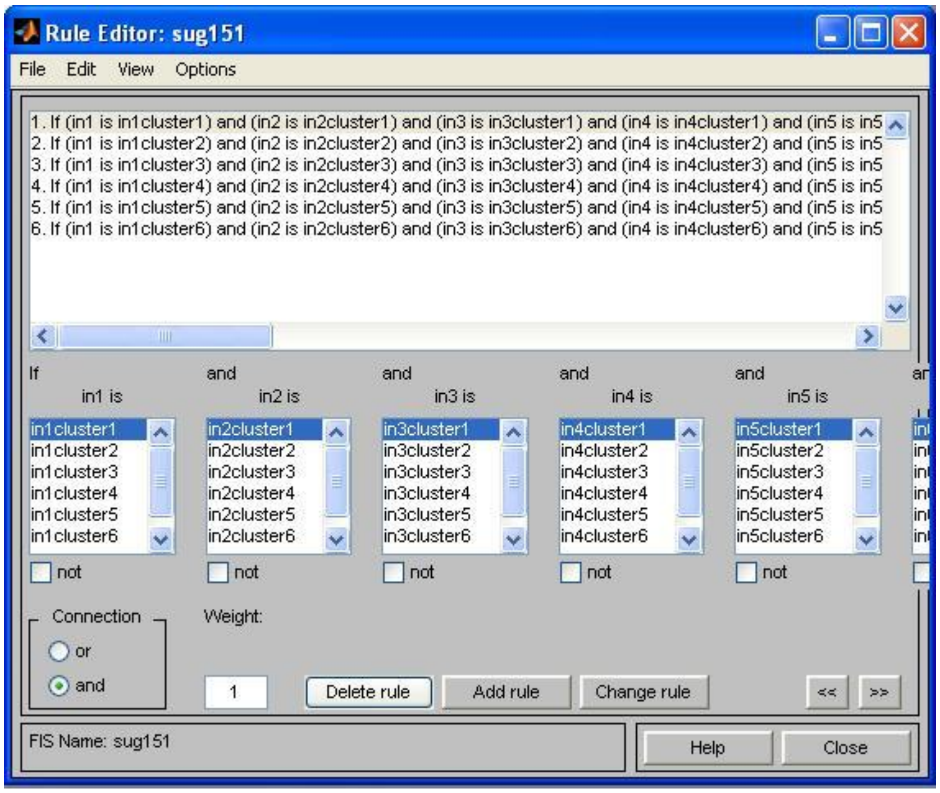


Figure 6-7: ANFIS prediction results and error distribution



```
1. If (in1 is in1cluster1) and (in2 is in2cluster1) and (in3 is in3cluster1) and (in4 is in4cluster1) and (in5 is in5
2. If (in1 is in1cluster2) and (in2 is in2cluster2) and (in3 is in3cluster2) and (in4 is in4cluster2) and (in5 is in5
3. If (in1 is in1cluster3) and (in2 is in2cluster3) and (in3 is in3cluster3) and (in4 is in4cluster3) and (in5 is in5
4. If (in1 is in1cluster4) and (in2 is in2cluster4) and (in3 is in3cluster4) and (in4 is in4cluster4) and (in5 is in5
5. If (in1 is in1cluster5) and (in2 is in2cluster5) and (in3 is in3cluster5) and (in4 is in4cluster5) and (in5 is in5
6. If (in1 is in1cluster6) and (in2 is in2cluster6) and (in3 is in3cluster6) and (in4 is in4cluster6) and (in5 is in5
```

If in1 is	and in2 is	and in3 is	and in4 is	and in5 is
in1cluster1	in2cluster1	in3cluster1	in4cluster1	in5cluster1
in1cluster2	in2cluster2	in3cluster2	in4cluster2	in5cluster2
in1cluster3	in2cluster3	in3cluster3	in4cluster3	in5cluster3
in1cluster4	in2cluster4	in3cluster4	in4cluster4	in5cluster4
in1cluster5	in2cluster5	in3cluster5	in4cluster5	in5cluster5
in1cluster6	in2cluster6	in3cluster6	in4cluster6	in5cluster6
☐ not	☐ not	☐ not	☐ not	☐ not

Connection: or [✓ and]

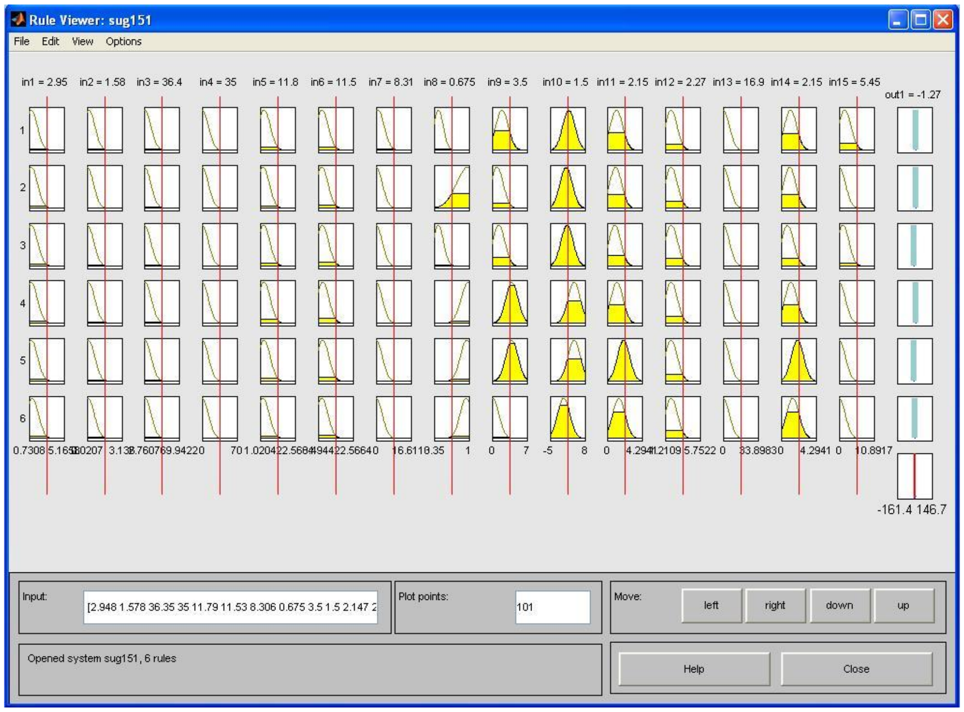
Weight: 1

FIS Name: sug151

Figure 6-8: Rule extracted from ANIFS system

The target/ANFIS output labels and prediction errors for each sample are shown in Figure 6-7. It can be seen in Figure 6-7 that the error distribution is quite random, which implies that a first-order Sugeno-type FIS may not be suitable for this problem. Compared with the OWFDT result, this classification results is slightly worse (64.54% vs. 79.28%). However, ANFIS converges much faster, due to its hybrid learning and its ability to construct reasonably good input MFs (Ozkan, 2006).

Figure 6-8 shows a view of the resulting rules from ANFIS. Six rules in total are extracted from the system. Each of these six rules consists of 15 antecedents (i.e. 15 attributes), which makes the rules difficult to interpret. Also note that the rule antecedent and consequent parts remain unchanged throughout the training process, as shown in Figure 6-8. It is also evident that no membership degrees are displayed. The consequent part of each rule represents a single MF (i.e. the number of rules is equal to the number of output MFs) with the same unit weights and there is no rule sharing in the ANFIS system. Thus, the ANFIS training adjusts parameters such as MFs and network weights instead of manipulating rules and network structures as some other systems such as EFuNN do in their training process.

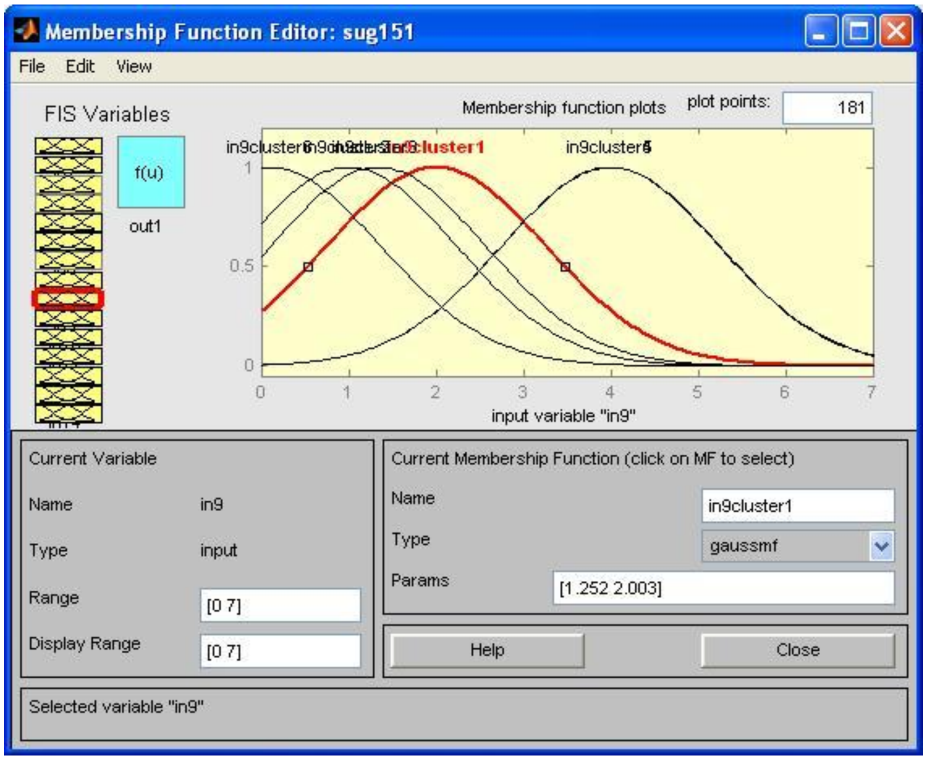


Input: [2.948 1.578 36.35 35 11.79 11.53 8.306 0.675 3.5 1.5 2.147 2

Plot points: 101

Opened system sug151, 6 rules

Figure 6-9: ANFIS rule viewer



plot points: 181

Current Variable
Name: in9
Type: input
Range: [0 7]
Display Range: [0 7]

Current Membership Function (click on MF to select)
Name: in9cluster1
Type: gaussmf
Params: [1.252 2.003]

Selected variable "in9"

Figure 6-10: Membership function for attribute 9

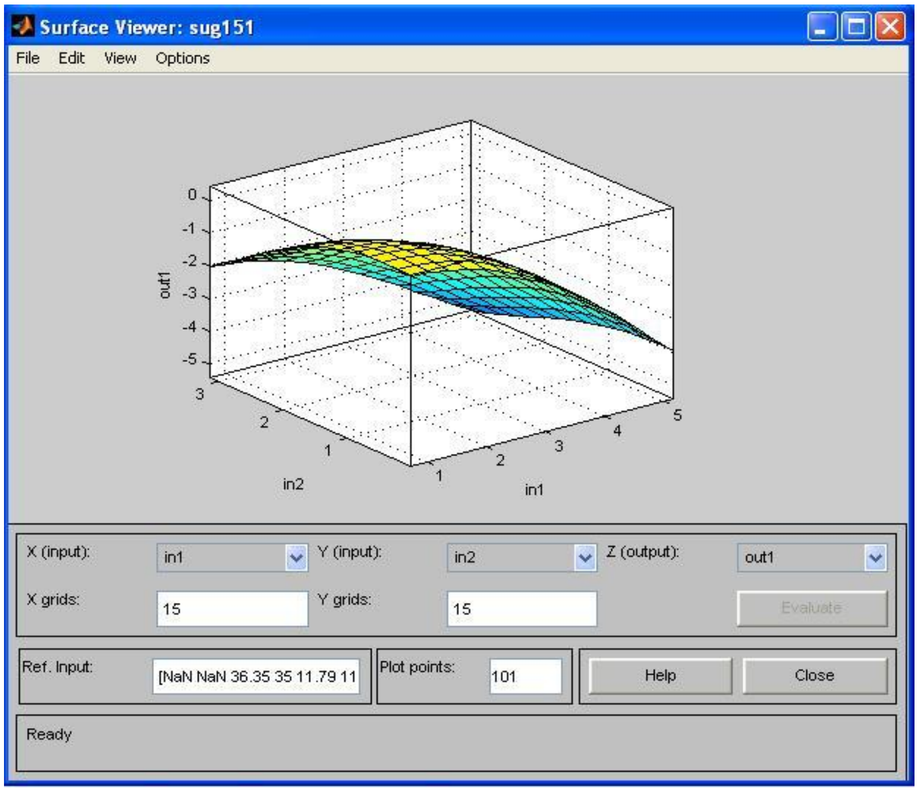


Figure 6-11: ANFIS rule surface viewer

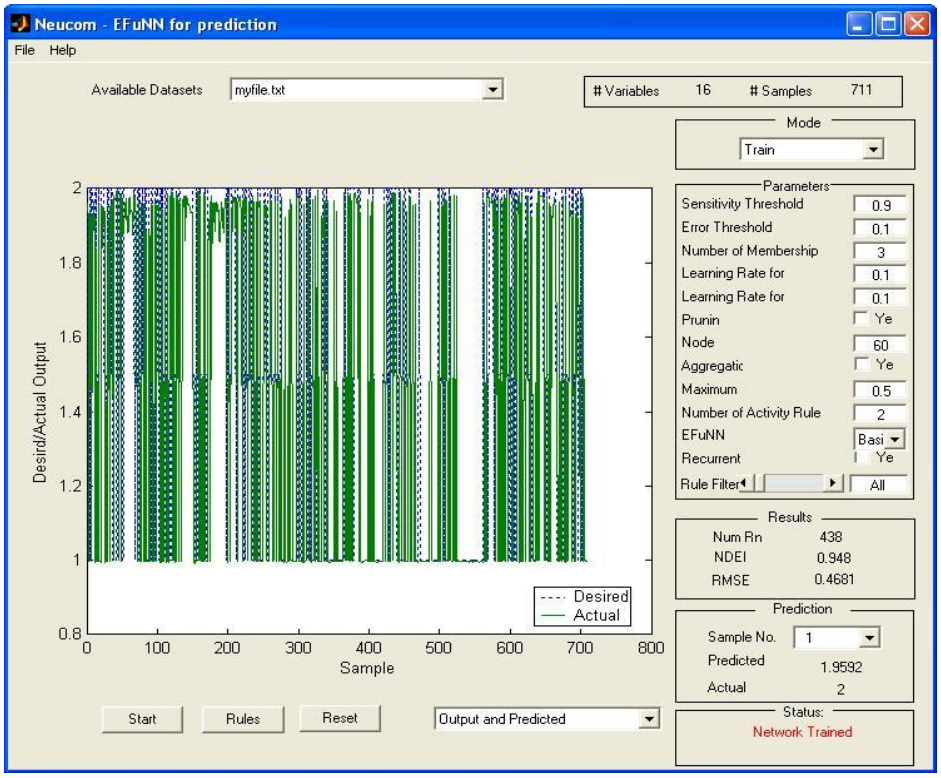
Figure 6-10 shows the membership functions associated with attribute 9 and Figure 6-11 shows the rule surface formed by the first two attributes. From these two figures, it can be seen that the concept representation rules learned by ANFIS are easier to understand (Boilot, et al., 2002), because the inputs to the ANFIS rule space are just attribute outputs. However, ANFIS cannot provide an insight into the data distribution and rule importance due to its fuzzy nature, as each input belongs to different sets to different degrees (Yang, 2009). On the other hand, OWFDT does assign a degree of importance to each rule by determining the degree of possibilities of data samples falling into all the possible classes.

Furthermore, Kasabov also points out that ANFIS has limited abilities for incremental, online learning (Kasabov, 2007) because of its structure that cannot adapt to the data at hand.

6.5.2 EFUNN

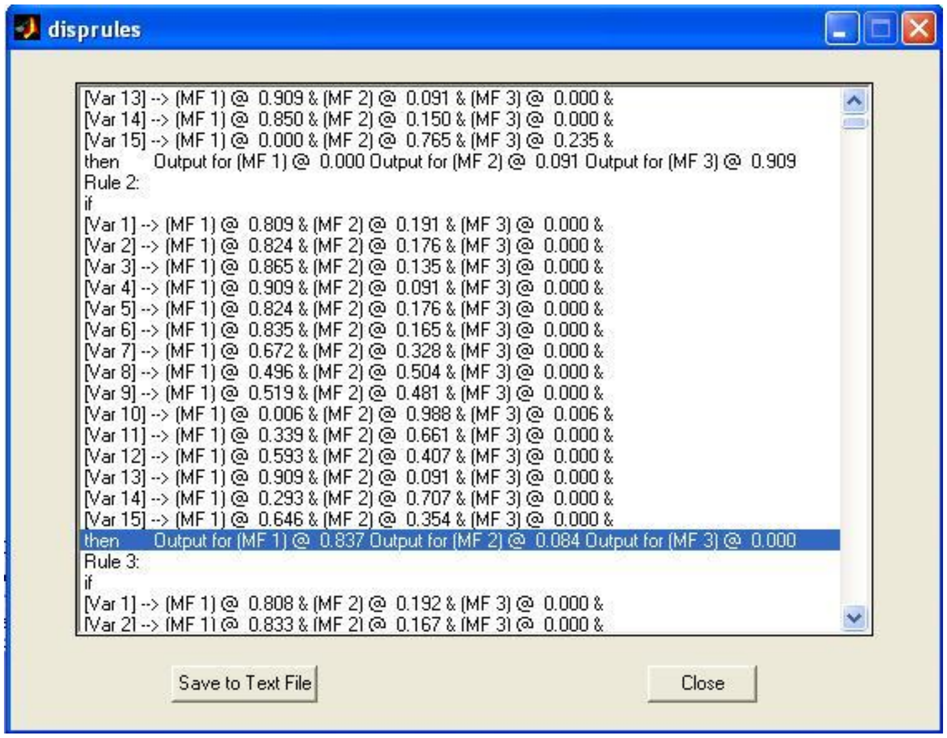
The EFuNN system achieved an RMSE of 0.3778 when it was applied to CLASH data. The system also produced 1293 rules and each of the rules has 15 antecedents. Due to its large size, it cannot be displayed here easily. Figure 6-12 shows the training and prediction results if one tenth of the data samples were exacted to feed the system. The training dataset consists of 711 samples and the training process produced 438 rules when it stopped at a RMSE of 0.4681. It is evident that too many rules affect the interpretability of the system.

Figure 6-13 shows the EFuNN rules which are quite different from the ANFIS rules as shown in Figure 6-8. The aim of EFuNN training is to find connection nodes that associate fuzzy inputs and outputs, which is different to the case in ANFIS where subtractive partitioning is applied and where training is performed mainly to adjust MF parameters. Thus, EFuNN rules show the membership degrees that each input/output belongs to as indicated in Figure 6-13. Take rule No. 2, which is highlighted in Figure 6-13, for an example. This rule shows that if attribute 1 belongs to its 1st/2nd/3rd MF to a degree of 0.809/0.191/0.000 and attribute 2 belongs to its 1st/2nd/3rd MF to a degree of 0.824/0.176/0.000 respectively etc., then the fuzzy output is [0.837 0.084 0]. Based on these fuzzy output values, aggregations can be performed to produce predicted output values.



```
Available Datasets   myfile.txt
# Variables   16     # Samples   711
Mode   Train
Parameters
Sensitivity Threshold        0.9
Error Threshold              0.1
Number of Membership         3
Learning Rate for            0.1
Learning Rate for            0.1
Prunin                       Ye
Node                         60
Aggregatic                   Ye
Maximum                      0.5
Number of Activity Rule      2
EFuNN                        Basi
Recurrent                    Ye
Rule Filter                  All
Results
Num Rn     438
NDEI       0.948
RMSE       0.4681
Prediction
Sample No.   1
Predicted    1.9592
Actual       2
Status:
Network Trained
```

Figure 6-12: EFuNN prediction results and targets



```
[Var 13] --> (MF 1) @  0.909 & (MF 2) @  0.091 & (MF 3) @  0.000 &
[Var 14] --> (MF 1) @  0.850 & (MF 2) @  0.150 & (MF 3) @  0.000 &
[Var 15] --> (MF 1) @  0.000 & (MF 2) @  0.765 & (MF 3) @  0.235 &
then      Output for (MF 1) @  0.000 Output for (MF 2) @  0.091 Output for (MF 3) @  0.909
Rule 2:
if
[Var 1] --> (MF 1) @  0.809 & (MF 2) @  0.191 & (MF 3) @  0.000 &
[Var 2] --> (MF 1) @  0.824 & (MF 2) @  0.176 & (MF 3) @  0.000 &
[Var 3] --> (MF 1) @  0.865 & (MF 2) @  0.135 & (MF 3) @  0.000 &
[Var 4] --> (MF 1) @  0.909 & (MF 2) @  0.091 & (MF 3) @  0.000 &
[Var 5] --> (MF 1) @  0.824 & (MF 2) @  0.176 & (MF 3) @  0.000 &
[Var 6] --> (MF 1) @  0.835 & (MF 2) @  0.165 & (MF 3) @  0.000 &
[Var 7] --> (MF 1) @  0.672 & (MF 2) @  0.328 & (MF 3) @  0.000 &
[Var 8] --> (MF 1) @  0.496 & (MF 2) @  0.504 & (MF 3) @  0.000 &
[Var 9] --> (MF 1) @  0.519 & (MF 2) @  0.481 & (MF 3) @  0.000 &
[Var 10] --> (MF 1) @  0.006 & (MF 2) @  0.988 & (MF 3) @  0.006 &
[Var 11] --> (MF 1) @  0.339 & (MF 2) @  0.661 & (MF 3) @  0.000 &
[Var 12] --> (MF 1) @  0.593 & (MF 2) @  0.407 & (MF 3) @  0.000 &
[Var 13] --> (MF 1) @  0.909 & (MF 2) @  0.091 & (MF 3) @  0.000 &
[Var 14] --> (MF 1) @  0.293 & (MF 2) @  0.707 & (MF 3) @  0.000 &
[Var 15] --> (MF 1) @  0.646 & (MF 2) @  0.354 & (MF 3) @  0.000 &
then      Output for (MF 1) @  0.837 Output for (MF 2) @  0.084 Output for (MF 3) @  0.000
Rule 3:
if
[Var 1] --> (MF 1) @  0.808 & (MF 2) @  0.192 & (MF 3) @  0.000 &
[Var 2] --> (MF 1) @  0.833 & (MF 2) @  0.167 & (MF 3) @  0.000 &
```

Figure 6-13: Rules exacted from EFuNN system

The membership functions in the EFuNN system do not change during the training process, but there are many parameters that need to be set before training, which highlights a potential important disadvantage for EFuNN. These parameters include sensitivity threshold, error threshold and the number of MF for each input variable and the learning rates *etc.* (Figure 6-12). Even though a trial and error approach is available and practical, when the dataset involves a large number of samples and features, determining the optimal parameters may be computationally expensive (Abraham & Nath, 2001).

6.5.3 Decision Tree

The principle of a Decision Tree (DT) is that it tries to split the high dimensional data into partitions, which are purer in terms of the class membership. An overly large tree will be grown using a criterion that generates optimal splits for the tree. To classify the new patterns, the DT starts at the top node and applies the rule. If the node satisfies the rule the tree takes the left path, and if not it takes the right path. Ultimately it reaches a terminal node that assigns an input pattern to one of the two categories. This large tree fits the training dataset very well, but does not generalize well to new patterns, so the rate at which new patterns are correctly classified is low with a misclassification error cost of 33.42%. For the CLASH data, a tree with 501 nodes was grown (it could not displayed here due to its large size).

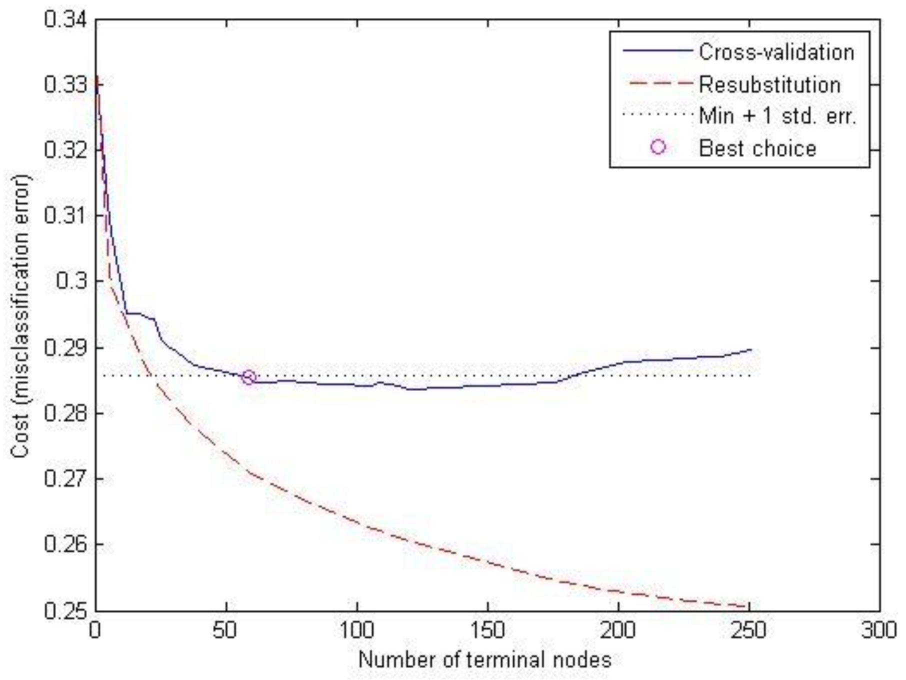


Figure 6-14: Estimated error for cross validation and resubstitution

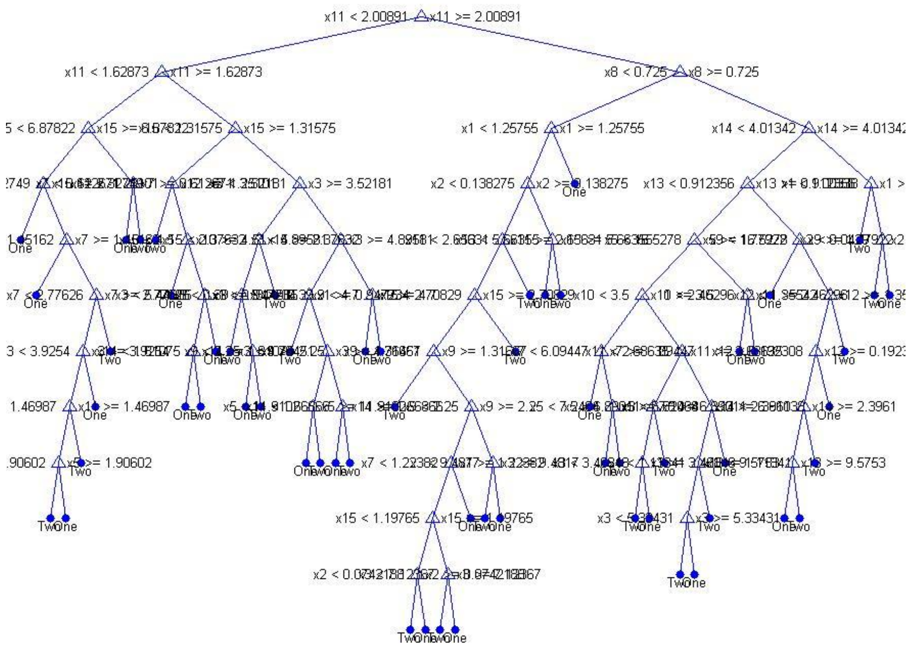


Figure 6-15: The pruned tree view

In order to find a smaller and simpler tree, a nested sequence of subtrees by pruning branches of the large tree were examined, which was based on the misclassification rate estimated by cross-validation or an independent test sample. Figure 6-14 shows tree test results using a ten folder cross validation in which a subset of 10% of the data was set aside for validation and the remaining 90% of the data was used for training. It indicated that the re-substitution error (the proportion of the original observations that were misclassified by various subsets of the original tree) decreased as the tree size grows. But the cross-validation results, which were used to estimate the true error for trees of various sizes, showed that beyond a certain point, increasing the tree size increases the error rate. As discussed section 2.3.5 in chapter Two, the simplest tree that is within one standard error of the minimum cost will be chosen as the pruned tree. Figure 6-14 shows the cutoff value which is equal to the minimum cost plus one standard error. The best level of pruning is the smallest tree under this cutoff. After setting the cutoff value, the pruned tree was obtained using the best level and the estimated misclassification cost was computed. Figure 6-15 displays the pruned classification tree. And for the CLASH data, the misclassification error is 28.60%.

6.5.4 Fuzzy ARTMAP

Fuzzy ARTMAP was trained using 90% percent of the CLASH dataset and tested using the other 10% of the data. It achieved an RMSE of 0.4849 with an accuracy of 76.51% for test dataset. The system also produced 131 committed coding nodes. Figure 6-16 shows the predicted and target class values and the prediction errors using 1% of the samples for display purpose. As can be seen from the figure, Fuzzy ARTMAP's predicted outputs are integer values. This is different from some other intelligent systems techniques, such as ANFIS and EFuNN which produce some non-integer and continuous values. This will also result in a relatively large value of RMSE even when the Fuzzy ARTMAP classifies most of the input samples correctly.

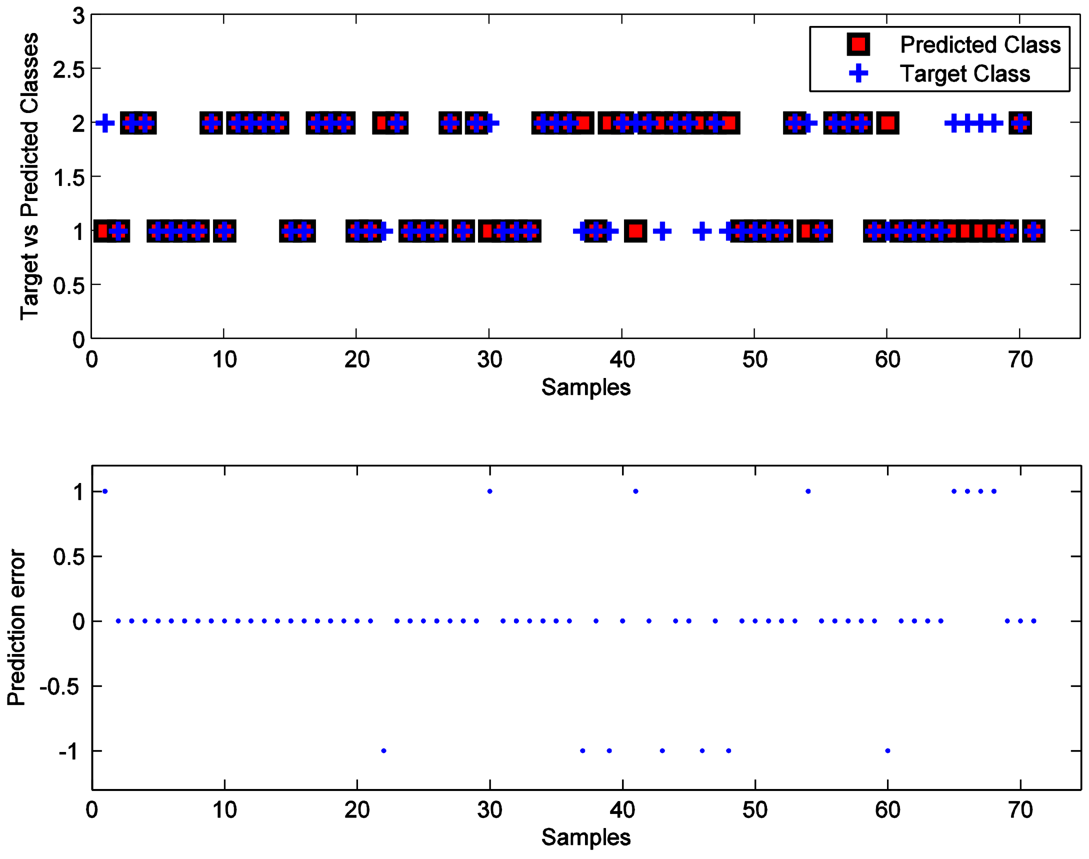


Figure 6-16: Fuzzy ARTMAP output results and prediction error

Fuzzy ARTMAP has the advantage of converging very quickly and requiring no fine tuning of parameters compared to OWFDT. And all the training information and the parameters are stored in the system. According to Mahadevan and Raghavendra, Fuzzy

ARTMAP system does not suffer from temporal instability during on-line training (Mahadevan & Raghavendra, 1997).

Its disadvantages are also quite obvious. Since Fuzzy ARTMAP only provides integer values as prediction results for the input samples, it can only be regarded as a predictor but not a generalizer.

When Carpenter *et al.* first proposed Fuzzy ARTMAP in 1992; they did not extract any rules from the training system. In order to increase its interpretability, by focusing on the clusters formed by committed nodes, some researchers have proposed methods in successive research to extract rules for Fuzzy ARTMAP systems (Andrés-Andrés, Gómez-Sánchez, & Bote-Lorenzo, 2005; Carpenter & Tan, 1995; Daxin, Yanheng, & Jian, 2006). But most of the research had encountered a bottleneck: there seems to be a tradeoff between the interpretability and the performance. Increasing the interpretability is always at the cost of damaging the system performance (Connolly, Granger, & Sabourin, 2008; Granger, Connolly, & Sabourin, 2008).

6.5.5 GNMM

First of all, in section 6.3, an MLP was trained using CLASH data with all available attributes using the LM algorithm. As a result, within 25 epochs it achieved an RMSE of 0.8674 with a regression value of 0.9363. Applying GNMM to the CLASH Data, the NN was trained for 200 iterations. By using the default cross folder validation, 90 percent of the dataset was used as training data and 10 percent of the data was used as validation data. The minimum RMSE value of 0.6952 was achieved at iteration 79. 2152 rules were extracted for training dataset and 496 rules were extracted using validation dataset. GNMM achieved an accuracy of 75.88% for the whole dataset.

Compared to an MLP, GNMM can reduce the instability of a NN structure caused by random weight initialization. By training GNMM in a number of iteration, it can find the minimum RMSE value which will help to achieve a higher accuracy. One obvious drawback is that GNMM is very computationally expensive. It took a CPU time of 12281.81 seconds to finish the simulation process.

Table 6-4: Comparison of different IST results for CLASH data

Methods	Accuracy	RMSE	Rule Numbers
OWFDT	79.28	0.4698	43
ANFIS	64.54	0.3324	6
EFuNN	74.40	0.3778	1293
DT	71.40	0.3581	59
GNMM	75.88	0.6952	2648
Fuzzy ARTMAP	76.51	0.4849	N/A

6.6 Conclusion

Table 6-4 shows a comparison of different IST results. Both OWFDT and Fuzzy ARTMAP produced discrete and integer values as outputs and this made the RMSE values slightly larger than of other techniques, which produced continuous values as output. But OWFDT's capability to have a better control on the splitting degree of the input features has resulted in less rules. This is in contrast to EFuNN which adopted a fuzzy-space-mapping approach. In addition to that, OWFDT's fuzzifying the input feature process has resulted in a higher accuracy compared to DT and other first-order fuzzy inference systems. OWFDT had demonstrated its ability to achieve a better classification accuracy whilst reducing the number of rules as shown in Table 6-4.

Table 6-5 summarizes the main features of all the intelligent systems techniques used in this chapter. Although a comprehensive study would be required to benchmark the performance of OWFDT against others, the current study within this chapter provides an overview of its outstanding characteristics. From Table 6-4 and Table 6-5 it is evident that compared with DT, OWFDT's fuzzifying input feature process can enhance its classification performance. Compared with FIS based systems such as ANFIS, which produces similar and low-interpretability rules, OWFDT can generate variable rules which can link them with membership functions. Compared to other NN based approaches such as EFuNN and Fuzzy ARTMAP, OWFDT presents the advantage of producing fewer rules.

One possible disadvantage is that when the algorithm tries to split the feature space into a smaller one, it may face the problem of the curse of dimensionality. It is very computationally expensive when the terminal nodes contain a pure feature.

Table 6-5: Summary of Features of OWFDT with other intelligent systems techniques

Technique	Core technique	System Input	Training method	Rule extraction
OWFDT	Back propagation	Fuzzified input according to membership function	Weight updated and gradient descent	Dividing fuzzified input space
ANFIS	Sugeno-type FIS	Fixed	Least square estimator and the gradient descent	Fuzzy rules, no rule sharing
GNMM	MLP	Fixed	Levenberg- Marquardt	Dividing input space
EFuNN	Mamdani-type FIS	Evolve over iterations	Hybrid unsupervised and supervised learning	Fuzzy rules, increase dramatically when more data presented
Decision Tree	Dichotomy	Original data	Probability theory	Dividing original data
Fuzzy ARTMAP	ART	Fuzzy inputs	Incremental supervised	Rule extraction based on committed nodes

References

Abraham, A., & Nath, B. (2001). A Neuro-Fuzzy Approach for Modelling Electricity Demand in Victoria. *Applied Soft Computing Journal, Elsevier Science, 1*(2), 127-138.

Andrés-Andrés, A., Gómez-Sánchez, E., & Bote-Lorenzo, M. L. (2005). Incremental Rule Pruning for Fuzzy ARTMAP Neural Network. In W. Duch, J. Kacprzyk, E. Oja & S. Zadrozny (Eds.), *Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005* (Vol. 3697, pp. 655-660): Springer Berlin / Heidelberg.

Boilot, P., Hines, E. L., Gardner, J. W., Pitt, R., John, S., Mitchell, J., et al. (2002). Classification of bacteria responsible for ENT and eye infections using the Cyranose system. *Sensors Journal, IEEE, 2*(3), 247-253.

Carpenter, G. A., & Tan, A.-H. (1995). Rule Extraction: From Neural Architecture to Symbolic Representation. *Connection Science, 7*(1), 3 - 27.

Connolly, J.-F., Granger, E., & Sabourin, R. (2008). Supervised Incremental Learning with the Fuzzy ARTMAP Neural Network. In L. Prevost, S. Marinai & F. Schwenker (Eds.), *Artificial Neural Networks in Pattern Recognition* (Vol. 5064, pp. 66-77): Springer Berlin / Heidelberg.

Cornelius, T. L. (1999). *Fuzzy Theory Systems: Techniques and Applications*: Academic Press, Inc.

Daxin, T., Yanheng, L., & Jian, W. (2006). *SLNN: A Neural Network for Fuzzy Neural Network's Structure Learning.* Paper presented at the Intelligent Systems Design and Applications, 2006. ISDA '06. Sixth International Conference on.

Geeraerts, J., Troch, P., De Rouck, J., Verhaeghe, H., & Bouma, J. J. (2007). Wave overtopping at coastal structures: prediction tools and related hazard analysis. *Journal of Cleaner Production, 15*(16), 1514-1521.

Granger, E., Connolly, J. F., & Sabourin, R. (2008). *A comparison of fuzzy ARTMAP and Gaussian ARTMAP neural networks for incremental learning.* Paper presented at the Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on.

Ingram, D. M., Gao, F., Causon, D. M., Mingham, C. G., & Troch, P. (2009). Numerical investigations of wave overtopping at coastal structures. *Costal Engineering, 56*, 190-202.

Karray, F. O., & Silva, C. W. D. (2004). *Soft Computing and Intelligent Systems Design: Theory, Tools and Applications*. New York: Addison Wesley.

Kasabov, N. (2007). *Evolving Connectionist Systems: The Knowledge Engineering Approach*. London: Springer.

Lykke Andersen, T., & Burcharth, H. F. (2009). Three-dimensional investigations of wave overtopping on rubble mound structures. *Coastal Engineering, 56*(2), 180-189.

Mahadevan, I., & Raghavendra, C. S. (1997). *Admission Control in ATM Networks using Fuzzy-ARTMAP.* Paper presented at the Proceedings of the International Workshop on Applications of Neural Networks, Melbourne, Australia.

Meer, V. d., W., J., Verhaeghe, H., & G.J.Steendam. (2005). *Database on wave overtopping at coastal structures, CLASH WP2 database*: Infram, Marknesse,The Netherlands.

Ozkan, C. (2006). *Surface Interpolation by Adaptive Neuro-fuzzy Inference System Based Local Ordinary Kriging*. Hyderabad, India: Springer Verlag.

TAW. (2002). *Technical report wave run-up and wave overtopping at dikes*: Technical Advisory Committee on Flood Defence, The Netherlands.

van der Meer, J. W., Verhaeghe, H., & Steendam, G. J. (2009). The new wave overtopping database for coastal structures. *Coastal Engineering, 56*(2), 108-120.

van Gent, M. R. A., van den Boogaard, H. F. P., Pozueta, B., & Medina, J. R. (2007). Neural network modelling of wave overtopping at coastal structures. *Coastal Engineering, 54*(8), 586-593.

Verhaeghe, H. (2005). *Neural Network Prediction ofWave Overtopping at Coastal Structures.* Universiteit Gent.

Verhaeghe, H., De Rouck, J., & van der Meer, J. (2008). Combined classifier-quantifier model: A 2-phases neural model for prediction of wave overtopping at coastal structures. *Coastal Engineering, 55*(5), 357-374.

Verhaeghe, H., Meer, V. d., & J.W., S., G.J. (2003). *Database on wave overtopping at coastal structures, CLASH WP2 internal report,* : Ghent University Belgium.

Verhaeghe, H., Meer, V. d., J.W., S., G.J., Besley, P., Franco, L., & van Gent, M. R. A. (2003). *Wave overtopping database as the starting point for a neural network* *prediction method.* Paper presented at the Proceedings of the International Conference on Coastal Structures, Portland, USA.

Yang, J. (2009). *Intelligent Data Mining using Artificial Neural Networks and Genetic Algorithms: Techniques and Applications.* University of Warwick.

Chapter VII: Conclusions and Suggestions for Future Work

This chapter summarises the main findings of this research, presents the conclusions and provides an outline of main results. It also includes suggestions for further research.

7.1 Overview of main research results

The main findings and the major original contributions to knowledge are presented as follows:

7.1.1 Optimized Weighted Fuzzy Decision Tree

The OWFDT technique proposed in this thesis has demonstrated its capacity to improve the prediction accuracy compared with that of Fuzzy ID3 and enhancing WFDT's ability to precisely classify an unseen instance through the weighted production rules. OWFDT has achieved its advantage through the following modifications as compared with traditional FDT:

(1) The Fuzzy C-Means algorithm makes the fuzzification of the input instances more efficient and effective when it is introduced to determine the possible cluster centers, which can be used to fuzzify the data in terms of each linguistic term;

(2) A momentum term to the gradient decent algorithm to train the neural network helps the learning process to avoid local minima. In addition, it can avoid oscillations or divergence which may be due to an inappropriately large learning rate or a low convergence rate due to a relatively small learning rate;

(3) By combining the leaf nodes weights with the leaf nodes certainty to infer the expected classification results, OWFDT proposed a new reasoning mechanism. This mechanism not only overcomes drawbacks in terms of not considering the leaf nodes' importance degree in inferring the results of the traditional FDT inference mechanism but also enhances WFDT's reasoning mechanism by combining each leaf node's degree of importance with leaf nodes (path) weights to infer an unknown test instance.

7.1.2 Case study results

7.1.2.1 Composite material failure mechanism classification

In chapter four, the signals recorded when composite materials encounter failure have been successfully used to classify the possible failure mechanisms. SOM and FCM were introduced to cluster the datasets into different groups representing different failure mechanisms. Five categories of failure mechanisms, mainly matrix cracking, fiber/matrix debonding, fiber/matrix pull out, fiber breakage and matrix breakage were detected.

Both FCM and SOM have resulted in similar cluster conclusions for seven datasets with a similarity ranging from 88.2% to 98.9%. Classifier designed using OWFDT also achieved better results (with the highest accuracy of 95.49% for 90° degree), which are 0.66% to 14.51% higher than traditional FDT.

7.1.2.2 Eye bacteria dataset

In chapter five, in order to classify six kinds of bacteria which are responsible for eye infection, MLP and DT were introduced to recognize the possible classes. Although a single MLP can classify the dataset, the accuracy was not high enough and the results were highly unstable and dependent on the network initialization. The decision tree constructed was able to discriminate different bacteria classes but with a high error cost of 20.56%. We have managed not only to enhance the accuracy but also stabilize the performance of MLP classifiers by introducing the bagging technique to MLP. Bagging MLP achieved accuracy as high as 100% and regression value for test dataset and validation dataset are above 0.97, which shows MLP outputs fit well with the targeted ones. By bagging the decision tree, we have successfully extracted eight most important features and reduced the dimensionality of the original data. In addition to that, the error cost of the classification tree has also reduced significantly to 9.44% making the decision tree more reliable.

7.1.2.3 OWFDT case study results

A total of six datasets were used in chapter three, four, five and six of the thesis to illustrate the implementation and demonstrate the usefulness of OWFDT. A summary of these case study datasets and results is shown in Table 7-1. Although these datasets belong to different categories, i.e. material, medical and civil engineering, they can all be regarded as prediction and classification.

Data I is concerned with the classification of iris species, namely setosa, versicolor, and virginica. Data II is concerned with breast cancer in order to predict whether it is benign or malignant. These two dataset are widely used as benchmarking study data for many newly-developed algorithms. Data III is concerned with Brain Computer Interface. Data IV is concerned with composite material failure mechanisms. Data V deals with the EN sensed eye bacteria data. In chapter six, a wave overtopping dataset is used to benchmark OWFDT with some existing intelligent systems techniques which bear some similarities to OWFDT. Although OWFDT was implemented to six datasets, the emphasis is different for the different datasets. For dataset I-V, OWFDT was used to illustrate the effectiveness and efficiency of OWFDT compared with traditional FDT and Weighted FDT in terms of the accuracy and the number of iterations used to train the neural networks. For dataset VI, emphasis was placed on OWFDT’s improvement of achieving a higher accuracy (see table 6-4 in chapter six) while producing less reasoning rules compared to ANFIS, EFuNN, Fuzzy ARTMAP, DT and GNMM.

7.2 Advantages and Disadvantages

By combing decision tree with fuzzy logic and neural networks, OWFDT is capable of not only classifying a dataset with a higher accuracy but also dealing with a dataset where the underlying relationship within the dataset is not understandable to human beings.

- OWFDT can overcome the drawbacks of fuzzy decision tree by taking the importance of every leaf node (rule) into account and assigning a weight factor to every rule. OWFDT is also able to reduce the average epochs of training process of weighted fuzzy decision tree by introducing the momentum term. In this way OWFDT has achieved higher classification or prediction accuracy over traditional fuzzy decision tree using a small number of average epochs.
- OWFDT can also extract rules which are friendlier to human beings than datasets themselves. It overcomes the nature of 'black box' of neural networks which don't provide any underlying knowledge and relationship between the physical meaning of the dataset and the actual output. The rules extracted using OWFDT can not only help us to understand the internal behavior of neural networks but also makes the classification results more understandable and transferable.

However, OWFDT also faces disadvantages based on the application throughout the thesis. An obvious disadvantage is that OWFDT is quite sensitive to noise which will pose obstacles for fuzzyfication of raw data. Furthermore, OWFDT may not be applicable to large categorical datasets since categorical data is difficult to be transferred to numerical data in a systematical way.

Table 7-1: A summary of case study data and results

Dataset		OWFDT results		Benchmarking literature	
Nature	Dimension (class x attribute x sample)	Classification results	Rule extraction	Methods	Results
I. Iris	3x4x150	96.00%	7	Fuzzy Decision Tree	90.67%
II. Wisconsin Breast Cancer	2x30x569	96.49%	11	Fuzzy Decision Tree	92.98%
III. Brain Computer Interface	2x12x378	92.00%	3	Fuzzy Decision Tree	89.00%
IV. Composite Material	(3-5)x4x(534-7052)	See table 2 in chapter 3	7-59	Fuzzy Decision Tree	See table 2 in chapter 3
V. Eye Bacteria classification	6x32x180	85.56%	41	Integer based GA using PNN (Boilot, 2003)	82.00%
VI. Wave Overtopping	2x15x7107	79.28%	43	ANFIS, EFuNN, Fuzzy ARTMAP, DT, GNMM	See table 4 in chapter 6

7.3 Further work

In OWFDT, Fuzzy C-Means was introduced to fuzzify all the features of raw data using the cluster centers and maximum and minimum values of each feature. One possible improvement is that we could introduce more partition methods corresponding to each feature so that each feature can have a personalized partition method (S. Mitra, Konwar, & Pal, 2002). Although this may increase the computational complexity, a higher accuracy may overtake this disadvantage when the accuracy of classification outweighs its computational cost.

As a data driven method, OWFDT also relies on the quality of the data. Successful implementation of OWFDT suggests that the data should not only be cleaned off errors and redundancy, but also be organized in a fashion that makes sense in the context of the application(Yang, 2009). Uncertainty, vagueness, and incompleteness of raw data may lead to problems for knowledge acquisition (Sushmita Mitra & Acharya, 2003). Thus, future works may also include applications of OWFDT to some incomplete and highly noisy data.

Although OWFDT has been able to solve many data mining applications as discussed in this chapter, a systematic way to integrate it into the process of KDD has to be devised so that OWFDT can be able to condense and transform the original data and present the knowledge in a more friendly way to final users.

References

Boilot, P. (2003). *Novell Intelligent data processing techniques for electronic nose: Feature Selection and Neuro-fuzzy Knowledge base.* University of Warwick.

Mitra, S., & Acharya, T. (2003). *Data Mining: Multimedia, Soft Computing, and Bioinformatics*. Hoboken, NJ: John Wiley.

Mitra, S., Konwar, K. M., & Pal, S. K. (2002). Fuzzy decision tree, linguistic rules and fuzzy knowledge-based network: generation and evaluation. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 32*(4), 328-339.

Yang, J. (2009). *Intelligent Data Mining using Artificial Neural Networks and Genetic Algorithms: Techniques and Applications.* University of Warwick.