PATTERN CLASSIFICATION BY AN INCREMENTAL LEARNING FUZZY NEURAL NETWORK

By

PHAYUNG MEESAD

Bachelor of Science

King Mongkut's Institute of Technology North Bangkok

Bangkok, Thailand

1994

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 1998

PATTERN CLASSIFICATION BY

AN INCREMENTAL LEARNING

FUZZY NEURAL NETWORK

Thesis Approved:

[illegible]

Thesis Adviser

R. Ramakumar

[illegible]

Wayne B. Powell

Dean of the Graduate College

PREFACE

To detect and identify defects in machine condition health monitoring, classical neural classifiers, such as Multilayer Perceptron (MLP) neural networks, are proposed to supervise the monitored system. A drawback of classical neural classifiers, off-line and iterative learning algorithms, is a long training time. In addition, they are often stuck at local minima, unable to achieve the optimum solution. Furthermore, in an operating mode, it is possible that new faults are developing while a monitored system is running. These new classes of defects need to be instantly detected and distinguished from those that have been trained to the classifier. Those classical neural classifiers need to be retrained by both old and new patterns in order to learn new patterns without forgetting the learned patterns. Conventional classifiers cannot detect and learn the new fault types on-line real-time.

Using incremental learning algorithms in the monitoring system it is possible to detect those new defects of machine conditions with the system operating while maintaining old knowledge. Inspired by the promising properties of an incremental learning algorithm named Fuzzy ARTMAP Neural Network, a new algorithm suitable for pattern classification based on fuzzy neural networks called an Incremental Learning Fuzzy Neuron Network (ILFN) is developed. The ILFN uses Gaussian neurons to represent the distributions of the input space, while the fuzzy ARTMAP neural network uses hyperboxes. The ILFN employs a hybrid supervised and unsupervised learning scheme to generate its prototypes. The network is a self-organized classifier with the

capability of adaptive learning of new information without forgetting old knowledge. The classifier can detect new classes of patterns and update its parameters while in an operating mode. Moreover, it is an on-line (real-time) and fast learning algorithm without knowing *a priori* information. In addition, it has the capability to make soft (fuzzy) and hard (crisp) decisions, and it is able to classify both linear separable and non-linear separable problems.

To prove the concept, simulations have been performed with the vibration data known as the Westland Data Set. This data set was obtained from the Internet at http://wisdom.arl.psu.edu/Westland/ collected from U.S. Navy CH-46E helicopters maintained by Applied Research Laboratory (ARL) at Penn State University. Using a simple Fast Fourier Transform (FFT) technique for the feature extraction part, the network, capable of one-pass, on-line, and incremental learning performed quite well. Training by various torque levels, the network achieved 100% correct prediction for the same torque level of testing data. Furthermore, the classification performance of the network has been tested using other benchmark data, such as the Fisher's Iris data, the two-spiral problem, and a vowel data set. Comparison studies among other well-known classifiers were preformed. The ILFN was found competitive with or even superior to many classifiers.

ACKNOWLEDGMENTS

I wish to express my sincere gratefulness to my academic advisor, Dr. Gary G. Yen, for his understanding guidance, productive instruction, brainstorming, and companionship. My genuine appreciation draws to my other committee member Dr. Rama G. Ramakumar and Dr. Carl D. Latino, whose direction, support, reassurance, and friendship are also worthwhile.

I would like to thank the Intelligent System and Control Laboratory at Oklahoma State University for supporting resources. Furthermore, I would also like to thank Kuo-Chung Lin, Liang-Wei Ho, Wei Feng, and other people in the Intelligent System and Control Laboratory for their discussions and recommendations.

I would like to thank Jean Hassen, Matthew and Porntip Baysden for their help in grammatical checking and proof reading. Moreover, I would also like to give my special gratitude to my wife and my daughter, Pongpun and Natchaporn, for their long lasting sedulousness and solace through times of dilemma, and for their love and understanding throughout this investigation. My gratitude also goes to my mother and my sisters for their devotion, encouragement, and tolerance.

Finally, I would like to thank the Thai Government for supporting my matriculation during these two and a half years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Machine Health Monitoring Overview

Machine condition health monitoring and fault diagnosis are critical issues to be addressed in the competitive world of manufacturing. Increased product quantity and improved product quality result when the production speed of the industrial machine is increased, and the downtime due to system failures is decreased. Machine condition health monitoring and fault diagnosis are used to detect and distinguish faults occurring in machinery so that it is possible to perform condition-based maintenance before catastrophic failures. Moreover, it decreases operation and maintenance costs and prolongs the service life of machinery [6, 50, 95, 101, 119].

To provide predictive and preventive maintenance plans, traditional health monitoring techniques are based upon conventional nondestructive testing and evaluation (NDT/E) such as fluoroscopy, radiography [109], ultrasonic [28, 114], acoustic emission [128], optical scan [55], thermal inspection [45], current test [32, 44], and magnetic analysis [43, 85]. Usually the conservative NDT/E methodologies are local in nature, passive and labor intensive. The prototype instruments that are developed are heavy, expensive, and fault-prone. Thus, it is difficult to implement these methods in a transportable, on-board, automatic, real-time, and global health assessment tool.

Three main techniques are used in machine condition health monitoring and fault detection and isolation: experimentally-based techniques, model-based techniques, and model-free techniques.

In the experimentally-based approaches, an expert is essential for comparing the measurements of vibration signatures of a monitored system with some known fault patterns. Modal analysis regularly used in the experimentally-based methods include natural frequencies, damping ratios, mode shapes, and curvature shapes [2, 93, 99]. The success of the experimentally-based techniques relies on *a priori* collection of fault patterns and proficient experts.

On the other hand, the model-based approaches have rigorous mathematical foundation in detecting machine defects. Examples of the model-based method are finite element model [83, 135], experimental model analysis [110, 121], periodic time-varying autoregressive models [85], state observer theory [56], severity based updates [76], and eigenstructure assignments [70, 111]. These methods assume an accurate mathematical model before using as a tool for detecting machine damages. Their performance depends on a precise numerical representation of the monitored structure.

The third type of machine health monitoring techniques stems from recently emerged computational intelligence. These "model-free approaches" include expert system [66], neural network [7, 86], fuzzy logic, and fuzzy neural networks [24]. These methods offer the potential of real-time decision-making via the use of effective learning and evolution algorithms [39].

1.2 Concept of Machine Health Monitoring

Figure 1.1 shows a simple idea of machine health monitoring systems. A health monitoring system consists of 1) the system to be monitored; 2) sensors; 3) signal processing; 4) feature extraction; 5) pattern classification or fault identification; 6) human operators; and 7) condition-based maintenance process. An optional, automatic control block may be subscribed to the system to instantly correct the problems or shut down the system before serious damage occurs.

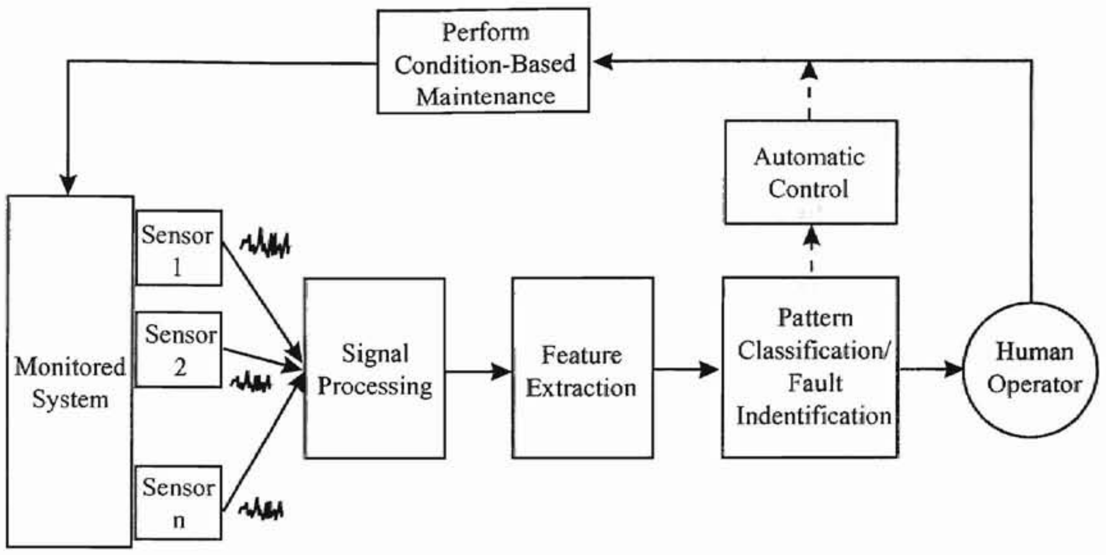


Figure 1.1 Conceptual Model of Machine Health Monitoring Systems

The monitoring system can be established in a bottoms-up hierarchy. The monitored system includes materials (e.g. steel, composites), elements (e.g. bearing balls/rollers, gear tooth), components involving the interaction of elements in the failure initiation and progression process (e.g., ball/roller bearings, gear meshes), subsystems (e.g., transmission), systems (e.g., power train), and platform (e.g., ship, helicopter). Sensors or transducers are used to sense the physical characteristics of the monitored systems. The physical characteristics such as vibration signals and acoustic emissions are the indications of system condition. From the sensors, digital signal processing and feature extraction are used to preprocess and reduce the dimension of data in order to obtain patterns containing enough information to discriminate in a lower dimension.

Next, pattern classification classifies and identifies the types of fault conditions. After the faults are identified, either human operators or automatic control systems feedback information to condition-based maintenance. The monitored system is then prescribed maintenance actions before severe damages occur [50].

Pattern classification or fault identification is a key component of condition health monitoring systems. Its function is to identify fault types induced from the monitored systems. Service and maintenance can be promptly and correctly performed if the pattern classifier makes an accurate decision. This thesis focuses on the investigation of a new methodology for pattern classification specifically suitable for machine condition health monitoring.

1.3 Motivation of the Research

In machine health monitoring systems, a vibration signal is one of the most common tools for detecting defects using pattern classification techniques. While operating, mechanical components generate vibration signals that contain information about the state of the machine [97]. Vibration data provide effective information for detecting and diagnosing some of the incipient failures of machines and equipment. The input data are entered into a classifier which is a component of a health monitoring system. With pattern classification techniques, signatures can be extracted from the vibration data that contain information about machine defects and their causes [119]. With the accurate decision of the classifier in a monitoring system, machine maintenance can be performed before catastrophic failures occur.

Vibration monitoring is based on the principle that components in engineering systems and plants produce vibrations during operation. When a machine is operating properly, vibration levels are generally small and constant. However, when faults develop which lead to variations of process dynamics, the vibration signatures (i.e., power spectrum density, natural frequency, and mode shape) also change [1, 11]. To detect these changes, classical off-line iterative learning classifiers are proposed to supervise the monitored system. These classifiers have a drawback in that they generally require a long training time. In addition, they are often stuck at local minima, unable to achieve the optimum solution.

Furthermore, in an operating mode, it is possible that new faults are evolving while a monitored system is running. These faults are different from those that have been trained to the classifier. These new classes of defects need to be promptly detected and distinguished from those that have been trained to the classifier. Conventional neural classifiers need to be retrained by both old and new data in order to learn new information while remembering existing information [112]. Moreover, the monitored system may generate one fault or multiple faults (more than one fault). A decision for these multiple defects is needed in order to perform correct maintenance. However, with a crisp decision, the traditional classifiers are able to detect one and only one fault.

1.4 Objectives of the Thesis

The primary objective of this thesis is to develop a methodology of pattern classification for condition health monitoring systems. This new classifier is called an "Incremental Learning Fuzzy Neural Network" (ILFN) implementing a neural network theory and the fuzzy set theory. The classifier is considered to have the following features.

1) *A hybrid supervised and unsupervised learning algorithm*: A supervised learning algorithm is used in the training phase where the corresponding targets are known. On the other hand, in the operating phase where the corresponding target is not known, an unsupervised learning algorithm is used.

2) *Fast, on-line, one-pass, incremental learning without local minimum problems*:

Many well-known neural networks and conventional pattern classification techniques use "off-line" learning which assumes all training patterns and their targets are known. On the other hand, for "on-line" learning only one training pattern and its target are needed at a time. Thus on-line learning requires less memory than off-line learning does. Off-line learning tends to use longer training time.

A "one-pass" learning algorithm, where training patterns are presented to the classifier only one time instead of many times, is preferred; however, in one-pass, the classifier should use as little computation time as possible.

"Incremental learning," the capability of learning new classes and quickly refining existing classes without forgetting learned information, is a very important concept of pattern classification. With incremental learning, classifiers learn new information without forgetting old information.

A gradient learning algorithm often has a problem of trapping at local minima; moreover, it needs an iterative presentation of data thus leading to long training times. So this new classifier avoids using the gradient learning procedure.

3) *Ability to detect new classes and label them differently from the existing corresponding targets*: In some condition health monitored systems, such as vibration monitoring systems, new patterns may be generated while the systems are operating. These new patterns need to be detected and learned by the classifiers in order to prescribe correct maintenance actions. After training, traditional classifiers cannot detect the difference between the learned fault patterns and unseen fault patterns. They can identify the new patterns only to the closest learned patterns even when they are significantly different. This may lead to misunderstanding and incorrect service.

4) *Ability to build decision regions that separate nonlinear separable problems*: Many neural classifiers have overcome the nonlinear separable classes. This new classifier should also provide the ability to build the decision boundaries to separate both linear and nonlinear separable classes.

5) *Ability to make decision boundaries of all overlapping classes*: Bayesian classifiers are generally used to classify overlapping classes; however, constructing the Bayesian classifiers requires knowledge of the probability density function for classes. Unfortunately, for on-line incremental learning classifiers, the probability density function for each class is unavailable beforehand. The classifiers should be able to find the probability density function or their equivalents on the fly. If the input class patterns are overlapped, a classifier should make a decision equivalent to a Bayesian classifier [112].

6) *A nonparametric classifier*: Parametric classifiers need *a priori* information about the probability density functions of pattern classes; on the other hand, nonparametric classifiers do not have *a priori* information available [112].

7) *Ability to provide both soft and hard classification decisions*: A "hard" decision means that a given pattern either belongs to or does not belong to class prototypes. On the other hand, a "soft" decision allows a given pattern belonging to more than one class prototype with different membership grades [112]. It is possible to detect multiple defects in monitored systems if a soft decision is used.

8) *Few tuning parameters*: Tuning parameters are used for controlling a system and there should be as few parameters as possible to tune in the system [112].

1.5 Organization of the Thesis

For the completeness of the presentation, the remainder of this thesis is organized as follows: Chapter 2 provides a literature review for the concept of pattern classification, neural networks, fuzzy set theory, and fuzzy-neural networks used for pattern classification problems. Chapter 3 introduces the proposed network architecture and the classification algorithm of the ILFN classifier. Chapter 4 shows the simulation results and comparisons to some existing classifiers on benchmark problems. Chapter 5 provides the conclusion of the research and possible future work.

CHAPTER II

LITERATURE REVIEW

2.1 Introduction

Pattern classification forms a fundamental solution to different problems in real world applications. The function of pattern classification is to categorize an unknown pattern into a distinct class based upon a suitable similarity measure. Thus, similar patterns are designated into the same classes while dissimilar patterns are classified into different classes.

Engineers and scientists have developed various methodologies to deal with classification problems. A large number of classification algorithms have been proposed to deal with classification problems. Statistical pattern classification is a traditional technique for classification problems [54, 73]. This classical classification technique makes use of statistical decision theory to classify patterns. Various researchers scrutinized parametric Bayesian classifiers [34] assuming that the forms of input distributions are known. The parameters of distributions are computed using all training data. The training data are usually assumed to be Gaussian when using Bayesian classifiers. Because of their simplicity, they are still widely used [4, 41, 80, 94].

Automatic pattern classification has been highly considered by scientists and engineers from different fields. Many researchers in the area of pattern classification have paid attention to neural network classifiers because of the capability of model-free and trainable systems, parallel computation, and noise tolerance of neural networks. These properties of artificial neural networks inspire the researchers to study neural network applications to deal with pattern classification problems. Neural networks with the abilities of real-time learning, parallel computation, and self-organizing make pattern classification more suitable to handle complex classification problems through their learning and generalization abilities [14, 86, 120, 132].

In addition, fuzzy set theory [134] has been extensively applied to pattern classifications. Fuzzy set theory supports pattern classification by dealing with inexact rather than exact problems. Fuzzy systems perform well on uncertain information, very similar to the way human reasoning does. The human brain performs very well even in imprecise circumstances. In the real world, most situations are fuzzy rather than crisp. Moreover, the information in pattern classification problems is imprecise rather than precise in nature, and fuzzy set theory allows us to properly model this vague information [15, 35, 40, 82].

The integration of neural networks and fuzzy sets is also an active area for pattern classification problems. A growing number of researchers have designed and examined various forms of fuzzy-neurons and neurofuzzy networks. The idea is to merge the capabilities of model-free and trainable systems, parallel computation, and noise tolerance of neural networks and the ability of dealing with imprecise situations of the fuzzy set theory. The integration of neural networks and the fuzzy set theory results in a classifier that has useful properties of both neural networks and fuzzy sets. The combination of neural networks and fuzzy sets forms a synergetic network that handles pattern classification problems very effectively and efficiently [24, 57, 77, 107, 108].

The following section discusses the basic idea of pattern classification. Some pattern classifiers developed from neural networks, fuzzy sets, and fuzzy-neural networks are also briefly discussed for the sake of completeness of the presentation.

2.2 Review of Pattern Classification Concept

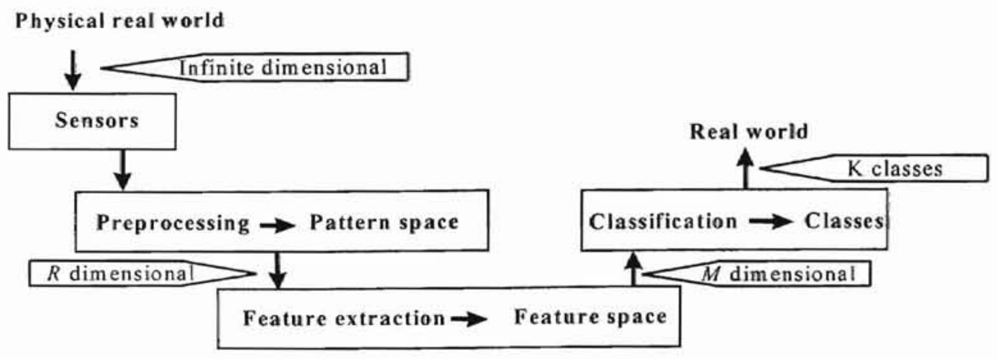


Figure 2.1 The Conceptualized Pattern Classification Problem

Figure 2.1 illustrates the framework of the pattern classification problem. The physical real world is sensed by a transducer system that feeds its data into the pattern space after a preprocessing procedure. The physical real world, or sensed system, can be characterized by a continuum of parameters that are basically infinite in dimensionality. Transducers are used to transform signals from real environment to the pattern vector space with the dimensionality of *R*, typically a large value. Then a feature extractor is employed to reduce the dimension from *R* to a much smaller value, *M*, while still preserving the discriminatory features for classification expectation. Using an *M*-dimension feature space, a classifier performs much faster than using an *R*-dimension pattern space. Finally, in the classification space, one of *K* classes is chosen for a given input pattern [3].

The data that will be classified are presented into pattern classifiers by sets of measurements. Each measurement associates an axis in a multidimensional space called "hyperspace." Figure 2.2 shows a two-dimensional space with three classes of patterns. Figure 2.3 illustrates a linear separable problem in which a line exists to separate the two classes. Figure 2.4 demonstrates a non-linear separable problem where a straight line cannot separate the two classes. A non-linear decision boundary is needed to separate this problem. An overlapping class is depicted in Figure 2.5. Neither a linear nor a non-linear boundary can separate this problem. However, the decision can be made by using "Bayes strategy" to reduce misclassification for this problem.

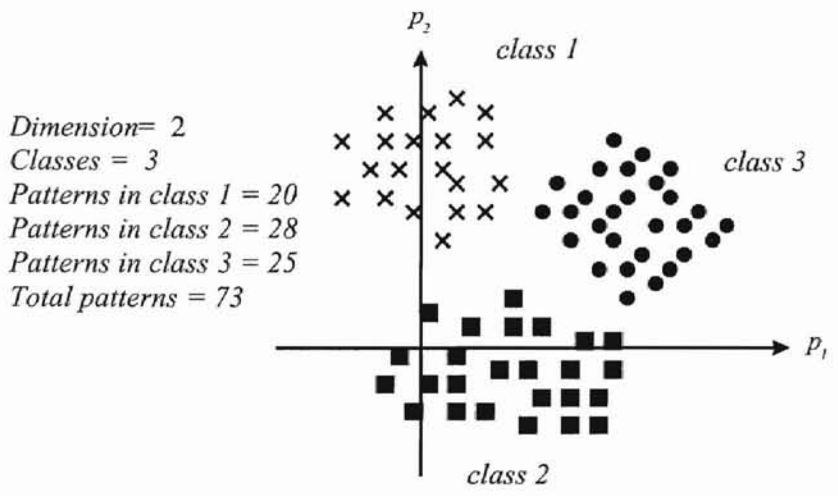


Figure 2.2 Example of a 2-Dimensional Vector Pattern Space

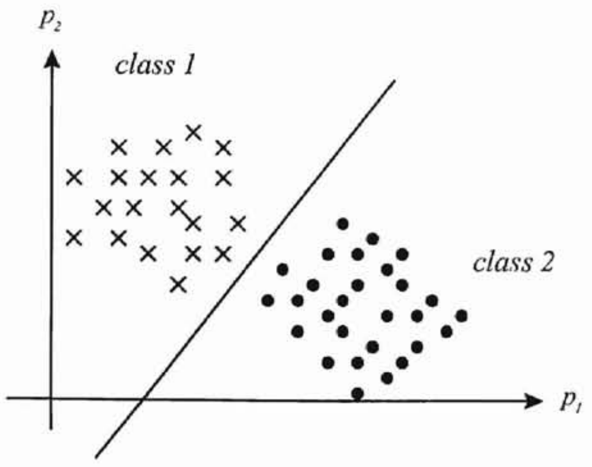


Figure 2.3 A Linearly Separable Problem

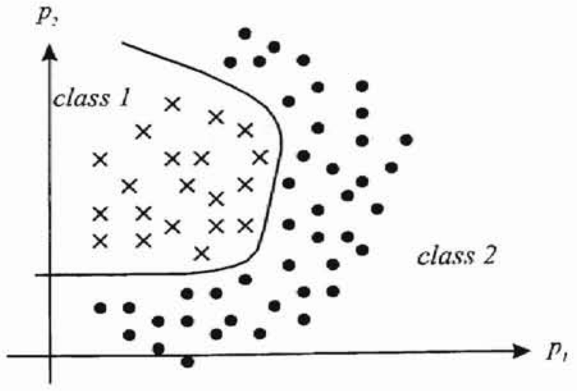


Figure 2.4 A Nonlinearly Separable Problem

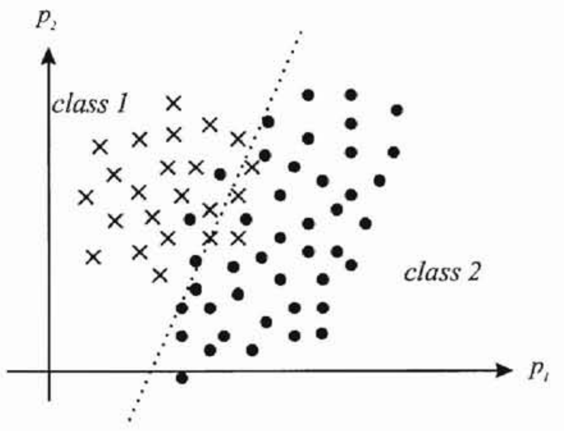


Figure 2.5 An Overlapping Problem

2.3 Neural Networks for Pattern Classification

An artificial neural network is a data processing system consisting of a massive number of simple and highly interconnected processing units operating in a parallel manner. The networks are inspired by the structure and the function of the human brain. The characteristics of an artificial neural network are model-free and trainable systems with parallel computation. These properties are considered as benefits to many applications in the real world, including pattern classification problems.

As a complement to the statistical pattern classification, scientists and engineers from different fields have considered neural networks for pattern classification. Neural network technologies make pattern classification capable of parallel computation, self-organization, and self-adjusting parameters.

Neural networks applied to pattern classification have two main types: a supervised learning and unsupervised learning (clustering) algorithm. The use of the supervised neural network as a classifier assumes the input and the corresponding target pairs are known. This approach assumes that appropriate input features have been chosen and that the training data are representative of all the problem conditions. Some examples of supervised learning networks are Multilayer Perceptron Network (MLP) trained by the Backpropagation algorithm (BP) [104-106], Probabilistic Neural Network (PNN) [115-118], Learning Vector Quantization Network (LVQ) [71], and Radial Basis Function Network (RBFN) [12, 52, 60, 91].

On the contrary, in unsupervised neural networks, the input does not have a corresponding target. Since there are no target outputs available, the network distinguishes the input data into a number of clusters and selects which features are important. The system learns to categorize the input patterns into a finite number of classes using some similarity measurements. The two most-used unsupervised neural networks are Adaptive Resonance Theory Networks (ART) [17] and Self-Organizing Maps (SOM) [71].

The MLP neural network trained by the backpropagation algorithm has been a good candidate for pattern classification problems. The MLP is a fully connected feedforward network with sigmoidal activation functions. There are many developed algorithms that are used to train the network such as the steepest descent, Newton's methods [8], conjugate gradient [23], and Levenberg-Marquardt algorithm [48]. Rumelhart and McClelland in [104-106] present an extensive detail of the MLP network. Two major drawbacks of steepest descent backpropagation algorithm are a long training time and the fact that there is no guarantee of convergence to a global minimum, especially when complex decision boundaries are required and networks have more hidden layers [31, 80]. Levenberg-Marquardt backpropagation is the fastest algorithm for training the MLP network but it requires a considerable amount of memory [48]. The architecture of the MLP network is shown in Figure 2.6.

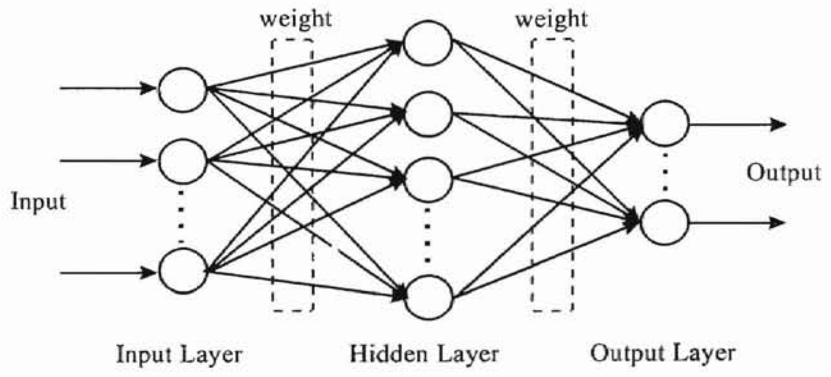


Figure 2.6 The MLP Neural Network

The Probabilistic Neural Network (PNN), developed by Donald Specht in 1988, is a useful methodology for solving pattern classification problems. Its decision boundaries are formed by conditional probability density functions (PDF). The network is able to form complex nonlinear decision boundaries created by the Bayes strategy when given enough examples. The training speed of the PNN is faster than the Backpropagation Neural Network (BPNN) to achieve the same level of generalization. On-line learning is another advantage of the PNN; thus it is suitable to use the PNN for real-time applications. However, the PNN uses extensive memory requiring one neuron for each training pattern. Henceforth, a lot of researchers have proposed various remedies to solve the memory problem, such as using clustering techniques to implement a cluster center which represents a prototype of training patterns [115-118]. Figure 2.7 shows the architecture of the PNN.

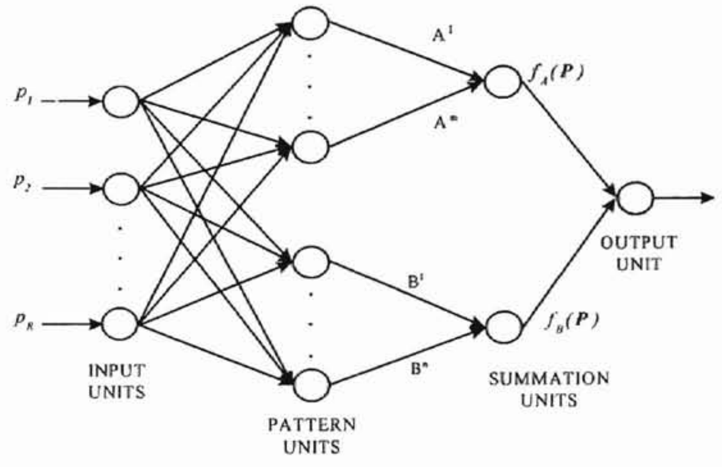


Figure 2.7 The PNN Network Architecture

Another good candidate for pattern classification is the Radial Basis Function Network (RBFN) [5, 12, 52, 84, 90, 60]. The RBFN is functionally equivalent to a fuzzy inference system [51, 63]. The network is a feedforward network consisting of three layers: an input layer, a hidden layer, and an output layer. Each neuron of the input layer connects to each element of an input vector. Neurons of the input layer are fully connected to neurons of the hidden layer via weights that represent the centers of radial basis functions in the hidden layer. The hidden layer has kernel functions (activation functions), usually Gaussian type, which are centered on the mean vectors of clusters or prototypes in the input space.

Training of the RBF network can proceed in two steps: first cluster the training patterns to a reasonable number of groups using SOM clustering [71], k-means clustering

[91], a successive approximation method [79], or APC-III algorithm [59]. After the training of the hidden layer, the output layer is trained by gradient descent method or least mean square error method [30]. It is worth noting that either APC-III or the successive approximation method is an incremental learning method (meaning that it learns new information without retraining old information). It can cluster input patterns within only a single pass through all patterns. A variety of techniques for training radial basis function networks are discussed in the literature [26, 58, 60, 65, 67, 72, 75, 88, 91]. In general, RBF networks require an order of magnitude less in training time compared to the backpropagation algorithm [93]. Moreover, their functions can be interpreted equivalent to a fuzzy inference system [63]. The architecture of the radial basis networks is shown in Figure 2.8. A drawback of the RBF networks is that they require off-line training that assumes knowing all the inputs and the corresponding targets.

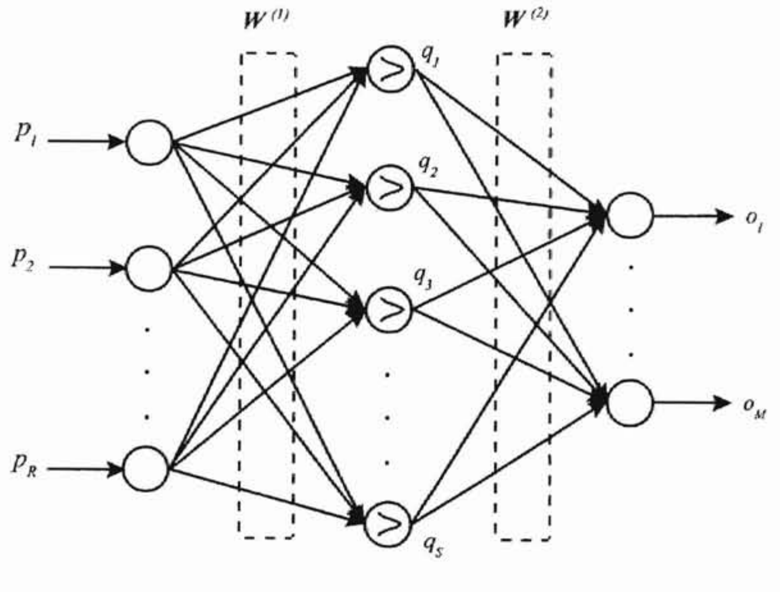


Figure 2.8 Radial Basis Function Neural Network

The Self-Organizing Map (SOM) neural network is an unsupervised learning algorithm that is very effective for pattern classification problems. The SOM network is usually composed of an input layer and an *M*-dimensional Kohonen or competitive layer. Typically the Kohonen layer is a two-dimensional layer. The weight vector is the same as the dimension of the input feature vectors. The weight vectors are randomly initialized in the feature space at the first stage. Then, the network determines the wining neuron for a given input vector. Next, all neurons within a certain neighborhood of the winning neuron are updated moving toward the input. The moving step is controlled by the learning rate [71]. One drawback of the SOM network is that it needs to know the number of clusters in advance. For some applications, it may not be acceptable that the number of clusters is not known. In addition, the choice of learning rate forces a trade-off between the speed of learning and the stability of the final weight vectors. Moreover, the SOM network needs iterative presentations of input patterns for learning.

A generalization of the SOM network, namely the Learning Vector Quantization (LVQ) neural network, has been extensively used for pattern classification problems. The LVQ network uses both an unsupervised and supervised learning algorithm. The LVQ algorithm applies a reinforced or punished learning principle. If the current training pattern is correctly classified, the winning prototype vector will be moved closer toward the input pattern. If the input pattern is incorrectly classified, the prototype vector will be moved away from the input [71]. A drawback of the LVQ network is that the number of clusters in the competitive layer needs to be determined. Moreover, it needs off-line training, assuming that all input patterns and corresponding targets are known. Furthermore, in the learning process, the LVQ requires iterative presentations of the input patterns. Thus, the LVQ network is not suitable for handling on-line real-time problems that require continuous learning.

2.4 Fuzzy Sets for Pattern Classification

Zadeh introduced fuzzy sets in 1965 as a means of representing and manipulating data that were not precise but rather fuzzy. As an alternative of crisp logic, fuzzy logic serves as a useful theoretical foundation for information characterization in uncertain and fuzzy circumstances. Using fuzzy set theory as the basis of inference establishes an estimated foundation for obtaining an accurate form that carries out the condition of inexact rather than exact rationale [134]. The main advantage of all fuzzy classification techniques lies in the fact that they provide a soft decision, a value that describes the degree to which a pattern fits within a class.

Fuzzy logic inference is comprised of three principal processes: a fuzzification process, a rule evaluation process, and a defuzzification process. The fuzzification process fuzzifies inputs using membership functions to obtain membership values between 0 and 1. In the rule evaluation process, the fuzzified inputs are determined via a set of "if-then" rules using fuzzy operators. Defuzzification process is then used to integrate the fuzzy outputs back to a crisp solution.

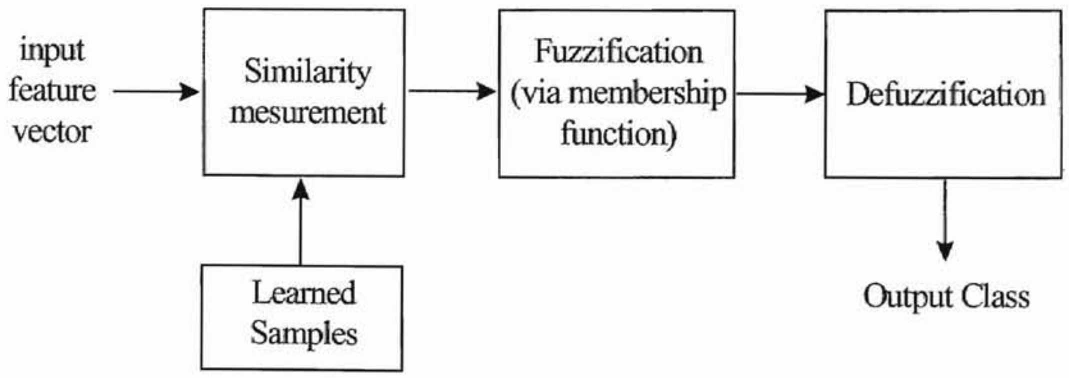


Figure 2.9 The Concept of Pattern Classification Problems Using Fuzzy Systems

A pattern classification system may be derived from the fuzzy set theory as depicted in Figure 2.9. At the first stage, the similarity criterion is determined between the input pattern and reference samples or prototype patterns. After that, the similarity measurement is fuzzified to membership domain by a membership function. Then, the membership values of all prototypes in that class are integrated to determine the overall degree of a specific class to which the input pattern belongs. Finally, the defuzzification process determines a crisp value of a certain class [74].

The fuzzy set theory has obviously had a great impact on pattern classification techniques. Various researchers have studied and developed different fuzzy classification algorithms. The main benefit of all fuzzy classifiers lies in the fact that they have the capability of human-like decisions, which is appropriate for real world problems [134]. Some existing fuzzy classification approaches are Fuzzy-Rule-Base Methods [61], Linguistic Recognition System [98], Weighted Fuzzy Pattern Matching [33], Fuzzy Integral [46], Fuzzy c-Means [9], Fuzzy k-Nearest-Neighbor [10, 69], and Fuzzy Decision Tree [22].

2.5 Fuzzy Neural Network for Pattern Classification

The integration of neural networks and fuzzy sets into the same architecture results in fuzzy neural network pattern classifiers. Because of their massive parallel computational units, neural networks have the advantage of fast computation so that it is possible to process real time estimation of extensive information. The benefit of fuzzy systems lies in their ability to handle unclear data usually experienced in real world problems [134]. Fuzzy neural networks tend to be very advantageous dealings with fuzzy problems in real life. Fuzzy neural classifiers have become a primary area of research.

Some examples of fuzzy neural networks and neural-fuzzy systems for pattern classification problems are Knowledge-Based Fuzzy MLP [89], Neural-Network-Based Fuzzy Classifier [126], Fuzzy Self-Organizing Map [127], Fuzzy Learning Vector Quantization [67], Adaptive Neural Fuzzy Inference System [62], On-line Self-Constructing Neural Fuzzy Inference Network (SONFIN) [64], Fuzzy Min-Max Neural Network [112-113], Fuzzy ART Neural Network [19], Fuzzy ARTMAP Neural Network [18, 20, 21], Gaussian ARTMAP Neural Network [130], and RBF Fuzzy ARTMAP Neural Network [123].

Fuzzy ARTMAP neural network is a good example of incremental learning networks that can learn new knowledge without forgetting existing knowledge. It can learn new information without retraining old information. The concept of the Fuzzy ARTMAP is used as the main model to develop the algorithm proposed in this study. The Fuzzy ARTMAP is briefly discussed here.

2.5.1 Fuzzy ARTMAP Neural Network

Figure 2.10 depicts the architecture of the Fuzzy ARTMAP neural network. The Fuzzy ARTMAP neural network classifies analog or binary input-output pattern through on-line supervised learning. It combines two Fuzzy ART [19] modules, Fuzzy ART$_a$ and Fuzzy ART$_b$. Input patterns are presented at Fuzzy ART$_a$ module while their corresponding outputs are presented at Fuzzy ART$_b$ module. The two modules are linked together via an inter Fuzzy ART module called a map field, F^{ab}, which is used to determine whether the correct mapping from inputs to outputs has been achieved and to realize the match tracking rule whereby the vigilance parameter of Fuzzy ART$_a$ increases in response to a predictive mismatch at Fuzzy ART$_b$ [18].

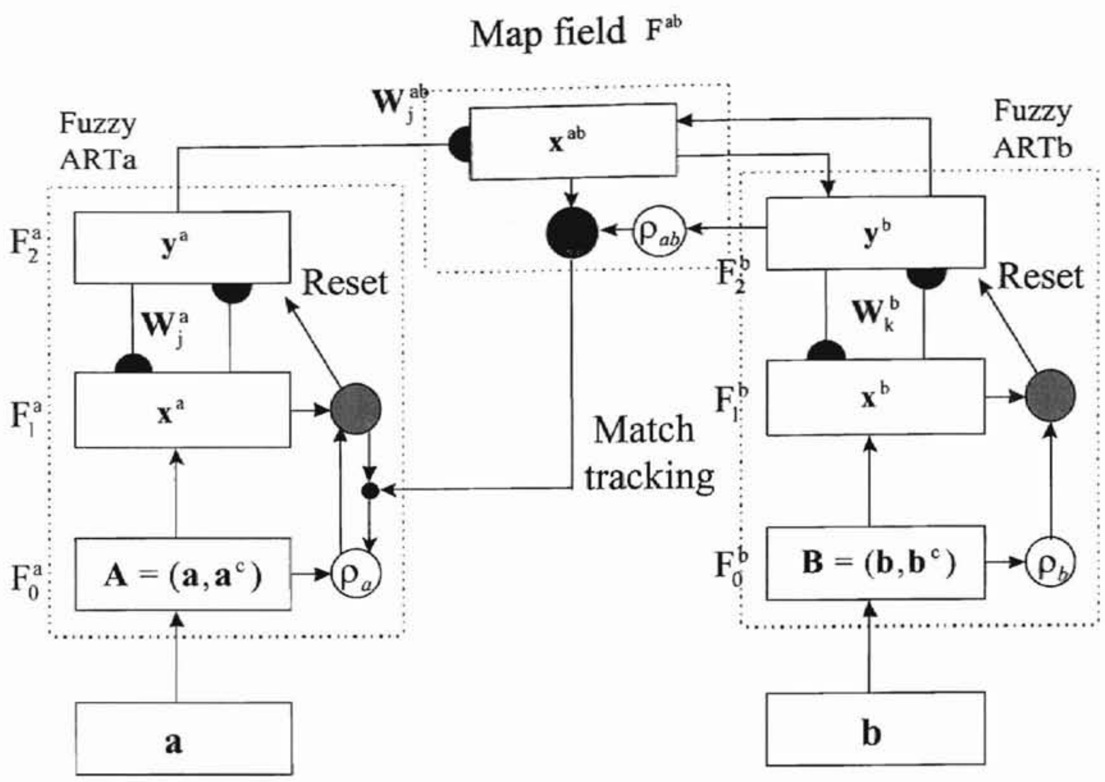


Figure 2.10 Fuzzy ARTMAP Architecture [18]

The main concept of the Fuzzy ARTMAP is that input patterns are presented to Fuzzy ART$_a$ to be clustered into groups while the corresponding targets are presented to the Fuzzy ART$_b$ to be also clustered into groups. Then the two modules are mapped to correct input and output pairs via a map field module. The Fuzzy ARTMAP learns to classify inputs by a fuzzy set of features or a pattern of fuzzy membership values between 0 and 1 [18]. A hyperbox is used to represent the distributions of the input space. Its minimum point and its maximum point define a hyperbox fuzzy set. A membership function is defined with respect to these hyperbox minimum and maximum values in each dimension. Extensive details of the Fuzzy ARTMAP neural network are discussed in [18].

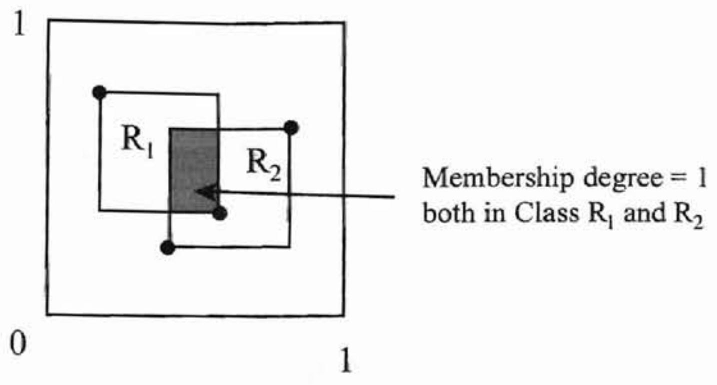


Figure 2.11 Two-Dimensional Hyperboxes of Fuzzy ART
Showing Ambiguous Classes in Shaded Area.

Despite the beneficial property of on-line incremental learning, some drawbacks of the Fuzzy ARTMAP neural network presented in the literature are as follows: 1) It is sensitive to the presentation order of training pattern. Different presentation orders of training pattern yield different decision boundaries. 2) It has no mechanism to avoid overfitting and hence should not be used with noisy data. 3) In the Fuzzy ART system full membership functions are allowed to overlap for each hyperbox, leading to the confusion of pattern classifying, as shown in Figure 2.11.

2.6 Summary of the Literature Survey

Pattern classification techniques have become important to handle many real-world applications. As a complement to statistic classifiers, neural network classifiers, fuzzy classifiers, and neural-fuzzy classifiers have been applied to deal with the classification problems. However, those neural network, fuzzy, and neural-fuzzy classifiers have some deficiencies in many aspects.

For example, the MLP and the LVQ classifiers require off-line training and iterative presentation of training input. Thus, they use extensive training time to learn input patterns. Furthermore, these networks need predetermination of their architecture parameters in advance. Repeated design work is needed to find the optimal parameters to gain reasonable result in classifying patterns. Moreover, sometimes they fail in the learning process by being unable to converge to the optimal solution because the initial random condition is unsuitable. Hence, MLP and LVQ classifiers are not acceptable in some cases of pattern classification problems that need fast, on-line, real-time, incremental learning.

Despite neural net classifiers, fuzzy classifiers provide the ability of handling vague information. In addition, they offer a soft decision that allows a pattern to belong to several classes in different membership degrees. This property is applicable in many pattern classification problems. Nevertheless, designing fuzzy classifiers needs expert knowledge to form "if-then" rules. Moreover, the designing process is time consuming, especially when the dimension of feature space becomes large. Thus, fuzzy classifiers are not candidates for on-line, real-time, incremental learning pattern classification.

The integration of fuzzy neural network for pattern classification has been increasingly applied in pattern classification problems since it provides trainable systems and parallel computations with the ability of dealing with inexact information and forming a soft decision. Despite many iterative off-line learning fuzzy-neural classifiers, the Fuzzy ARTMAP neural network is an on-line supervised incremental learning classifier. Thus, it is acceptable for real-time learning pattern classification. However, some drawbacks of the Fuzzy ARTMAP neural network have been discussed in the literature.

To overcome some of those deficiencies of neural net classifiers, fuzzy classifiers, and fuzzy-neural network classifiers, we have developed a novel class of fuzzy-neural network, called an "Incremental Learning Fuzzy Neural Network" (ILFN). The ILFN preserves all the benefits of existing fuzzy neural networks, while addressing the issues of (1) fast, one-pass, on-line, real-time, incremental learning; (2) forming a soft or hard decision, or both soft and hard decisions; (3) dealing with nonlinear and overlapping classes; and (4) detecting new classes with a fast update of its parameters while in an operating mode. Details of the ILFN classifier are given in the next chapter.

CHAPTER III

NETWORK ARCHITECTURE AND CLASSIFICATION ALGORITHM

3.1 Introduction

In this Chapter, the network architecture and the classification algorithm of the proposed ILFN network are introduced. The ILFN is a synergetic combination of fuzzy sets [134] and neural networks. A fuzzy set membership function is employed as a discriminant function to determine the degree of similarity of input patterns to the prototypes of the input space. Gaussian neurons are used in the hidden layer of the input subsystem of the ILFN system. The concept of the fuzzy ARTMAP [18] is applied in the ILFN such that input patterns are presented to an input subsystem to be clustered into groups, while the corresponding targets are presented to a target subsystem. Then the outputs from the two subsystems are mapped together via a decision layer.

The ILFN classifier employs both an unsupervised and supervised learning scheme. The supervised learning algorithm is used when the classifier is in a training phase, provided input and output pairs are given. On the contrary, the unsupervised learning algorithm is used when the target is to be determined, especially when the system is in an operating phase. The system is allowed to detect novel categories that may be developing during operation. The learning process of the ILFN network is developed not only in the training step but also in the operation step.

3.2 The ILFN Network Architecture

The network architecture of the ILFN classifier is distinguished by two different modes: a training mode (shown in Figure 3.1) and an operating mode (shown in Figure 3.2). The two modes have differences only in the controller module and the target labeling module. The training mode uses the supervised learning scheme requiring pairs of input and target of patterns to construct prototypes of the system. On the other hand, the operating mode uses the unsupervised learning algorithm to determine the target class for a given input pattern. When the system detects new categories, it uses the target labeling module to assign the corresponding targets to the coming input patterns. The targets that are assigned to the novel prototypes are significantly different from the existing targets in the target module. The following discussion describes the details of the architecture of the ILFN classifier.

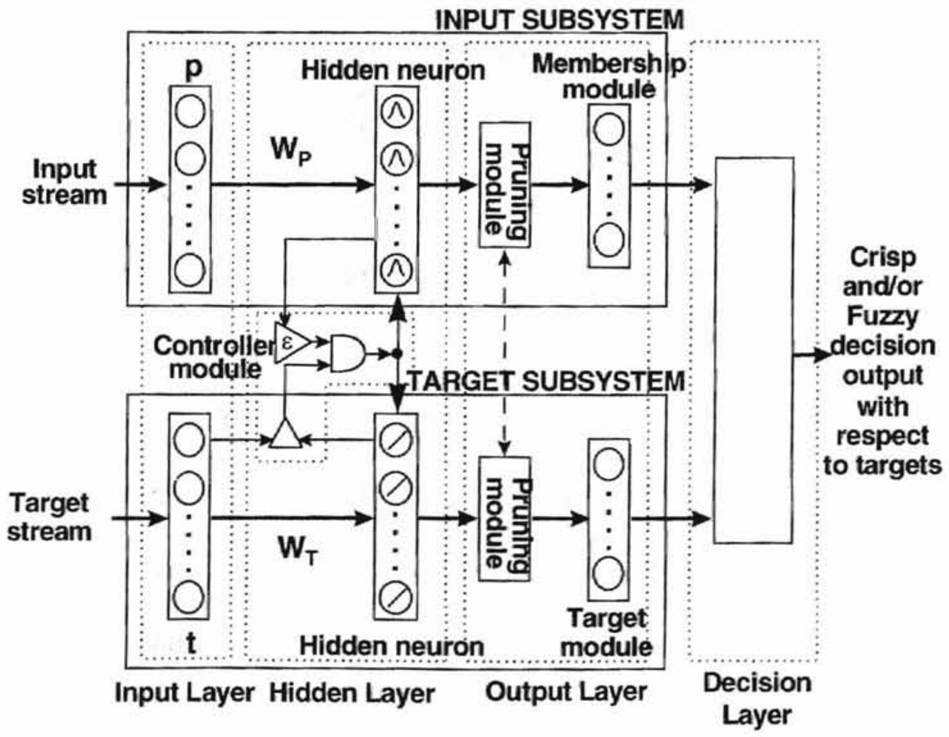


Figure 3.1 Network Architecture of the ILFN Classifier in the Training Mode

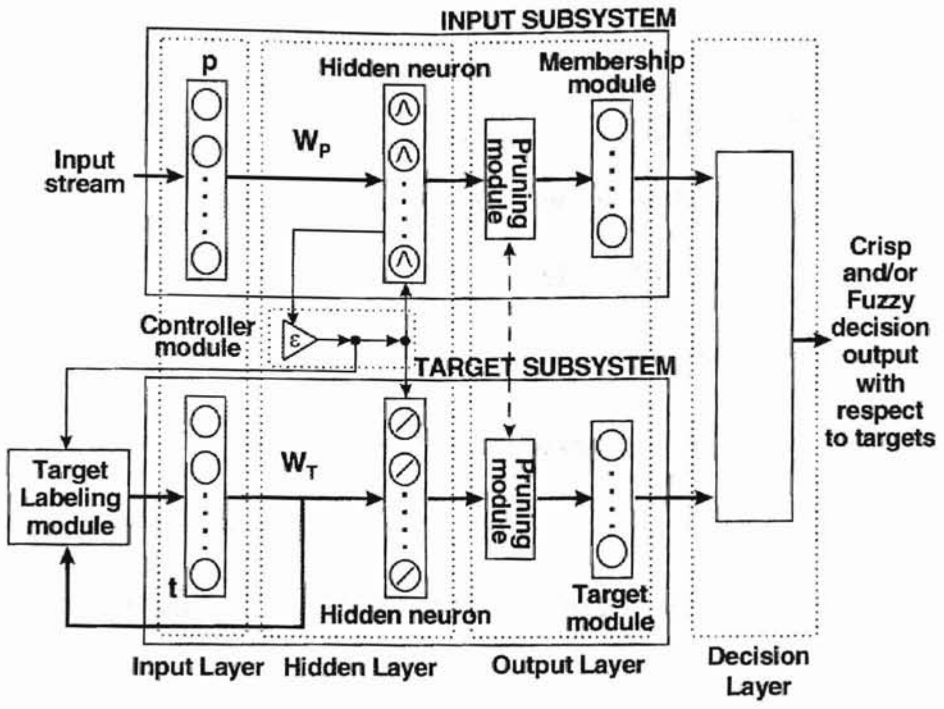


Figure 3.2 Network Architecture of the ILFN Classifier in the Operating Mode

The ILFN system has four layers: one input layer, one hidden layer, one output layer, and one decision layer, as shown in Figure 3.1 and Figure 3.2. Generally, the system is composed of two subsystems: an input subsystem and a target subsystem. Each subsystem has three layers: one input layer, one hidden layer, and one output layer. The hidden layer of both the input subsystem and the target subsystem are linked together via a controller module which is used to control the growing neurons in the hidden layer. Each output layer of both subsystems consists of two modules. The output layer of the input subsystem consists of a pruning module and a membership module, while the output layer of the target subsystem consists of a pruning module and a target module. The membership module of the input subsystem and the target module of the target subsystem are simultaneously updated with their number of neurons controlled by the pruning modules. The output of the classifier is linked together via a decision layer.

3.2.1 Input Subsystem

Figure 3.3 illustrates the input subsystem of the ILFN classifier. Each neuron of the input layer connects to each element of an input vector. Neurons of the input layer are fully connected to neurons of the hidden layer via a dynamic synaptic weight matrix, $\mathbf{W}_P$, whose rows represent prototype vectors which are the centriods of radial basis function in the hidden layer. When the neurons of the hidden layer grow, $\mathbf{W}_p$ adds more rows. In addition, it is a long-term-memory trainable weight. (Long-term memory refers to the information that will be stored and used from the beginning to the end of the process.)

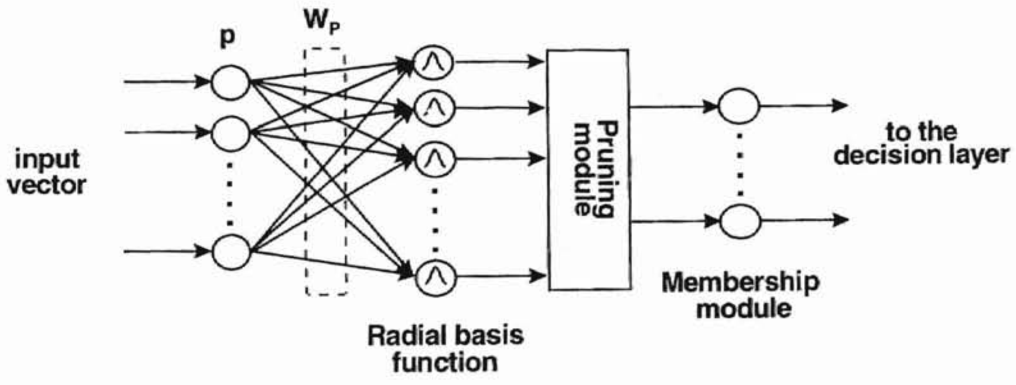


Figure 3.3 The Input Subsystem of the ILFN Classifier

Gaussian membership functions are used in the hidden layer of the ILFN. The Gaussian functions are centered on the mean vectors of clusters which are called prototypes of the input pattern space. The membership functions are used to fuzzify input vectors, $\mathbf{p}$, into membership values, m_i, with respect to the distance measure between the input vectors, **p**, and the vector prototypes The membership function at the *i*th neuron, $m_i(\mathbf{p}, \mathbf{w}_{Pi})$, is defined by the following equation:

$$m_i(\mathbf{p}, \mathbf{w}_{Pi}) = \exp\left(-\frac{\left\|\mathbf{p} - \mathbf{w}_{Pi}\right\|^2}{2\sigma_i^2}\right), \; i = 1, 2, \ldots, L \tag{3.1}$$

where $\|\cdot\|$ denotes the Euclidean distance which is used as similarity measure between two vectors. A vector **p** represents an input vector. The weight vector between the input layer and the *i*th hidden neuron, $\mathbf{w}_{Pi}$, is the center or mean vector of data at the *i*th neuron in the hidden layer. σ_i represents the standard deviation of the *i*th neuron in the hidden layer. The membership function, $m_i(\mathbf{p}, \mathbf{w}_{Pi})$, of the hidden layer is used to fuzzify the distance between a given input vector **p** and the *i*th centers $\mathbf{w}_{Pi}$ into a real value m_i which represents the degree of similarity between **p** and $\mathbf{w}_{Pi}$. The membership functions produce localized, bounded, and radially symmetric kernels. The membership value monotonically decreases as the distance from the function's centers increases.

The pruning module in the output layer of the input subsystem which is short term memory (referring to information that will be stored and used for only a short period of time, i.e., only for each presenting input) is used to eliminate redundant classes from the hidden layer. Instead of passing many duplicate subclasses, only distinguished classes are passed to the membership module making the system easier to interpret at the output. The pruning module in the input subsystem works together with the pruning module of the target subsystem. Moreover, they have the same number of neurons.

The membership module in the output layer of the input subsystem receives information transmitted from the pruning module and passes them to the decision layer.

The information stored in the membership module is a short-term memory, which means that the information in the membership module differs for different input vectors. Each membership value in the membership module indicates the degree of similarity of an input vector with respect to the target classes of the classifier. The membership values will be mapped to classes in the target module in the target subsystem via the decision layer.

3.2.2 Target Subsystem

The target subsystem of the ILFN classifier is depicted in Figure 3.4. Each neuron of the input layer in the target subsystem is fully connected to each element of a target vector. A synaptic weight matrix, $\mathbf{W}_T$, is used to connect the neurons of the input layer to the neurons of the hidden layer. Unlike $\mathbf{W}_P$ in the input subsystem which is a trainable weight, $\mathbf{W}_T$ needs no training. However, $\mathbf{W}_T$ increases the number of rows when more hidden neurons are added. In addition, $\mathbf{W}_T$ is a long-term-memory weight like $\mathbf{W}_P$. The hidden functions of the target subsystem are simply linear functions.

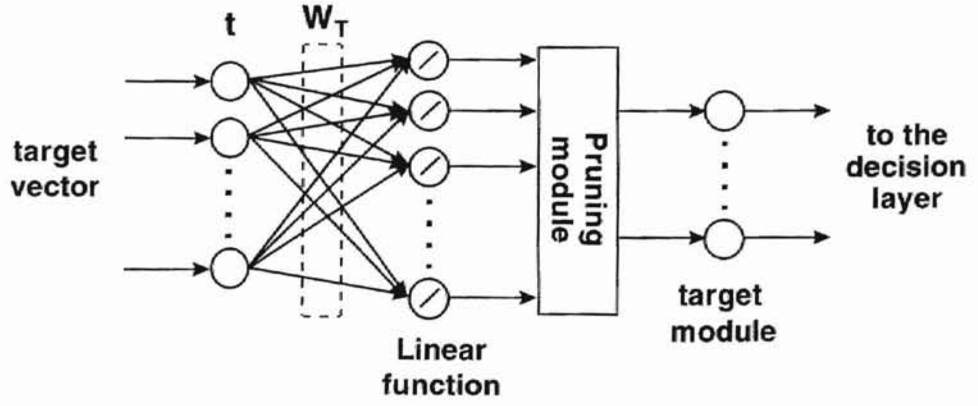


Figure 3.4 The Target Subsystem of the ILFN Classifier

As in the input subsystem, the pruning module of the output layer in the target subsystem, which is also short term memory, is used to eliminate redundant classes in the hidden layer. Instead of passing many duplicate subclasses, only subclasses that have the highest degree of membership for a given input are passed to the membership module. As mentioned before, the pruning module in the target subsystem works together with the pruning module in the input subsystem and they have the same number of neurons.

The target module, which is in the output layer of the target subsystem, receives information passed from the pruning module and submits it to the decision layer. Each neuron of the target module is a class or a target of an input vector. The target module is a short-term memory as is the membership module of the input subsystem. In the same order of indices, the target module will be mapped to the membership module of the input subsystem via the decision layer.

3.2.3 Controller Module

The controller module is used to control the growing of the neurons of the hidden layer of both input subsystem and target subsystem. There are some differences of the controller module in training mode and operating mode.

In the training mode, there are three components in the controller module: two comparators and one AND gate. One comparator is used to compare the winning membership value from the hidden layer of the input subsystem to the threshold, ε. The output of this comparator becomes "true" if the winning membership value is smaller than the ε. This implies that the input vector is significantly different from all existing prototype vectors. The output is sent to one input of the AND gate. Another comparator, which has two inputs, is used to compare the desired target to the predicted output which is stored in the hidden layer of the target subsystem. The output of the comparator becomes "true" if both the desired target and the predicted output are the same, and it is sent to another input of the AND gate. If both input of the AND gate are "true," its output becomes "true." This allows the system to add one more neuron in the hidden units. In other words, the system generates more neurons whenever the membership value of the winning neuron is smaller than the threshold, ε, and the desired target and the decision output are the same.

In the operating mode, the controller module of the ILFN classifier has only one component which is a comparator. The comparator is used to compare the winning membership value in the hidden layer to the threshold, ε. The output of this comparator becomes "true" if the winning membership value is smaller than ε. If the output of the comparator is "true," meaning that a new category is detected, the system adds a new neuron to the hidden layer using the input pattern as the new prototype, and the target labeling module distinguishably assigns a corresponding target to the new prototype.

3.2.4 Target Labeling Module

The target labeling module is used only in the operating mode (see Figure 3.2) that the system is allowed to detect new classes and update the existing information. The learning algorithm in the operating mode is an unsupervised learning since the target is unknown. The module receives one input from the output of the controller module in hidden layer of the target subsystem. The input from the controller module is utilized to tell the target labeling module to assign a target when a new neuron is added to the system. The other input of the target labeling module, representing targets of prototypes, is used to check the existing targets in order to assign a new target that differs from the existing targets.

3.2.5 Decision Layer

The decision layer is used to map the membership values in the membership module of the input subsystem to the target classes in the target module of the target subsystem. The output from the decision layer is the output of the system. The decision output can be interpreted as a soft decision or a hard decision. For the soft decision, the decision output assigns different membership values to the pattern classes or prototypes. This allows a given pattern belonging to more than one class with different degrees of similarity measure. On the other hand, for the hard decision, only one decision which is the class that has the highest membership value is chosen as the output.

3.3 Mathematical Model of the ILFN Classifier

Let $\Re^M$ be a pattern vector space. Let $\mathbf{p} = \left[\mathrm{p}_1\ \mathrm{p}_2 \ldots \mathrm{p}_\mathrm{M}\right]^\mathrm{T} \in \Re^M$ be an input vector. Each element of the vector is a measurement or feature, and each one corresponds to one dimension (axis) in the space. For M elements of the vector we have an M-dimensional space, or M-space. Let $\Re^N$ be N-dimensional space. Let $\mathbf{t} = \left[\mathrm{t}_1\ \mathrm{t}_2 \ldots \mathrm{t}_\mathrm{N}\right]^\mathrm{T} \in \Re^N$ be a corresponding target or class vector of the input vector $\mathbf{p}$. Let matrix $\mathbf{W}_\mathrm{P} = \left[\mathbf{w}_{\mathrm{p}1}\ \mathbf{w}_{\mathrm{p}2} \ldots \mathbf{w}_{\mathrm{p}L}\right]^\mathrm{T}$ be a synaptic weight matrix whose each row vector, $\mathbf{w}_{\mathrm{p}i}^\mathrm{T}$, $i = 1, \ldots, L$, represents a prototype of the pattern space. Each class may have more than one prototype. Each prototype $\mathbf{w}_{pi}^{\mathrm{T}}$ is the mean vector of the patterns that belong to the ith node. Let matrix $\mathbf{W}_{\mathrm{T}} = \left[\mathbf{w}_{\mathrm{T}1}\ \mathbf{w}_{\mathrm{T}2} \cdots \mathbf{w}_{\mathrm{T}L}\right]^{\mathrm{T}}$ be a synaptic weight matrix whose each row vector, $\mathbf{w}_{\mathrm{T}i}^{\mathrm{T}}$, $i = 1, \ldots, L$, represents a target of a prototype stored in $\mathbf{W}_{\mathrm{P}}$ in the same order of their neurons. The number of rows of $\mathbf{W}_{\mathrm{P}}$ and $\mathbf{W}_{\mathrm{T}}$ are the same and they grow dynamically as more neurons are added into the hidden layer.

3.3.1 Similarity Measure

In order to analyze distances between objects or points in the pattern space, a distance measure is used. There are a number of distance metrics that can be used as a tool to measure a similarity between vectors. For example, Euclidean distance, Mahalanobis distance, and Minkowski distance [71] are shown below:

3.3.1.1 Euclidean Distance (hypercircle):

$$d(\mathbf{p},\mathbf{w}) = \sqrt{(\mathbf{p}-\mathbf{w})^{\mathrm{T}}(\mathbf{p}-\mathbf{w})}, \tag{3.2}$$

where $\mathbf{p} \equiv$ an input vector,

$\mathbf{w} \equiv$ a prototype vector (or mean vector).

3.3.1.2 Mahalanobis Distance (hyperellipsoid):

$$d(\mathbf{p},\mathbf{w}) = \sqrt{(\mathbf{p}-\mathbf{w})^{\mathrm{T}}\mathbf{A}^{-1}(\mathbf{p}-\mathbf{w})}, \tag{3.3}$$

where $\mathbf{A} \equiv$ a covariance matrix (symmetric and positive definite) determining shape and orientation of input patterns.

3.3.1.3 Minkowski Distance:

$$d(\mathbf{p},\mathbf{w}) = \left[\sum_{i=1}^{n} \left| p_i - w_i \right|^{\lambda} \right]^{\frac{1}{\lambda}} \tag{3.4}$$

where $\lambda \equiv$ a parameter used to control the shape of regions of attraction .

The necessary conditions that the similarity measure must satisfy with relation to these points $(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ in the pattern space are as following [3]:

1) $d(\boldsymbol{x},\boldsymbol{y}) = d(\boldsymbol{y},\boldsymbol{x})$,
2) $d(\boldsymbol{x},\boldsymbol{y}) \leq d(\boldsymbol{y},\boldsymbol{z}) + d(\boldsymbol{x},\boldsymbol{z})$,
3) $d(\boldsymbol{x},\boldsymbol{y}) \geq 0$,
4) $d(\boldsymbol{x},\boldsymbol{y}) = 0$ iff $y = \boldsymbol{x}$.

Since it is a suitable representation (i.e., the shape and orientation of the class patterns) for any input space, generally Euclidean distance is used as a distance metric [74]. In our work, each datum **p** presented to the network is measured distances to the prototypes stored in $\mathbf{W_p}$ (each row vector of $\mathbf{W_p}$ represents a prototype), as follows

$$\mathbf{D}(\mathbf{p},\mathbf{W}_\mathrm{P}) = [d(\mathbf{p},\mathbf{w}_{\mathrm{P}1}) \quad d(\mathbf{p},\mathbf{w}_{\mathrm{P}2}) \quad \ldots \; d(\mathbf{p},\mathbf{w}_{\mathrm{P}L})]\,, \tag{3.5}$$

$$d(\mathbf{p},\mathbf{w}_{\mathrm{P}i}) = \sqrt{(\mathbf{p}-\mathbf{w}_{\mathrm{P}i})^{\mathrm{T}}(\mathbf{p}-\mathbf{w}_{\mathrm{P}i})}\;, \qquad i = 1, 2, \ldots, L \tag{3.6}$$

where $\mathbf{D}(\mathbf{p},\mathbf{W}_\mathrm{P})$ is a row vector of $d(\mathbf{p},\mathbf{w}_{\mathrm{P}i})$ which is the Euclidean distance between the input vector **p** and the vector prototype $\mathbf{w}_{\mathrm{P}i}$, the rows of the weight matrix $\mathbf{W}_\mathbf{P}$.

3.3.2 Membership Function

The membership function is employed to represent the degree of similarity between the input pattern and the reference prototypes. Consider the K pattern classes: $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K$. The membership function will be defined such that for all points $\mathbf{p}$ within the region describing $\mathbf{w}_k$, $k = 1, 2, \ldots, K$, there exists a function $m_k(\mathbf{p},\mathbf{w}_k)$ such that $m_k(\mathbf{p},\mathbf{w}_k) > m_j(\mathbf{p},\mathbf{w}_j)$ for all $j \neq k$. Mathematically, $m_k(\mathbf{p},\mathbf{w}_k) > m_j(\mathbf{p},\mathbf{w}_j)$ $\forall$ $\mathbf{p} \in \mathbf{w}_k$, and $\forall\ j \neq k$. Figure 3.5 is an example of a one-dimensional space. A point $\mathbf{p}$ is classified to be the class of $\mathbf{w}_k$ since $m_k(\mathbf{p},\mathbf{w}_k)$ is larger than $m_j(\mathbf{p},\mathbf{w}_j)$.

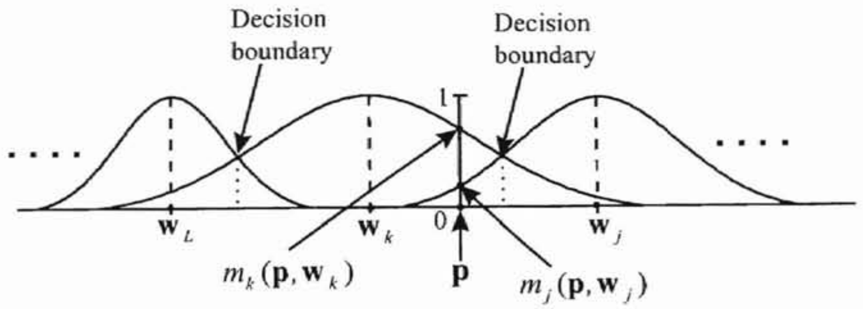


Figure 3.5 A One-Dimensional Pattern Space with Gaussian Membership Functions

Thus within region $\mathbf{w}_k$, the kth membership function will have the largest value. The decision boundary separating region $\mathbf{w}_k$ and $\mathbf{w}_j$ is given by $m_k(\mathbf{p},\mathbf{w}_k) - m_j(\mathbf{p},\mathbf{w}_j) = 0$ which is equivalent to those points in the space which have equal membership functions for both $\mathbf{w}_k$ and $\mathbf{w}_j$, as shown in Figure 3.5.

The membership function $m(\mathbf{p},\mathbf{w})$ should satisfy the following conditions [3, 74]:

1) $m(\mathbf{p},\mathbf{w})$ should be unity if $d(\mathbf{p},\mathbf{w}) = 0$ (i.e., maximized for $\mathbf{p} = \mathbf{w}$),

2) $m(\mathbf{p},\mathbf{w})$ should monotonically decrease to zero as $d(\mathbf{p},\mathbf{w})$ increases,

3) $m(\mathbf{p},\mathbf{w})$ should be approximately zero as $d(\mathbf{p},\mathbf{w})$ reaches infinity,

4) $m(\mathbf{p}_1,\mathbf{w}) = m(\mathbf{p}_2,\mathbf{w})$ if $d(\mathbf{p}_1,\mathbf{w}) = d(\mathbf{p}_2,\mathbf{w})$

In this thesis Gaussian radial basis functions are used as membership functions for fuzzifying input vector **p** by the equation

$$m_i(\mathbf{p}, \mathbf{w}_{\mathrm{P}i}) = \exp\left(-\frac{\left\|\mathbf{p} - \mathbf{w}_{\mathrm{P}i}\right\|^2}{2\sigma_i^2}\right), \; i = 1, 2, \ldots, L. \tag{3.7}$$

The Gaussian membership functions fuzzify an input pattern into membership domain which is stored in the vector **m**,

$$\mathbf{m} \; = \; [m_1(\mathbf{p},\mathbf{w}_{\mathrm{P}1}) \quad m_2(\mathbf{p},\mathbf{w}_{\mathrm{P}2}) \quad \ldots \quad m_L(\mathbf{p},\mathbf{w}_{\mathrm{P}L})] \; , \tag{3.8}$$

where **m** is a row vector of membership values m_i, $i = 1, 2, \ldots, L$, which represents the degree of similarity between **p** and $\mathbf{w}_{\mathrm{P}i}$.

3.3.3 ILFN System Dynamics

Both $\mathbf{W_p}$ and $\mathbf{W_T}$ are allowed to dynamically grow the number of neurons when the system detects new classes. However, only $\mathbf{W_p}$ can adaptively change its information or learn new prototypes. At the initialized state, there is no neuron in the hidden layer. The first neuron in the hidden layer is setup after the first input vector **p** is presented to the input subsystem of the network while the first target vector **t** is presented to the input layer in the target subsystem. Then both $\mathbf{W_p}$ and $\mathbf{W_T}$ setup the first neuron using **p** and **t** respectively. The next input vector will be compared to the existing prototype. If there is a significant difference, then a new neuron is added to the hidden layer; **p** is added to $\mathbf{W_p}$ and **t** is added to $\mathbf{W_T}$. On the other hand, if it meets the similarity criterion then, instead of adding a new neuron, the learning process is performed. The $\mathbf{W_p}$ and other parameters are updated to include the new data to the existing subclasses.

3.3.4 Learning Process in ILFN System

The learning process takes place only in the hidden layer which is the changing of the synaptic weight $\mathbf{W_p}$ which keeps the prototypes of the input space. Each input vector $\boldsymbol{p}$ in input space is fuzzified to a membership value at each node of the hidden layer with respect to distance measure between input vector $\boldsymbol{p}$ and the synaptic weight matrix $\mathbf{W_P}$. The winning node of the hidden layer is determined by the defuzzification process using the fuzzy OR operation [134] defined as

$$winner = \max(\mathbf{m}) = m_1 \vee m_2 \vee \ldots \vee m_L, \tag{3.9}$$

$$J \equiv winner\ index = \arg\max(\mathbf{m}), \tag{3.10}$$

where $m_1 \vee m_2 = m_1$ if $m_1 > m_2$; $m_1 \vee m_2 = m_2$ if $m_1 < m_2$. Only the parameters of the winner node (i.e., Jth neuron) including count, mean, and standard deviation are updated, while other losing nodes remain the same, as follows:

$$\mathrm{C}_J = \mathrm{C}_J + 1, \tag{3.11}$$

$$\mu_J = \frac{\mu_J(\mathrm{C}_J - 1) + \mathbf{p}}{C_J}, \tag{3.12}$$

$$\sigma_J = \begin{cases} \sqrt{(1 - \frac{1}{\mathrm{C}_J})\sigma_J^2 + \frac{(\mu_J - \mathbf{p})^2}{\mathrm{C}_J}} & \text{if } \mathrm{C}_J > 1, \\ \sigma_0 & \text{otherwise.} \end{cases} \tag{3.13}$$

C_J represents the number of inputs that have been counted into the Jth subclass. The mean μ_J, the center or prototype of the Jth subclass, is indeed a row in the synaptic weight $\mathbf{W_P}$. The standard deviation, σ_J, will be used to indicate the spread of the data in the Jth subclass. σ_0 is the initial standard deviation representing the isotropic spread in pattern space of a new category for the first sample.

3.3.5 Decision Boundaries

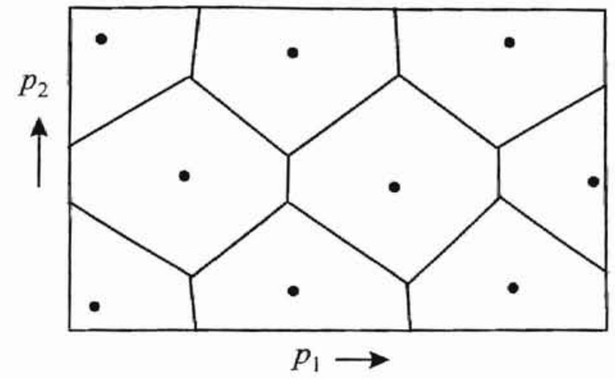


Figure 3.6 Two-Dimensional Voronoi Tessellation

To understand the decision boundaries made by the ILFN, a concept of the vector quantization method called *Voronoi tesselation* is introduced. The vector quantization method is widely used in pattern recognition problems. Figure 3.6 illustrates a two dimensional space where a finite number of prototypes (or *codebook* or *reference vectors*) is shown as points corresponding to their coordinates. This space is separated into portions, bordered by lines (hyperplanes in multi-dimensional space) such that each portion contains a prototype vector that is the "nearest neighbor" to any vector within the same portion. These lines, or the "midplanes" of the neighboring prototype vectors, together compound the Voronoi tessellation. All $\mathbf{p}$ vectors that have a particular prototype vector as their closest neighbor, i.e, all **p** vectors in the corresponding portion of the Voronoi tessellation, are said to make up the *Voronoi set* [71].

The purpose of pattern classification is to determine to what category of class a given sample belongs. Through an observation or measurement process, we obtain a set of numbers which make up the observation vector. The observation vector serves as the input to a decision rule by which we assign the sample to one of the given classes.

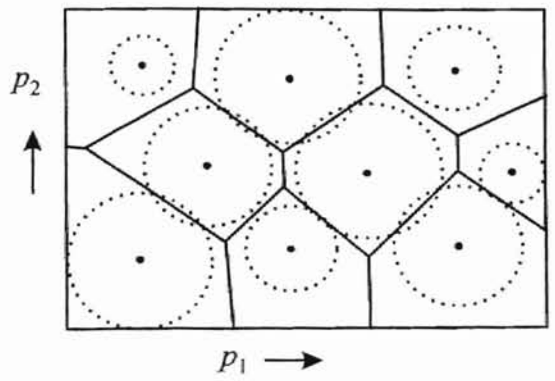


Figure 3.7 The Decision Boundaries among Prototypes of the ILFN

The decision boundaries of the ILFN network distinguish among prototypes in the Voronoi tessellation. Each prototype has its own region separated by the decision boundaries. However, the decision boundaries of the ILFN network are slightly different from the Voronoi tessellation of the vector quantization. For the vector quantization, the decision boundary is the (imaginary) line drawn perpendicular at half the distance to the (imaginary) line between two prototypes. Since the ILFN uses Guassian type membership functions with different standard deviations, the boundary is not half the distance between the adjacent prototypes. However, the decision boundary between the neighboring prototype vectors is a line containing the points that have the same degree of the membership value, as an example shown in Figure 3.7. Figure 3.7 shows the decision boundaries among prototypes of the ILFN in which dotted circles indicate the spread of statistical data for each prototype.

3.4 Classification Algorithm

There are two learning procedures in the classification algorithm of the ILFN network: learning in a training procedure and learning in an operating procedure. The summaries of the classification algorithm in the training and operating procedures of the ILFN are as follows.

3.4.1 Training Procedure

Step 1: Set the user-defined threshold parameter, ε, and the initial standard deviation σ_0.

Step 2: Read in the first input pattern

- Use the first input pattern to set up the first prototype (or mean) to $\mathbf{W_P}$.
- Set the number of patterns for the first node to be 1.
- Set the standard deviation equal to the initial standard deviation, σ_0.
- Set a new neuron to $\mathbf{W_T}$ using the first target $\mathbf{t}$ to be the corresponding target of the prototype in $\mathbf{W_P}$.

Step 3: Read in the next training sample with an input and target pattern.

Step 4: Measure Euclidean distance between the input $\mathbf{p}$ and the prototype $\mathbf{W_P}$, using equation (3.5) and (3.6).

Step 5: Calculate membership values for each node, using the Gaussian type radial basis function as in equation (3.7) and (3.8).

Step 6: Assign membership values to each node (i.e., subclass). The current input pattern has different degrees for each node or subclass. For each class, select the maximum membership value from each subclass to represent the degree of similarity with respect to that class.

Step 7: Identify the largest membership using the Fuzzy OR operator as in equation (3.9) and (3.10).

Step 8: For the winner node, perform two rules:

Rule 1: If the winner is larger than ε and the target $\mathbf{t}$ is the same value as $\mathbf{W_T}$ at the winner node then update weight $\mathbf{W_P}$, the standard deviation, and the number of patterns belong to this node, using equation (3.11), (3.12), and (3.13).

Rule 2: If Rule 1 is not satisfied, then:

- Set a new node center for $\mathbf{W_P}$ using the input pattern $\mathbf{p}$.
- Set the number of patterns for the new node to be 1.
- Set the initial standard deviation to the new node.
- Add a new neuron to $\mathbf{W_T}$ using the new target $\mathbf{t}$ as the corresponding target of a new prototype in $\mathbf{W_P}$.

Step 9: If there are no more input patterns, then stop. Otherwise, go to step 3.

3.4.2 Operating Procedure

Step 1: Read in an input pattern.

Step 2: Measure Euclidean distance between the input $\mathbf{p}$ and the weight $\mathbf{W_P}$, using equation (3.5) and (3.6).

Step 3: Calculate membership values for each node, using the Gaussian type radial basis function as in equation (3.7) and (3.8).

Step 4: Assign a membership value to each node (i.e., subclass). The current input pattern has different degrees for each node or subclass. For each class, select the maximum membership value from each subclass to represent the degree of that class.

Step 5: Find the largest membership value using the Fuzzy OR operator as in equation (3.9) and (3.10).

Step 6: For the winner node, perform two rules:

Rule 1: If the winner is larger than ε and the number of patterns is less than the maximum number of allowed patterns, then update the weight $\mathbf{W_P}$, the standard deviation, and the number of patterns belonging to this node, using equation (3.11), (3.12), and (3.13).

Rule 2: If the winner is smaller than ε then

- Set a new node center for $\mathbf{W_P}$ using the input pattern $\mathbf{p}$.
- Set the number of patterns for the new node to be 1.
- Set the initial standard deviation to the new node.
- Add a new neuron to $\mathbf{W_T}$ and assign a new target as the corresponding target of a new prototype in $\mathbf{W_P}$. (The assigned new target must be significantly different from the existing targets already stored in $\mathbf{W_T}$. For example, if the existing targets in $\mathbf{W_T} = [1\ 2\ 3]^T$, the new target should be "4," that is $\mathbf{W_T}$ becomes $[1\ 2\ 3\ 4]^T$.)

Step 7: If there are no more input patterns, then stop. Otherwise, go to step 1.

3.5 Summary of the ILFN Algorithm

The ILFN algorithm can learn pattern classes within one pass through the infinite number of training data in the pattern space and it can add new pattern classes or prototypes on the fly. Moreover, it can refine current pattern classes as new information is acquired and it uses simple operations that allow quick performance. The system learns adaptively from given examples as a supervised learning algorithm. Input vectors are presented to the system one at a time as on-line learning. Each row (i.e., node of the hidden layer) of $\mathbf{W_P}$ represents a mean or centroid of a cluster. The number of clusters is determined by both the threshold and class prediction. The system generates many clusters if the threshold is large and few cluster if it is small. However, clusters that belong to the same class are grouped together via the pruning module. Each node of $\mathbf{W_T}$ stores the corresponding target of the input prototype patterns. The system has the ability to learn new information on-line without forgetting the learned information.

CHAPTER IV

SIMULATIONS AND RESULTS

To demonstrate the performance of the ILFN classifier, software simulations were used in our experiments. The simulation programs were written to run under MATLAB version 5.1 or higher. A Pentium 233MMX PC hosted the simulation programs. Four data sets were used for training and testing the classifier in our studies. The first benchmark data set was the well-known Fisher's Iris data set [38]. The second data set was a vowel data set. The third data set was the two-spiral problem. The vowel data set and the two-spiral data set are electronically available from the connectionist benchmark collection at Carnegie-Mellon University, Pittsburgh, PA [129]. For the first three data sets used in this study, the results have shown that the ILFN is capable of learning on-line real-time in only one pass through all training data. In addition, the prediction capability of the ILFN classifier was found to be as good as or even better in many cases than many existing classifiers. With the ability of "fast, one-pass, on-line, real-time, incremental learning," the ILFN has shown to be applicable in real-world applications. The last and most important data set was a time-series vibration data set known as Westland vibration data [16]. The detail of four experiments is as follows.

4.1 Fisher's Iris Flower Data Set

The Fisher's Iris flower data set consists of 150 patterns and four features: sepal length, sepal width, petal length, and petal width. The four features describe the shape and size of the Iris flowers. Each pattern in the data set falls into one of three classes: Setosa, Versicolour and Virginica, with a total of 50 patterns per class. For the purpose of this experiment, we will call them Class 1, Class 2, and Class 3, respectively. Class 1 is linearly separable from the other two. However, Class 2 and Class 3 are not linearly separable from each other.

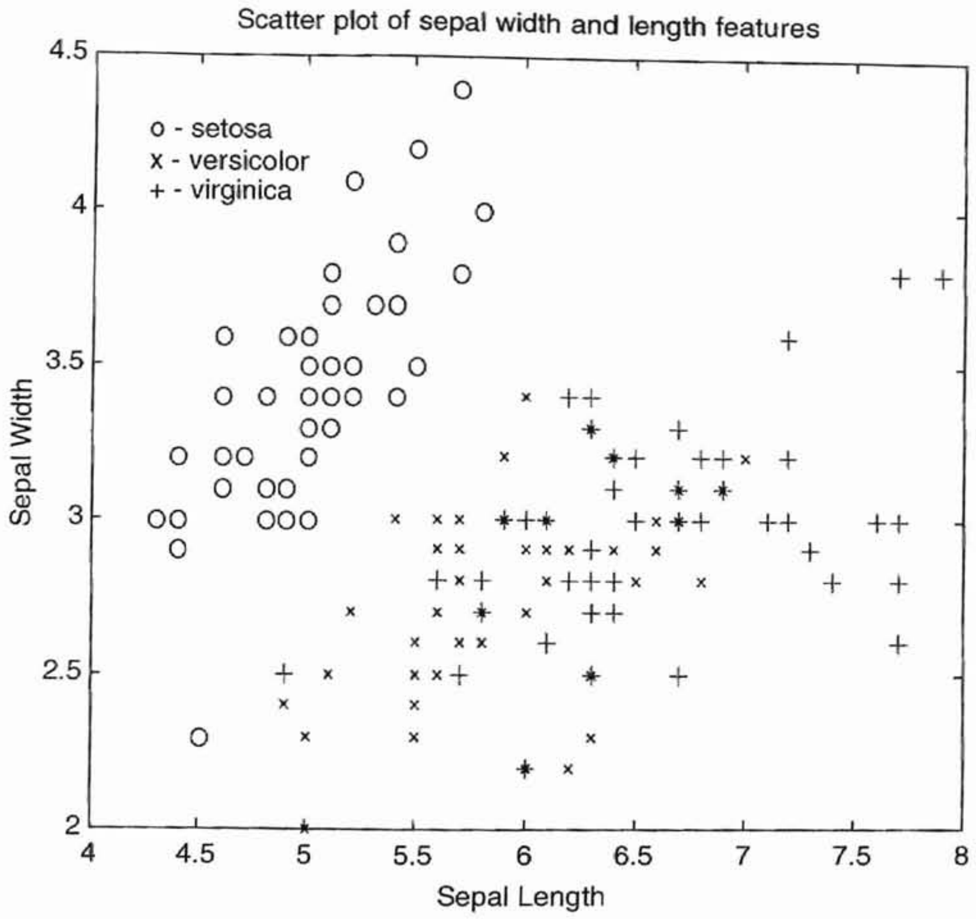


Figure 4.1 Scatter Plot of Sepal Width and Length Features of the Fisher's Iris Data

Figure 4.1 shows the scatter plot of Iris data for sepal width and length features. It is worth noting from the plot that Class 1 can be easily separated from Class 2 and Class 3. However, Class 2 and Class 3 seem very difficult to separate since there is an overlap between them. Moreover, in Figure 4.2, the petal width and length features are plotted showing that Class 1 is very well separated from Class 2 and Class 3. However, Class 2 and Class 3 remain overlapped [38].

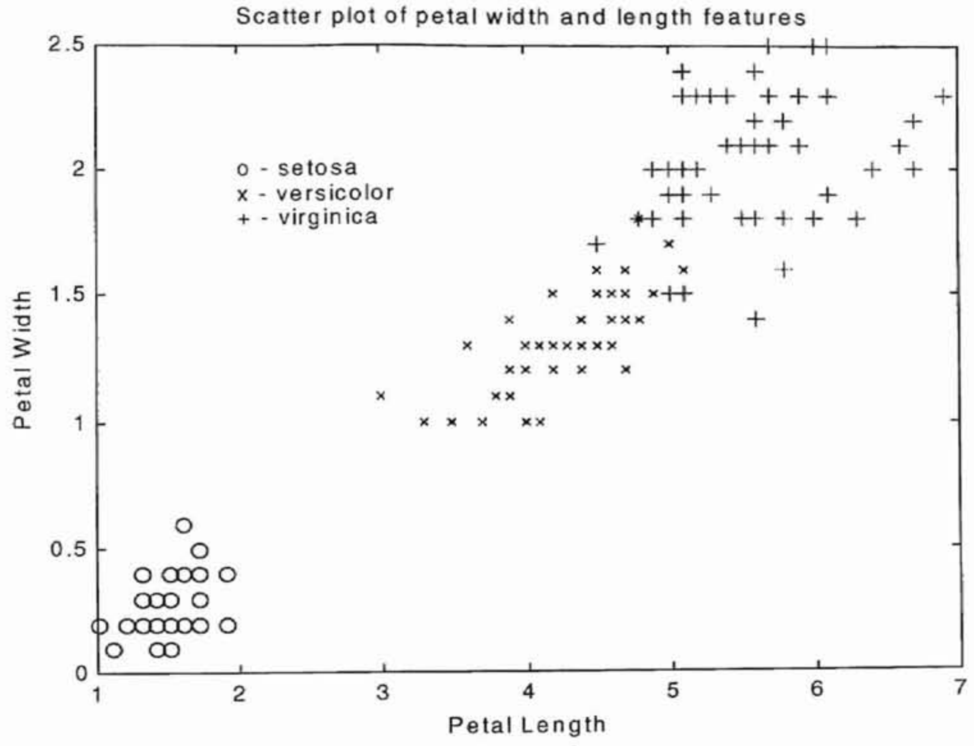


Figure 4.2 Scatter Plot of Petal Width and Length Features of the Fisher's Iris Data

4.1.1 A Comparison between the ILFN and the Fuzzy ARTMAP

In this study, the training data set was composed of the first 25 patterns of each class, while the testing data set was composed of the remaining 25 patterns of each class. Twenty trials were performed in this experiment. For each trial, the presentation order of the training data was randomly selected. To compare the performance of the ILFN with a similarly supervised on-line incremental learning classifier, the Fuzzy ARTMAP neural network was used in this study. The ILFN and the Fuzzy ARTMAP were trained with the same training data set. Then, both networks were tested for the robustness using the same testing data. The parameters of the ILFN were set as follows: the threshold ε was set between 0 and 1, and the initial standard deviation, $\sigma_0 = 0.001$. The parameters of the Fuzzy ARTMAP were set as follows: the vigilance parameters $\rho_a = 0.5$ and $\rho_b = 0.5$, and the learning rate $\beta = 1$. The results of the study are shown in Table 4.1.

TABLE 4.1

COMPARISON PERFORMANCE BETWEEN
THE ILFN AND FUZZY ARTMAP

	ILFN Classifier			**Fuzzy ARTMAP**		
Trial number	Number iterations	Hidden nodes	**% correct testing set**	Number iterations	Hidden nodes	**% correct testing set**
1	1	6	**94.67**	1	4	**92**
2	1	7	**98.67**	3	6	**90.67**
3	1	4	**96**	3	6	**93.33**
4	1	5	**97.33**	2	5	**96**
5	1	6	**93.33**	2	6	**93.33**
6	1	6	**98.67**	4	6	**94.67**
7	1	6	**94.67**	1	4	**92**
8	1	7	**98.67**	3	6	**90.67**
9	1	8	**97.33**	2	5	**93.33**
10	1	6	**93.33**	2	5	**94.67**
11	1	5	**96**	2	5	**96**
12	1	6	**93.33**	2	6	**93.33**
13	1	7	**98.67**	2	5	**96**
14	1	6	**94.67**	1	4	**92**
15	1	6	**98.67**	2	4	**94.67**
16	1	7	**98.67**	3	6	**90.67**
17	1	6	**94.67**	1	4	**92**
18	1	5	**96**	2	5	**94.67**
19	1	6	**94.67**	2	6	**93.33**
20	1	7	**97.33**	2	5	**96**
Average	1	6.1	**96.268**	2.1	5.15	**93.467**

Remark: Fuzzy ARTMAP used in this study is in the Art Gallery version 1, written by Lar Liden. The Art Gallery was obtained from ftp://cns-ftp.bu.edu/pub/ ART_GALLERY/Windows/win_gal.zip.

From Table 4.1, using the testing data, the IFLN achieved maximum correct classification of 98.67% and minimum correct classification of 93.33%. The average of 96.268% correct classification was obtained. On the other hand, the Fuzzy ARTMAP classifier achieved the average of 93.33%, the maximum of 96%, and the minimum of 92% correct classification. It was found that the ILFN classifier performed better than the Fuzzy ARTMAP did in this data set.

Moreover, the ILFN used only one-iteration learning through all training data while the Fuzzy ARTMAP used one to four iterations to learn the training patterns. However, both algorithms used training times within only a few seconds. For this data set, the number of nodes of the ILFN was not sensitive to the threshold value, ε, i.e., different values of ε (between 0 and 1) yielded the same number of hidden neurons and the same performance of correct classification. On the contrary, the number of hidden neurons of the Fuzzy ARTMAP was very sensitive to the choices of vigilance parameters, ρ_a and ρ_b.

4.1.2 Comparisons among other Classifiers

Table 4.2 shows the classification performance among other classifiers with the Fisher Iris data. The classifiers in row one to row six were reported by Simpson [112], showing that most of the classifiers were able to predict testing data with the number of incorrect classification between 2-4. (See details in [112] on how to construct the training and the testing data for these experiments.) It is worth mentioning that those classifiers, except the fuzzy min-max classifier, cannot learn on-line. Fuzzy min-max classifier, which is an unsupervised algorithm, uses hyperboxes for representing the input distribution; on the other hand, the ILFN classifier uses the Gaussian function which is more appropriate to represent the distribution of data space. The summary results of the

ILFN and the Fuzzy ARTMAP in this study are also included in the last two rows of Table 4.2.

TABLE 4.2

A COMPARISON PERFORMANCE
AMONG EXISTING CLASSIFIERS

Technique	No. Wrong	Remarks
Bayes classifier*	2	Very amenable to this type of data.
k-nearest neighbor*	4	Scales up poorly.
Fuzzy k-NN*	4	Allows fuzzy labels for data points.
Perceptron*	3	Limited to linear discrimination.
Fuzzy perceptron*	2	Fuzzifies linear boundaries.
Fuzzy min-max *	2	Single pass learning, learns on-line. (hyperbox distribution).
Fuzzy ARTMAP	2-6 (Run 20 trials)	Learns on-line with 1 to 4 passes less than one second. Uses hyperbox distribution.
ILFN classifier	1-5 (Run 20 trials)	One-pass on-line learning within less than one second. Uses Guassian distribution.

* According to Simpson in [112]

4.2 Vowel Recognition Data

The Vowel Recognition Data (Deterding Data) [29] used speaker independent recognition of the eleven steady-state vowels of British English spoken by 15 speakers for a speaker normalization study, using a specified training set of lpc derived log area ratios. Four male and four female speakers were used for training, and an additional four male and three female speakers were used for testing. The data set is in 10-dimensional input space with 528 samples for the training set and 462 samples for the testing set.

In our study, using different thresholds ranging from 10^{-1} down to 10^{-20}, the ILFN classifier generated hidden neurons ranging in number from 127 down to 78. Larger thresholds allowed the classifier to create more neurons than smaller thresholds. However, a larger number of neurons in the hidden layer do not imply a better performance in predicting the testing data. The results of the ILFN experiment with the vowel recognition are shown in Table 4.3.

TABLE 4.3

THE ILFN CLASSIFIER PERFORMANCE ON VOWEL DATA USING DIFFERENT VALUES OF THE THRESHOLD

Threshold	# hidden nodes	training time	% correct of training data	% correct of testing data
10^{-1}	127	2.48 S	99.05	52.81
10^{-4}	101	1.92 S	99.05	54.98
10^{-8}	**90**	**1.92 S**	**98.67**	**57.36**
10^{-12}	82	1.92 S	98.48	54.55
10^{-20}	78	1.92 S	97.54	53.90

Table 4.3 shows the ILFN classifier performance on vowel data using different values of the threshold, ε. The classifier generalized the testing data in various percentages of correct prediction. When using the threshold of 10^{-1}, 127 hidden nodes were generated and the correct prediction of testing data was only 52.81%. On the other hand, 54.98% correct prediction was achieved using the threshold of 10^{-4} given 101 neurons in the hidden layer. Using the threshold of 10^{-8}, the ILFN classifier was able to classify with the highest generalization of 57.36% for the number of hidden nodes of 90. Again when thresholds smaller than 10^{-8} were used, the percent of correct prediction was

decreased. The proposed IFLN classifier was trained in one pass through all data with the average training time less than two seconds.

The vowel classification using various nonlinear classifiers is shown in Table 4.4. The comparison study was performed by Tony Robinson [103]. In Robinson's study, the best results with the correct prediction of 56% were reported using the nearest neighbor classifier. On the other hand, the IFLN can achieve 57.36%. The complete detail of Robinson's discussion is archived at http://www.boltz.cs.cmu.edu/benchmarks/vowel.html, CMU Repository of Neural Network Benchmarks.

TABLE 4.4

VOWEL CLASSIFICATION WITH DIFFERENT NONLINEAR CLASSIFIERS [103]

Classifiers	no.of hidden units	no. correct	percent correct
Single-layer perceptron	-	154	33
Multi-layer perceptron	88	234	51
Multi-layer perceptron	22	206	45
Multi-layer perceptron	11	203	44
Modified Kanerva Model	528	231	50
Modified Kanerva Model	88	197	43
Radial Basis Function	528	247	53
Radial Basis Function	88	220	48
Gaussian node network	528	252	55
Gaussian node network	88	247	53
Gaussian node network	22	250	54
Gaussian node network	11	211	47
Square node network	88	253	55
Square node network	22	236	51
Square node network	11	217	50
Nearest neighbor	-	260	56
ILFN	**90**	**265**	**57.36**

4.3 Two-Spiral Benchmark

Learning to tell two spirals apart is a neural network benchmark task proposed by Alexis Wieland in 1989. The objective is to learn to discriminate between two sets of training points that lie on two distinct spirals in the x-y plane. These spirals coil three times around the origin and around one another.

The training set exemplar sequence is $(\mathbf{p}_{(i)}, \mathbf{t}_{(i)})$, $i = 1, 2, \ldots, 194$ with $\mathbf{p}_i \in \Re^2$ and $\mathbf{t} \in \Re^1$. For $n = 0, 1, \ldots, 96$, the training patterns are obtained from the following equations:

$$\mathbf{p}_{(2n+1)} = [x_n \quad y_n]^{\mathrm{T}}, \tag{4.1}$$

$$\mathbf{t}_{(2n+1)} = [\,0\,], \tag{4.2}$$

$$\mathbf{p}_{(2n+2)} = [-x_n \quad -y_n]^{\mathrm{T}}, \tag{4.3}$$

$$\mathbf{t}_{(2n+2)} = [\,1\,]; \tag{4.4}$$

where

$$x_n = r_{\mathrm{n}} \cos(\theta), \tag{4.5}$$

$$y_n = r_{\mathrm{n}} \sin(\theta), \tag{4.6}$$

and

$$r_{\mathrm{n}} = 6.5\left(\frac{104-n}{104}\right), \tag{4.7}$$

$$\theta = \frac{\pi n}{16}. \tag{4.8}$$

From the above equations, 194 pairs of inputs and outputs in two dimensions were obtained. The classification task was to train the patterns in this data set to the ILFN classifier producing the correct outputs for all of the inputs. Figure 4.3 shows 194 patterns of the two-spiral problem.

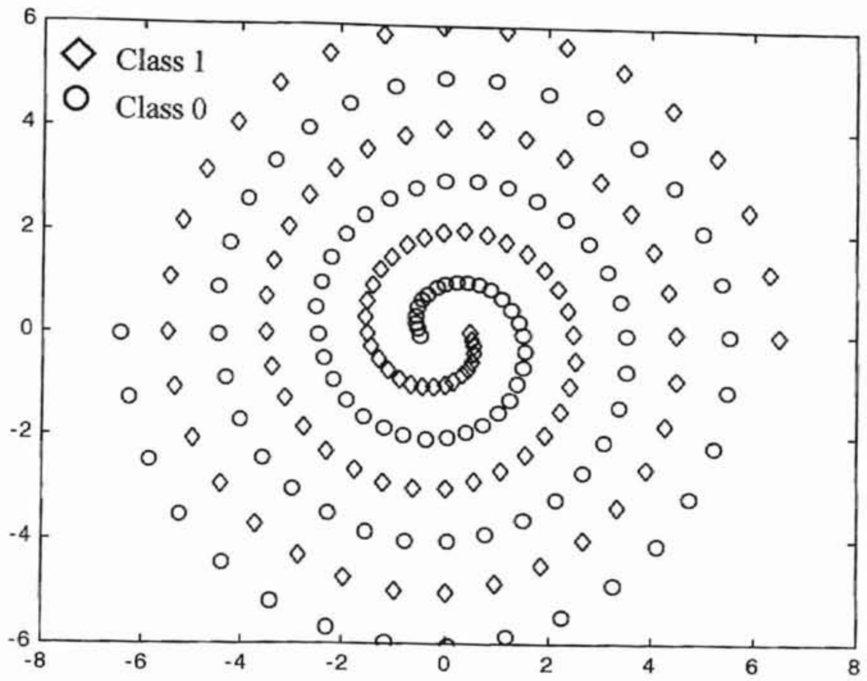


Figure 4.3 A Plot of the Two-Spiral Problem

4.3.1 The ILFN Experimental Result of the Two-Spiral Problem

TABLE 4.5

THE ILFN CLASSIFIER PERFORMANCE
ON TWO-SPIRAL DATA USING DIFFERENT THRESHOLDS

Threshold	# hidden nodes	training time (S)	% correct of training data
5×10^{-1}	146	0.72	100
5×10^{-13}	92	0.49	100
5×10^{-14}	90	0.49	100
5×10^{-15}	82	0.49	96.97
5×10^{-16}	78	0.49	94.85
5×10^{-17}	56	0.44	89.69
5×10^{-18}	52	0.44	86.60
less than 5×10^{-19}	46	0.39	77.32

Table 4.5 shows the results of the ILFN classifier on the two-spiral problem using different threshold values. For the threshold of $5x10^{-1}$, 146 nodes were generated with the result of 100% correction using the same training data as the testing data. In the experiment, the classifier generated fewer neurons when the threshold was reduced; however, the testing results still achieved 100% accuracy with the final threshold value of $5x10^{-14}$ and the hidden layer of 90 nodes. The minimum number of hidden nodes is 46, setting the threshold value to be less than or equal to $5x10^{-19}$ and achieving the correctness of 77.32%. It is worth noticing that our proposed classifier was able to distinguish between two spirals in only one epoch within less than one-second training time. Figure 4.4 depicts the two-spiral problem with the prototypes created by the ILFN when the threshold ε was set to be $5x10^{-14}$, obtaining 90 nodes in the hidden layer.

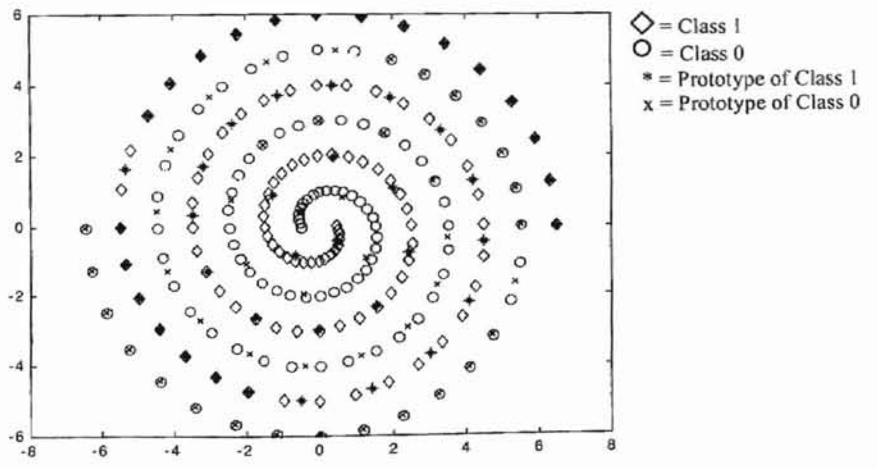


Figure 4.4 90 Prototypes Created by the ILFN for the Two-Spiral Problem

4.3.2 An MLP Experiment on the Two-Spiral Problem

To compare the speed performance with the ILFN, we also used the MLP trained by the Levenberg-Marquardt algorithm [48], one of the fastest training algorithms for the

MLP. The MLP network was constructed with one input layer with 2 neurons, two hidden layers with 20 neurons each, and one output layer with one neuron. Logsigmoidal functions were applied in the network. With the sum-squared error (SSE) of 0.001, the MLP network used a training time of 1.38 hours with 248 iterations to converge.

4.3.3 Results from other Researchers on the Two-Spiral Problem

Fahlman's quickprop algorithm with hyperbolic arctangent error [36] was used with the same network and starting values, and the training times were 4500 iterations, 12,300 iterations, and 6800 iterations with average of 7900 iterations [37]. (Note that each iteration probably took more than ten seconds.) More results on the two-spiral experiments from other researchers can be electronically achieved on the Internet at http://www.boltz.cs.cmu.edu/benchmarks/two-spirals.html [129].

Using the IFLN classifier, the results from the experiments used with the above three data sets were found to be competitive with many existing classifiers. However, the ILFN features an advantage with the capability of fast, on-line, incremental learning. This feature is very important for vibration monitoring systems that allow the classifier to learn new information during operation when new classes of failure modes are developing. Only new data (i.e., new classes) are added to the classifier without retraining the old data that have been learned. The ILFN was developed to handle this kind of problem. To demonstrate the performance of the ILFN on a vibration classification task, the experiments with the Westland vibration data set were performed as described in the next section.

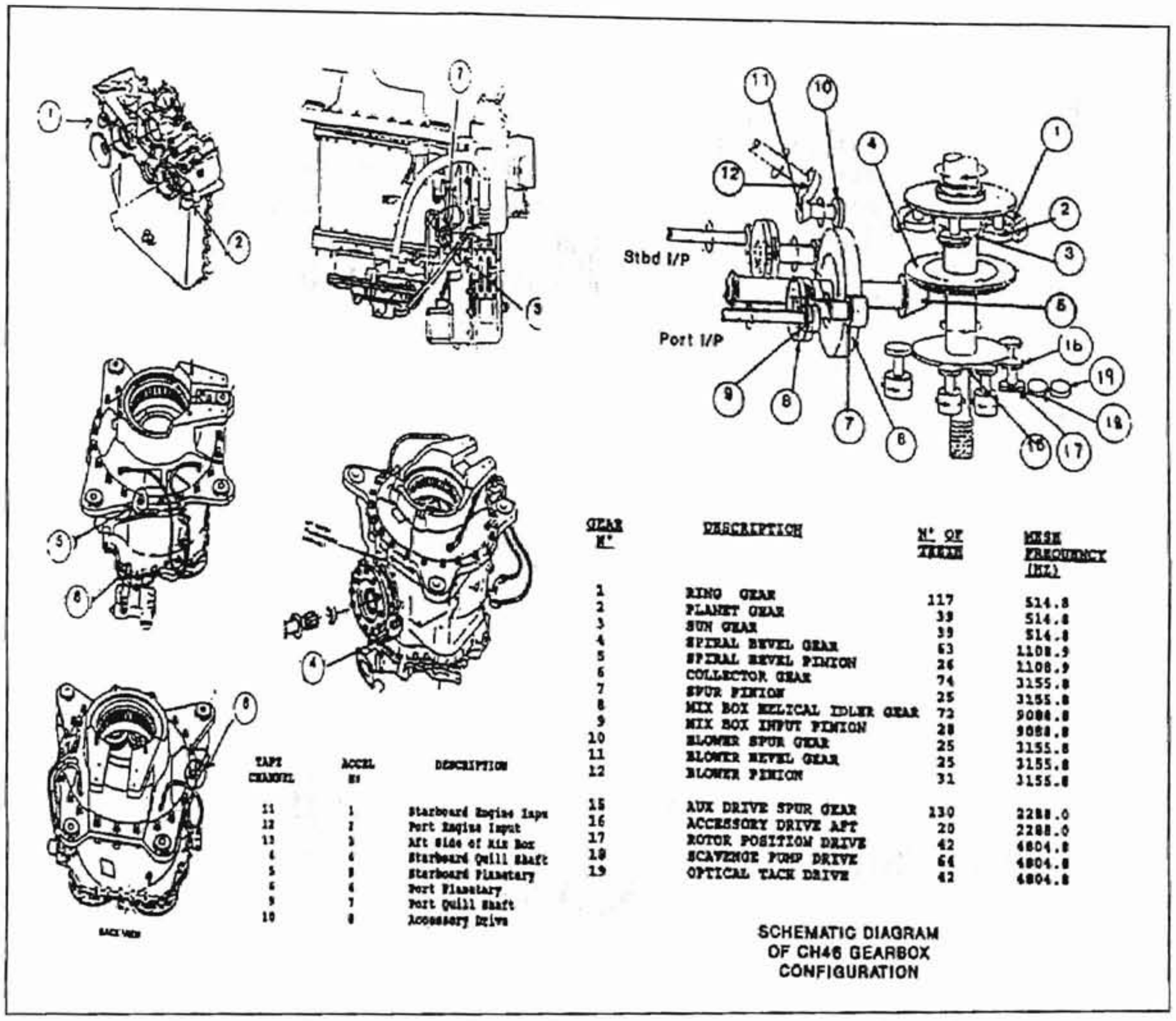
4.4 The Westland Vibration Data Set

This data set consists of vibration data recorded using eight accelerometers mounted on different locations (shown in Figure 4.5) on the aft main power transmission of a U.S. Navy CH-46E helicopter. The CH-46E Chinook is a twin-rotor, fore/aft transmission rotorcraft powered by two turbine engines. The data was archived at the Applied Research Laboratory (ARL) of Penn State University. The vibration data was collected by using an International Recording Instruments Group analog tape recorder and a single mixbox and aft main transmission installed on a test stand and run at nine different torque levels (i.e., 100%, 80%, 75%, 70%, 60%, 50%, 45%, 40%, 27%). While collecting the data, only one faulted component was installed in the mixbox and transmission and vibration data were recorded. The data were recorded for many types of faults listed in Table 4.6. Employing a 10-channel data acquisition system, the data were digitized at a sample rate of 103,116.08 Hz with 16-bit quantization level and were saved in 1.506-MB data files. All together, there are 71 files; each file contains all eight accelerometer signals. The data files used in this study were one-second data files [16].

TABLE 4.6

A LIST OF THE FAULT TYPES
CREATED IN THE TEST GEARBOX

Fault #	Description
2	Epicyclic Planet Gear Bore/Bearing/Inner Race Corrosion Spalling
3	Spiral Bevel Input Pinion Bearing Journal Corrosion Pitting/Spalling
4	Spiral Bevel Input Pinion Gear Tooth Spalling/Scuffing
5	High Speed Helical Input Pinion Tooth Chipping and Freewheel Clutch Bearing False Brinnelling
6	Helical Idler Gear Crack Propagation
7	Collector Gear Crack Propagation
8	Quill Shaft Crack Propagation
9	No Defect



TAPE CHANNEL	ACCEL N°	DESCRIPTION
11	1	Starboard Engine Inpu
12	2	Port Engine Input
13	3	Aft Side of Mix Box
4	4	Starboard Quill Shaft
5	5	Starboard Planetary
6	6	Port Planetary
9	7	Port Quill Shaft
10	8	Accessory Drive

GEAR N°	DESCRIPTION	N° OF TEETH	MESH FREQUENCY (HZ)
1	RING GEAR	117	514.8
2	PLANET GEAR	39	514.8
3	SUN GEAR	39	514.8
4	SPIRAL BEVEL GEAR	63	1108.9
5	SPIRAL BEVEL PINION	26	1108.9
6	COLLECTOR GEAR	74	3155.8
7	SPUR PINION	25	3155.8
8	MIX BOX HELICAL IDLER GEAR	72	9088.8
9	MIX BOX INPUT PINION	28	9088.8
10	BLOWER SPUR GEAR	25	3155.8
11	BLOWER BEVEL GEAR	25	3155.8
12	BLOWER PINION	31	3155.8
15	AUX DRIVE SPUR GEAR	130	2288.0
16	ACCESSORY DRIVE APT	20	2288.0
17	ROTOR POSITION DRIVE	42	4804.8
18	SCAVENGE PUMP DRIVE	64	4804.8
19	OPTICAL TACH DRIVE	42	4804.8

Figure 4.5 Accelerometer Positions on the Aft Transmission of the CH-46E Helicopter

4.4.1 Westland Data Characteristics

Figures 4.6 and 4.7 show two samples of vibration data in time domain pertaining to fault Class 2 and Class 3 from Accelerometer 1 of the Westland Data Archive. However, it is difficult to discriminate the two raw time-series data. The raw time series data provide little information to use for classification.

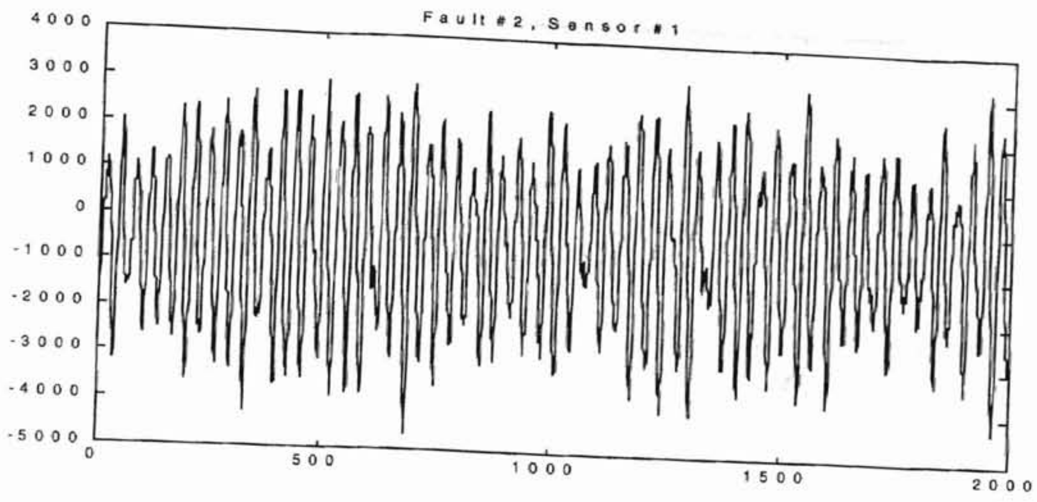


Figure 4.6 A Plot of Time Series Data of Fault 2 from Sensor 1

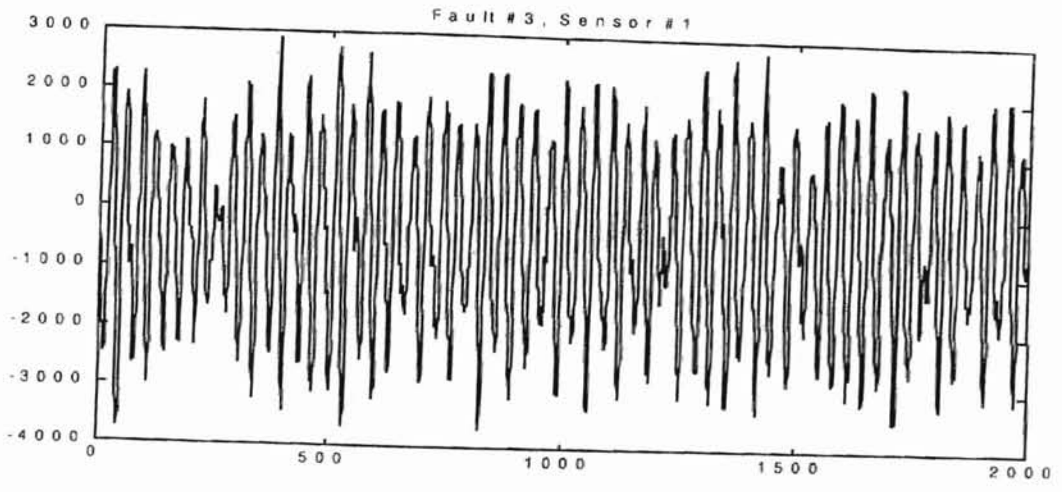


Figure 4.7 A Plot of Time Series Data of Fault 3 form Sensor 1

It is preferable to transform the signal from time domain to frequency domain. The vibration signatures in frequency domain are shown in Figure 4.8 and Figure 4.9, which are power spectral density plots of the two signals given in Figure 4.6 and Figure 4.7, respectively. It is easy to see that frequency contents above 20 kHz are less useful. The effective information for classification is in frequency range of 3 kHz to 10 kHz.

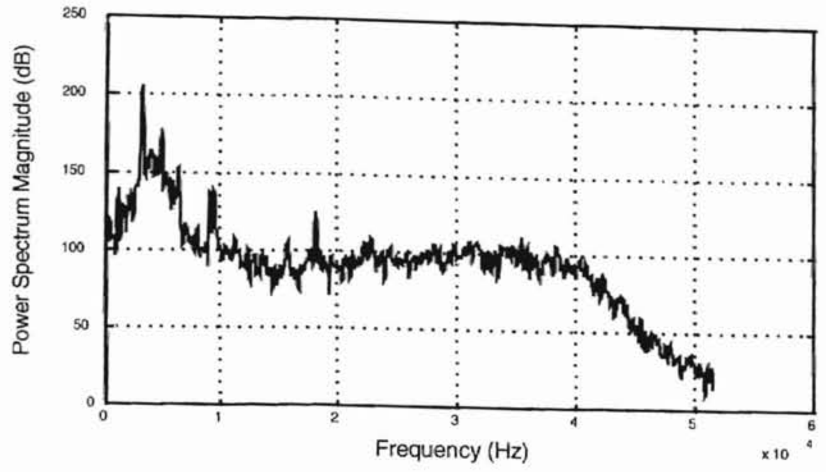


Figure 4.8 Power Spectrum Density dB Plot of Fault 2 from Sensor 1

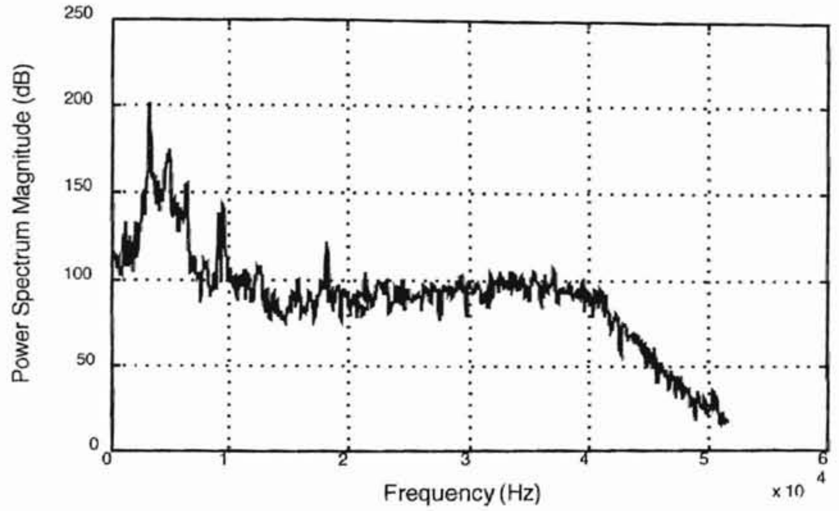


Figure 4.9 Power Spectrum Density dB Plot of Fault 3 from Sensor 1

For the interested frequency range of 0-12 kHz, Figures 4.10 and 4.11 illustrate a "zoom-in" version of the power spectrum density plot shown in Figures 4.8 and 4.9, respectively. These two signatures from Sensor 1 seem difficult to separate. Figure 4.12 illustrates the power spectrum density plot of Fault 2 from Sensor 3. Figure 4.13 represents the power spectrum density plot of Fault 3 from Sensor 3. The two signatures from Sensor 3 are relatively easy to classify.

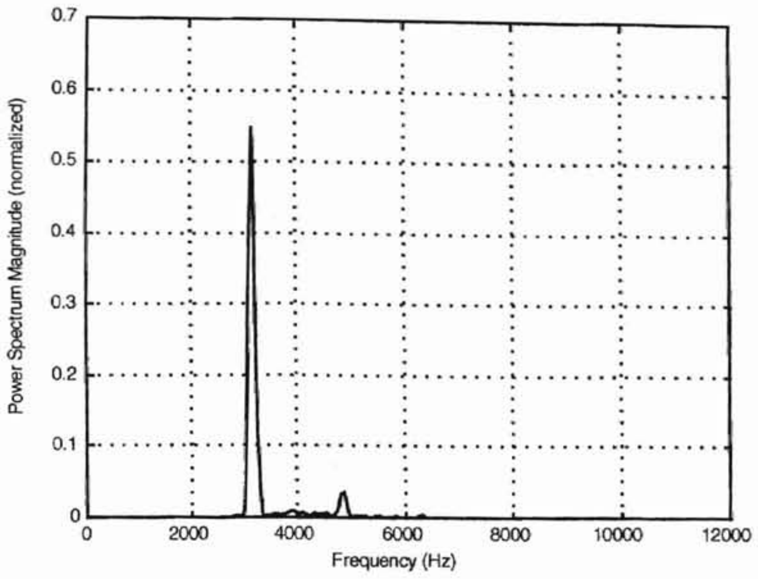


Figure 4.10 Power Spectrum Density Plot of Fault 2 from Sensor 1

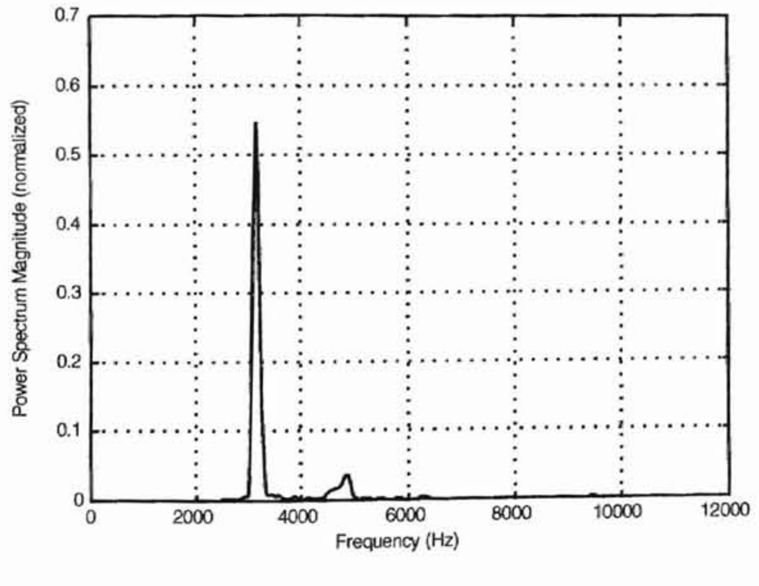


Figure 4.11 Power Spectrum Density Plot of Fault 3 from Sensor 1

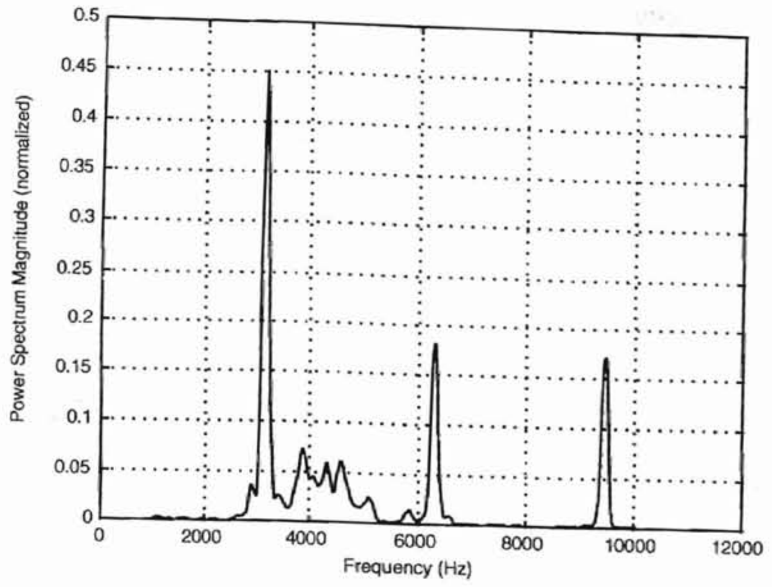


Figure 4.12 Power Spectrum Density Plot of Fault 2 from Sensor 3

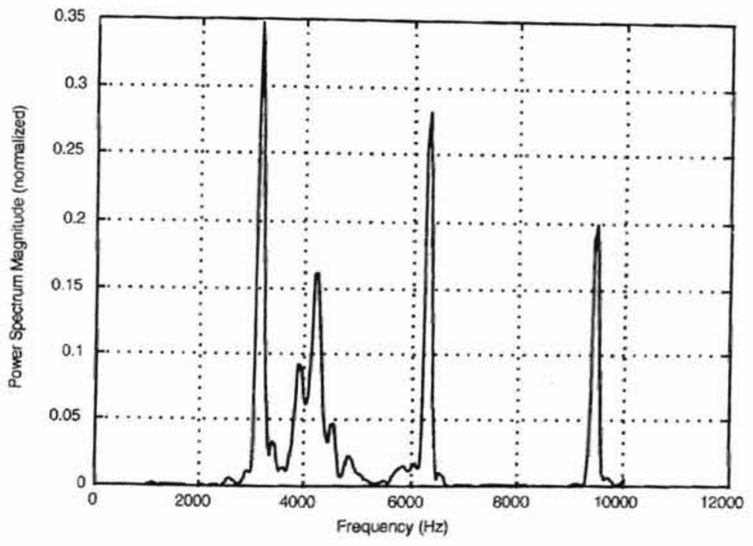


Figure 4.13 Power Spectrum Density Plot of Fault 3 from Sensor 3

More sample plots on frequency domain of 100% torque level of Westland vibration data are shown in Figures 4.14 and 4.15. Figures 4.14 and 4.15 show sample patterns of Fault 2, Fault 3, Fault 4, Fault 5, Fault 6, Fault 7, Fault 8, and no fault from all 8 accelerometers.

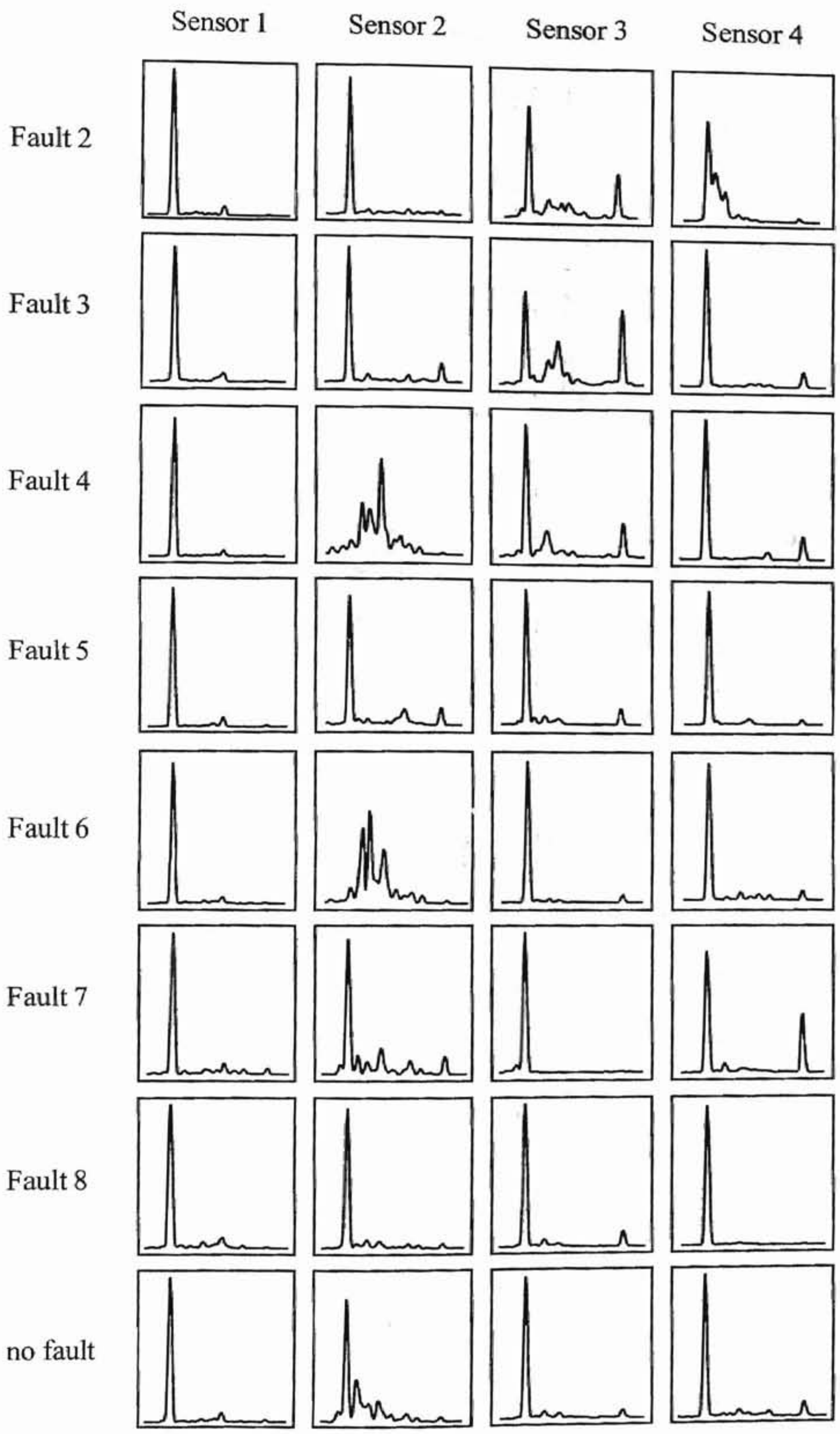


Figure 4.14 Power Spectrum Density Plot of 100% Torque Load with Different Faults from 8 Sensors

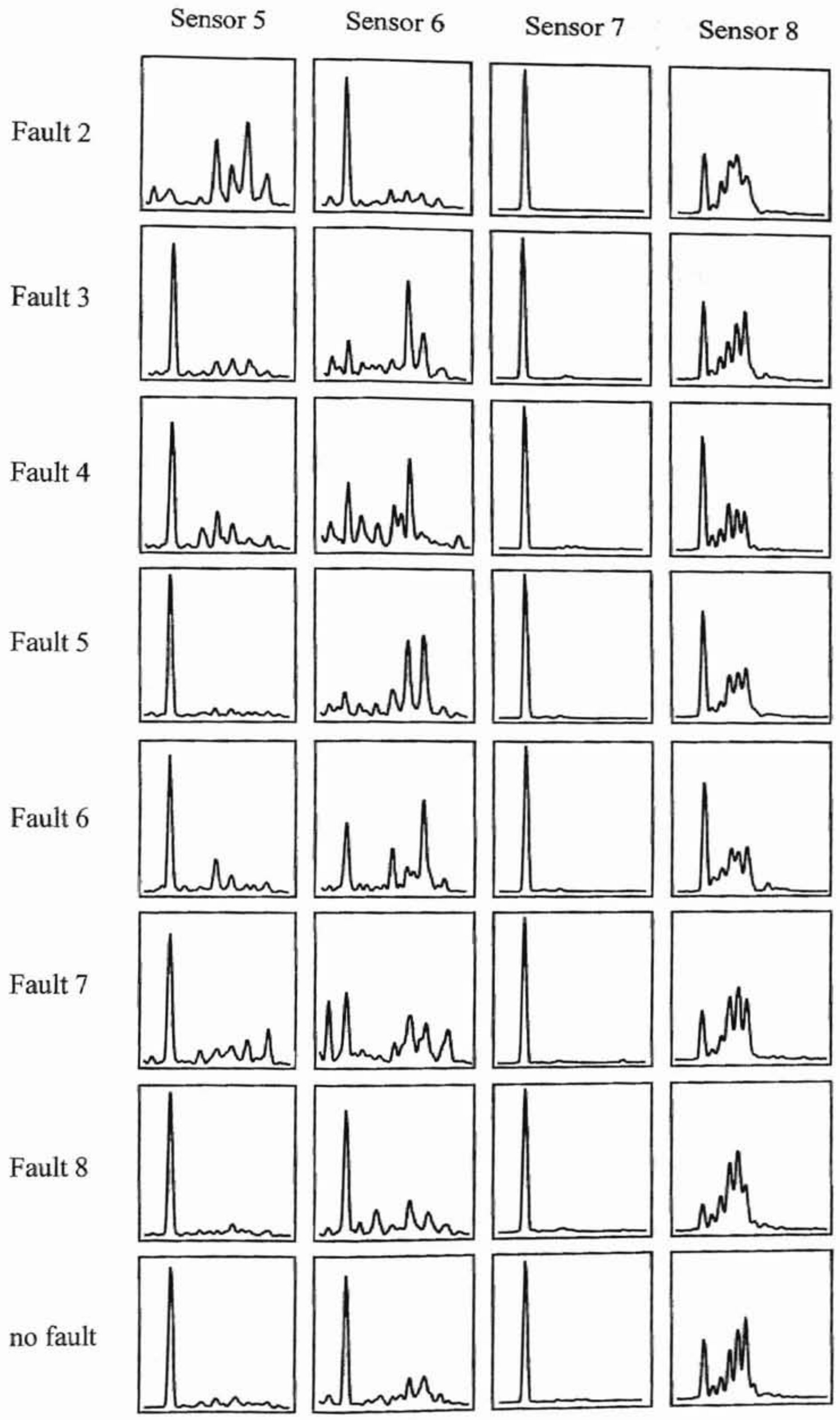


Figure 4.15 Power Spectrum Density Plot of 100% Torque Load with Different Faults from 8 Sensors (continue).

It is worth noting that data from each sensor alone is not sufficient to classify the fault classes. Moreover, it is easier to classify the data by using all patterns obtained from 8 sensors. Integrating fault patterns from all 8 accelerometers, it is more informative for classification as shown in Figure 4.16 through Figure 4.23. Figure 4.16 through Figure 4.23 show frequency domain plots of all fault classes that are pattern vectors combined from 8 sensors. All signatures in Figure 4.16 through Figure 4.23 are more informative and easier to distinguish. In this study, most of our experiments used the combined signatures from all 8 sensors as training patterns, except the comparison with the Fuzzy ARTMAP neural network for which only data from Sensor 1 was used.

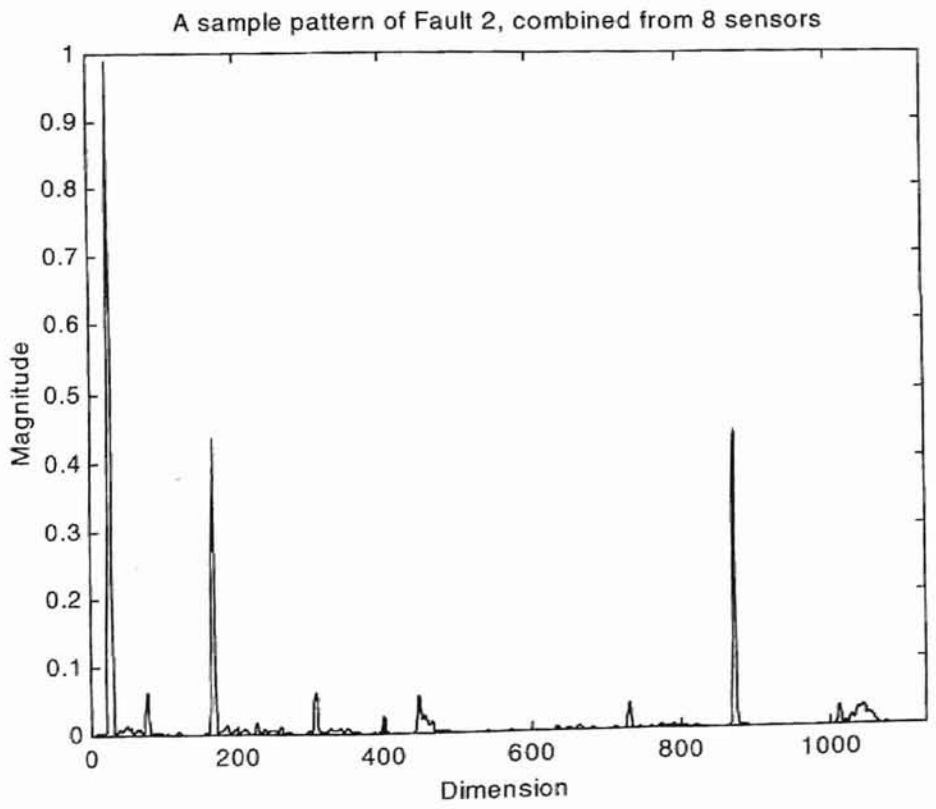


Figure 4.16 A Frequency Domain Plot of Fault 2, Combined from 8 Sensors

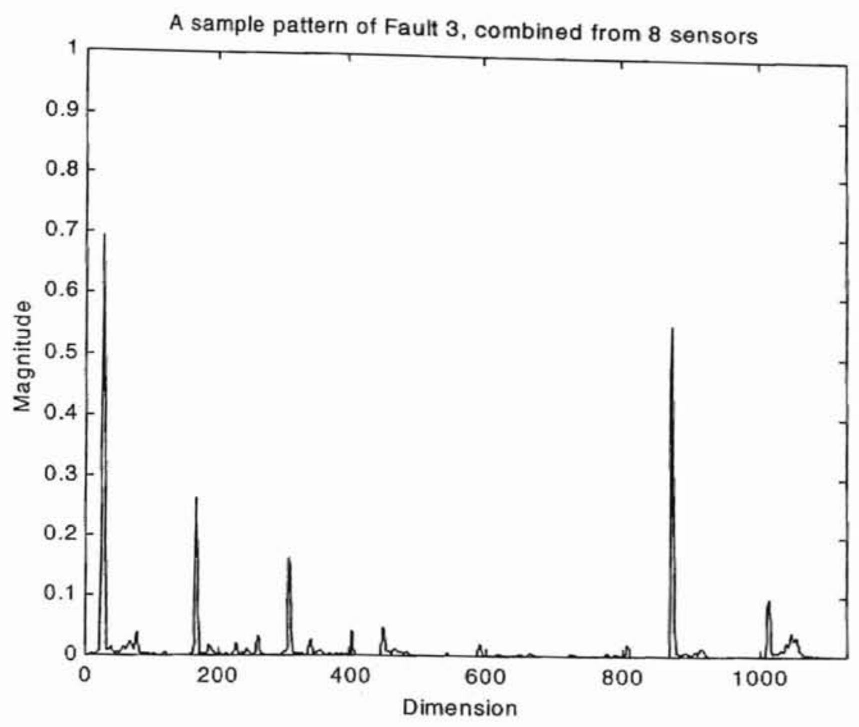


Figure 4.17 A Frequency Domain Plot of Fault 3, Combined from 8 Sensors

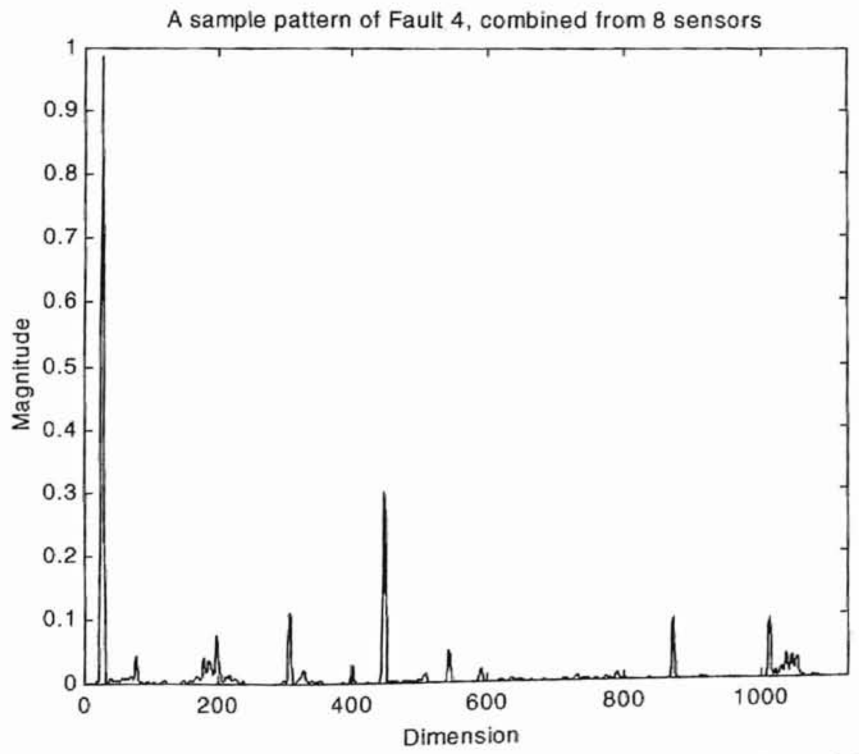


Figure 4.18 A Frequency Domain Plot of Fault 4, Combined from 8 Sensors

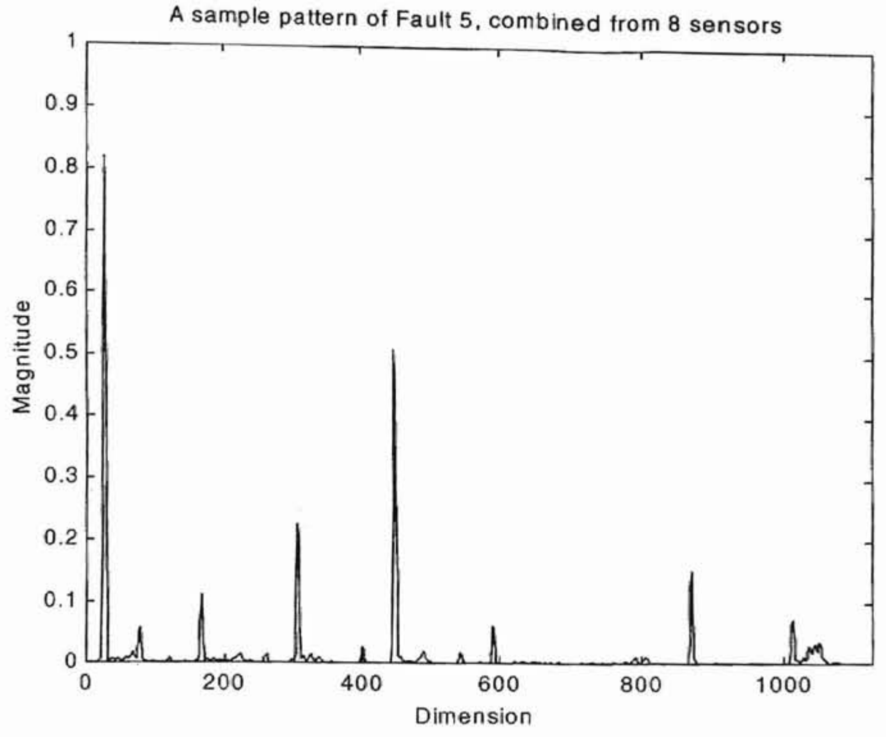


Figure 4.19 A Frequency Domain Plot of Fault 5, Combined from 8 Sensors

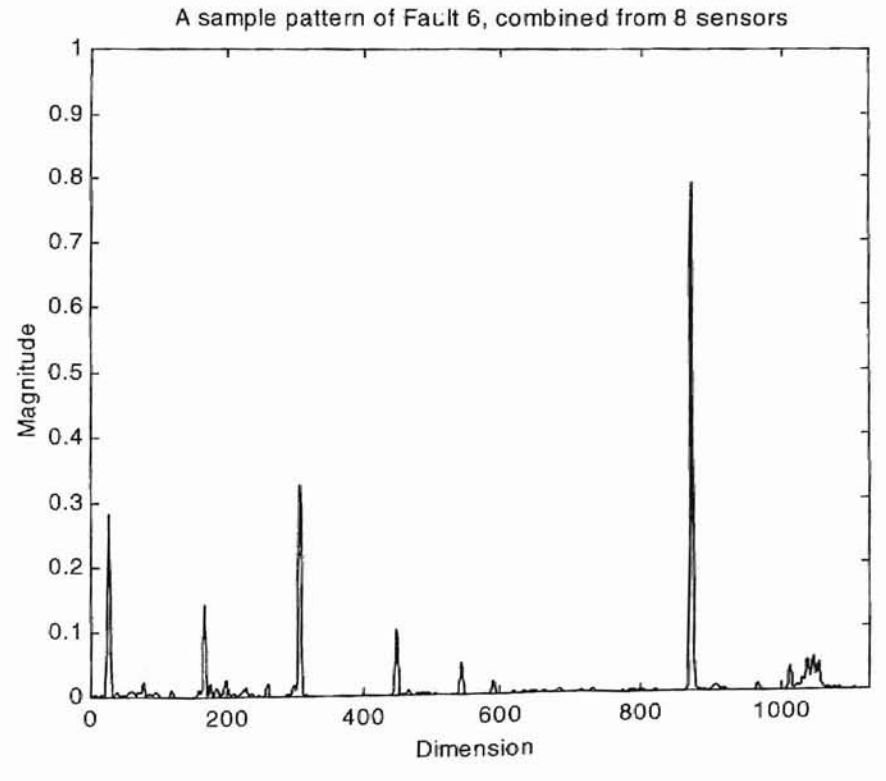


Figure 4.20 A Frequency Domain Plot of Fault 6, Combined from 8 Sensors

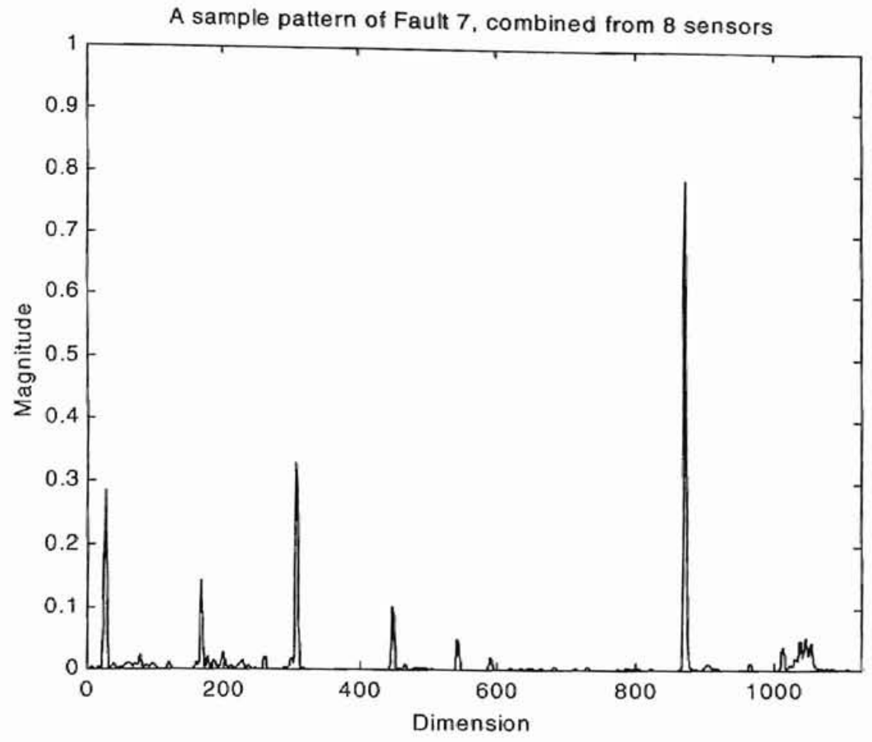


Figure 4.21 A Frequency Domain Plot of Fault 7, Combined from 8 Sensors

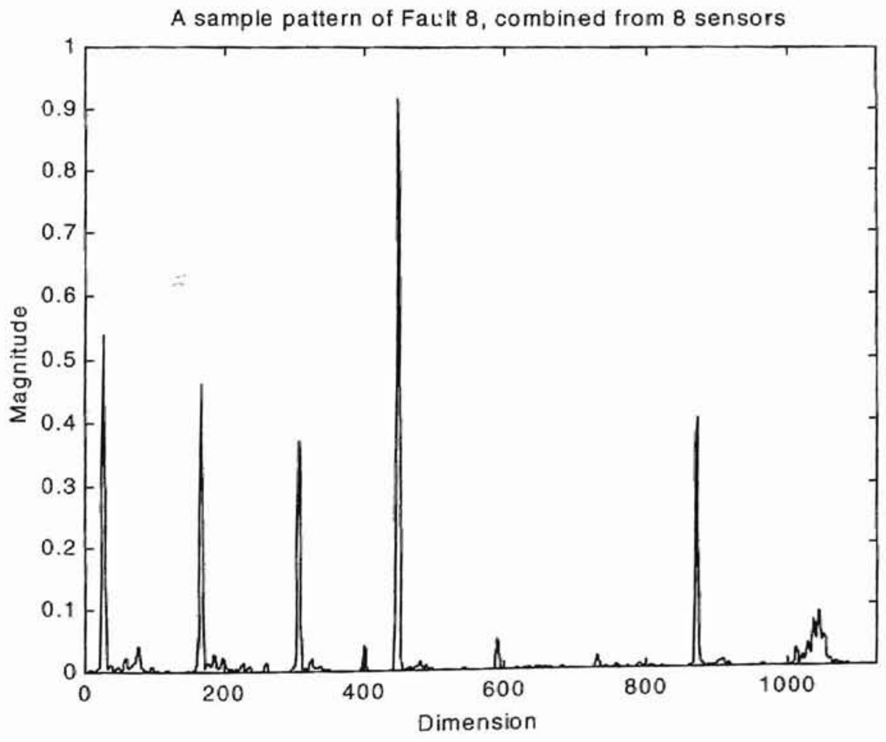


Figure 4.22 A Frequency Domain Plot of Fault 8, Combined from 8 Sensors

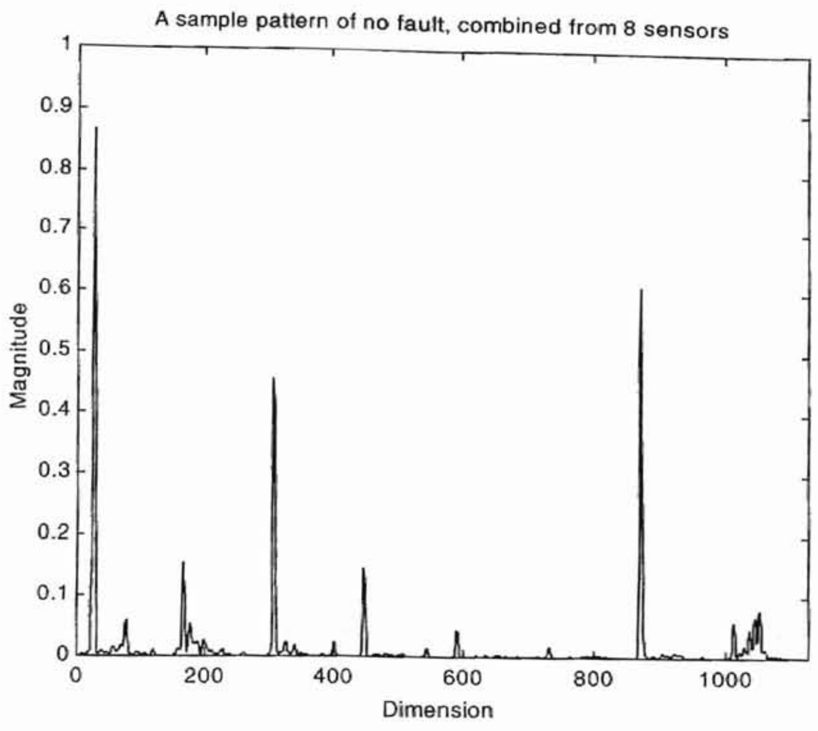


Figure 4.23 A Frequency Domain Plot of No Fault, Combined from 8 Sensors

4.4.2 Using ILFN Classifier on Westland Vibration Data

In our experiments, vibration time-series data were preprocessed using FFT technique in transformation from time domain to frequency domain. Power spectrum command (SPECTRUM in Matlab Signal Processing Toolbox) with Hanning window of 1024 samples was utilized. We filtered the data with the interested frequency band of 3 kHz - 10 kHz, getting a 141x1 vector for each channel. Vectors from 8 channels were set into one vector (1128x1 vector, as shown in Figure 4.16 through Figure 4.23). Then, it was input into the ILFN classifier as well as other classifiers used in this study (except the Fuzzy ARTMAP which used only Channel 1). The Fault types and torque levels of the Westland vibration data used in the experiments are shown in Table 4.7 and Table 4.8. The summary results from the experiments are given in Table 4.9.

TABEL 4.7

FAULT TYPES AND TORQUE LEVELS OF WESTLAND VIBRATION DATA USED IN THE EXPERIMENTS

FAULT TYPES	TORQUE LEVELS								
	100%	80%	75%	70%	60%	50%	45%	40%	27%
2	USED	NA	NA	NA	NA	NA	NA	NA	NA
3	USED	USED	USED	USED	USED	USED	USED	USED	USED
4	USED	USED	USED	USED	USED	USED	USED	USED	USED
5	USED	USED	USED	USED	NA	NA	NA	NA	NA
6	USED	USED	USED	USED	NA	NA	NA	NA	NA
7	USED	USED	USED	USED	USED	USED	NA	USED	USED
8	USED	USED	USED	USED	USED	USED	USED	USED	USED
9 (no fault)	USED	USED	USED	USED	USED	USED	USED	USED	USED

NA=Not available

TABEL 4.8

TORQUE LEVELS AND THE NUMBER OF PATTERNS USED IN THE EXPERIMENTS

Torque levels	100%	80%	75%	70%	60%	50%	45%	40%	27%
# patterns	400	700	350	700	500	500	400	500	500

Table 4.9 shows the results from all experiments in our studies. In the experiments of the Westland data set, all torque load levels (i.e., 100%, 80%, 75%, 70%, 60%, 50%, 45%, 40%, and 27%) were used to train the ILFN classifier. Only 10 patterns were used for training, and the remaining data were used for testing when the same torque level was used for both training and testing. All patterns were used for training when different torque load levels were used for testing. For the last column of Table 4.9, the training set was composed from the first 10 patterns of each torque level.

TABEL 4.9

PERCENT CORRECT CLASSIFICATION OF
THE ILFN FOR THE WESTLAND VIBRATION DATA
WITH DIFFERENT TORQUE LEVELS FOR TRAINING AND TESTING

		TRAINING DATA (TORQUE LEVELS)									
		100%	**80%**	**75%**	**70%**	**60%**	**50%**	**45%**	**40%**	**27%**	**all torque levels**
	Hidden nodes	8	7	7	7	5	5	4	5	5	51
TESTING DATA (TORQUE LEVELS)	**100%**	**100**	60.57	35	36	40	40	50	30.8	40	**100**
	80%	71.43	**100**	71.29	59.71	50	33.33	36.36	33.33	33.33	**100**
	75%	81.42	71.43	**100**	96.29	40	33.33	36.36	33.33	39	**100**
	70%	57.14	71.43	100	**100**	80	33.33	36.36	33.33	39	**100**
	60%	41.2	48.6	80.4	93.2	**100**	92.6	56	40	78.2	**100**
	50%	37.2	59.8	80	80	99.8	**100**	100	100	100	**100**
	45%	42.5	50	50	53.5	81.25	100	**100**	100	95.75	**100**
	40%	4	40	60	59.3	80.2	100	100	**100**	100	**100**
	27%	7.4	42.4	60	60	80	80.6	91.25	100	**100**	**100**

Remark: (1) Only 10 patterns were used for training when the same torque load was used for testing.
(2) All patterns were used for training when the different torque load were used for testing.
(3) For the last column, only 10 patterns from each torque load level were used for training.

In Table 4.9, the columns represent the training data with different torque levels, and the rows indicate the testing data with different torque levels. The percent of correct classification results are interpreted by crossing each column with each row. For instance, 100% correct classification was achieved when the ILFN was trained by the 40% torque level and was tested by the 50% torque level. The numbers of hidden neurons resulting from the training process of the ILFN are shown in Table 4.9 in the row leading with the words "Hidden nodes." These numbers indicate how many prototypes that the ILFN classifier has created.

Using 10 patterns of the 100% torque level for training, the classifier created 8 neurons in the hidden layer. The ILFN obtained 100% correct classification using the remaining data of the same torque load for testing. Moreover, using all 400 patterns of the 100% torque level for training, the ILFN also created 8 neurons in the hidden layer. The other torque levels were used to test the "robustness" of the ILFN network. The correct classification of 71.43% was achieved for the 80% torque load, and for the 75% torque load, 81.42% correct prediction was obtained. The ILFN classifier yielded the correct classification of 57.14%, 41.2%, 37.2%, 42.5%, 4%, and 7.4% for torque levels of 70%, 60%, 50%, 45%, 40%, and 27%, respectively.

It is worth noticing that when the same torque level was used both for training and testing, the ILFN achieved 100% correct classification. (Note that testing patterns were different from the training patterns, i.e., obtained from different time series, but they were in the same torque level.) Furthermore, using high torque levels (i.e., 100% and 80%) for training, in the testing phase the ILFN achieved perfect classification only when the testing patterns from the same torque level were used. However, using 75%, 70%, 60%, 50%, 45%, 40%, or 27% torque level for testing, the ILFN was able to correctly classify a larger range of torque levels. For example, when the ILFN was trained by a 50% torque level, 100% correct classification was obtained from the range of 40% through 60% torque levels.

4.4.3 A Comparison between the Fuzzy ARTMAP and the ILFN

Another experiment was performed to compare the classification performance between the Fuzzy ARTMAP [18] and the ILFN classifier since both of them are supervised incremental learning algorithms. Data from each sensor alone was used to train and test the networks because using data from an individual sensor was more difficult to classify than using combined data from all sensors. 200 patterns from each channel of 100% torque load were used for training, and the other 200 patterns were used for testing. For this experiment, each pattern from each sensor was a 141-dimensional vector.

The Fuzzy ARTMAP was set at the vigilance parameters $\rho_a = 0.7$ and $\rho_b = 0.7$ and the learning rate $\beta = 1$. The ILFN was set at the threshold $\varepsilon = 0.001$ and the initial standard deviation $\sigma_0 = 0.001$. Table 4.10 shows the result of this experiment.

TABLE 4.10

A PERFORMANCE COMPARISON BETWEEN THE FUZZY ARTMAP AND THE ILFN (USING WESTLAND VIBRATION DATA FROM INDIVIDUAL SENSORS)

		Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6	Sensor 7	Sonsor 8
Fuzzy	Hidden nodes	a = 9, b = 4	a = 4, b = 4	a = 4, b = 4	a = 4, b = 4	a = 4, b = 4	a = 4, b = 4	a = 4, b = 4	a = 4, b = 4
ARTMAP	Learning iter.	2	2	2	2	2	2	2	2
$\rho a = 0.7$ $\rho b = 0.7$	% correct on testing data	64	80	90.5	84	67	57	72.5	43.5
ILFN	Hidden nodes	15	12	8	12	13	10	12	15
$\varepsilon = 0.001$	Learning iter.	1	1	1	1	1	1	1	1
$\sigma 0 = 0.001$	% correct on testing data	79.5	79.5	95.5	98.5	50	78.5	91	78.5

Table 4.10 shows a comparison of classification performance between the Fuzzy ARTMAP and the ILFN classifier. Using data from each sensor alone, the Fuzzy ARTMAP (with the specific parameters) was able to classify the testing data with the highest correct classification of 90.5% from Sensor 3. It resulted in the lowest correct classification of 43.5% using Sensor 8. The results were varied when the Fuzzy ARTMAP performed on other sensors. (Note that the Fuzzy ARTMAP is sensitive to the parameter. By changing the values of the parameter, the correct classification may be

significantly changed up or down. A larger vigilance value yields higher correct classification.)

On the contrary, the ILFN was able to classify the data from Sensor 4 and Sensor 3 with 98.5% and 95.5% correct classification, respectively. The lowest correct classification was on Sensor 5 with 50% correct classification. The Fuzzy ARTMAP performed better than the ILFN did on Sensor 5. However, on the other sensors, the ILFN classifier was able to classify the testing data with higher correct classification than did the Fuzzy ARTMAP. The number of hidden neurons of the ILFN was higher than that of the Fuzzy ARTMAP. However, the number of hidden neurons of the ILFN was automatically grown controlled by the threshold ε, and it was not sensitive to ε. Changing the value of ε had little effect on the ILFN performance.

4.4.4 Comparisons among Other Classifiers

More experimental results on Westland vibration data are shown in Table 4.11. Table 4.11 shows the comparison among the Multilayer Perceptorn (MLP), Radial Basis Function Network (RBFN), Learning Vector Quantization (LVQ), and ILFN classifier. This experiment was performed using 200 patterns of 100% torque levels to train each classifier. The testing data sets were composed of 200 patterns from 100% torque load, 700 patterns from 80% torque, 350 patterns from 75% torque, and 700 patterns from 70% torque load. The data used were 1128-dimensional vectors that were combined from all 8 sensors.

The first network was the MLP trained by the Backpropagation (BP) with variable learning rates. The MLP was comprised of one hidden layer with 10 hidden nodes and one output layer with 4 nodes. Logsigmoidal functions were utilized in the MLP network. The sum of square error (SSE) goal was set to 0.001. The MLP was trained for 10 trials. To meet the SSE goal, the MLP network used a training time of 475 iterations with 400 seconds on the average of 10 runs. We noticed that for many trials the MLP was stuck at some local minima, unable to converge to the global minimum.

TABEL 4.11

PERCENT CORRECT CLASSIFICATION OF THE MLP, RBFN, LVQ, AND ILFN, TRAINED BY 100% TORQUE LEVEL AND TESTED BY DIFFERENT TORQUE LEVELS

		Classifier types (trained with 100% torque level)			
		MLP	RBFN	LVQ	ILFN
	Learning time	400S 475 epochs	12 S 1 epoch	194 S 500 epochs	4 S 1 epoch
Testing data (torque level)	100%	96.5	100	100	100
	80%	58.71	4.57	71.43	71.43
	75%	61.14	0.57	74.29	74
	70%	37.43	0.57	57.14	57.44
	Learning type	OFF-LINE	OFF-LINE	OFF-LINE	ON-LINE

MLP => 1 hidden layer, 10 hidden neurons; trained by BP with variable learning rate; sum square error goal (SSE) = 0.001
RBFN => 8 hidden neurons determined by one-pass clustering; 4 output neurons.
LVQ => 8 hidden neurons; 4 ouput neurons.

The second network was the RBF network. Using one-pass self-selection of the hidden centers by a successive approximation method [79], the RBFN constructed 8 hidden neurons in the hidden layer. Then, the output weight was determined using the method proposed by Haykin [52]. The RBFN quickly learned within a single iteration.

The third network was the LVQ network. The LVQ network of Neural Network Toolbox version 2.0.4 of Matlab version 5.1 was used in this study. The network was composed of an 8-neuron LVQ layer and a 4-neuron linear layer. The maximum training time of the LVQ was set to be 500 iterations. The LVQ used approximately 194 seconds for training.

The ILFN incrementally learned and generated 8 neurons in the hidden layer. The training time was about 4 seconds within a single iteration. On this data set the ILFN used training time approximately 100 times, 3 times, and 64 times faster than the MLP, the RBFN (constructed in this experiment), and the LVQ, respectively.

For the generalization capacity, it was shown that the ILFN was competitive with the LVQ. In Table 4.11, both the ILFN and the LVQ were able to classify the 100%-torque-load testing data with 100% correct classification. The percent correct classification of the two networks was reduced to 71%, 74%, and 57% when using 80%, 75%, and 70% torque load, respectively. However, considering the capability of on-line, real-time, incremental learning, the ILFN was superior to the LVQ. Moreover, based on generalization and fast on-line learning ability, the ILFN was superior to the MLP and to the RBFN off-line learning algorithms.

4.4.5 An ILFN Learning Simulation in an Operating Mode

To study the ability of the ILFN in an operating mode, we used 100% torque load to train the ILFN network. First, only a "No Fault" class was trained to the network. Acting as a monitoring system, the ILFN repeatedly received unseen patterns in order to classify them. The ILFN was able to detect new classes, and it learned the incoming faults by creating new neurons and designated new targets for the unseen patterns that were significantly different than the patterns that had been learned as shown in Table 4.12.

Table 4.12 illustrates the performance of the ILFN in an operating mode. In Table 4.12, first the ILFN was trained using class "No Fault" with the corresponding target "0000." Then, patterns from Fault 8, Fault 5, Fault 2, Fault 3, Fault 6, Fault 7, and Fault 4 were presented to the ILFN network without targets. The ILFN assigned targets to be "0001," "0010," "0011," "0100," "0101," "0110," and "0111," respectively. In order to have different targets with the existing targets, first the ILFN classifier checked the existing targets finding the highest number in the target module. Then, using the increment of the highest number by one, the ILFN classifier assigned the new target to the incoming pattern.

TABLE 4.12

THE ILFN ASSIGNED CLASSES TO
THE UNSEEN PATTERNS (IN A BINARY FORMAT)

	Faults	**Labeled classes**			
Learned Fault	No fault	0	0	0	0
Unseen faults	Fault 8	0	0	0	1
	Fault 5	0	0	1	0
	Fault 2	0	0	1	1
	Fault 3	0	1	0	0
	Fault 6	0	1	0	1
	Fault 7	0	1	1	0
	Fault 4	0	1	0	0

CHAPTER V

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions of the Research

A new algorithm based on fuzzy neural networks called "Incremental Learning Fuzzy Neuron Network" (ILFN) has been developed for pattern classification. The ILFN employs a hybrid supervised and unsupervised learning scheme to generate its prototypes. The network is a self-organized classifier with the capability of adaptive learning of new information without forgetting existing information. The classifier can detect new classes and update its parameters while in an operation mode. Moreover, it utilizes fast real-time on-line learning without knowing *a priori* information (i.e., without knowing the probability density functions of pattern classes). In addition, it has the capability to make both soft (fuzzy) and hard (crisp) decisions and is able to classify both linear separable and non-linear separable problems.

The network is a synergetic combination of fuzzy sets and neural networks. It employs the fast parallel computation and learning capability of neural networks. In addition, fuzzy set theory adds the ability to represent and manipulate imprecise information.

The ILFN consists of two subsystems: one input subsystem and one target subsystem. Each subsystem is comprised of three layers: an input layer, a hidden layer, and an output layer. Input patterns are presented to the input subsystem while the corresponding targets are presented to the target subsystem during training. The learning process occurs in the hidden layer of the system. The two subsystems are linked together via the decision layer where the output of the network is obtained.

The hidden layer of the input subsystem employs the Gaussian radial basis functions as a fuzzy membership function. The membership function is used to fuzzify the distance between input patterns and prototypes into the membership domain. If the degree of membership function of the distance between a pattern to a prototype closes to "1," the pattern likely belongs to the prototype. On the contrary, if the degree of membership function closes to "0," this indicates the pattern is different from the prototype. With different degrees of membership function, input patterns are considered to have various grades of similarity to the prototypes. Hence, a given input pattern is allowed to have many class prototypes with different degrees.

For a pattern space with the bounded number of categories, even if the number of exemplars in the pattern space is unbounded, the ILFN surely converges to limits in response to an arbitrary sequence of input vectors. Convergence of the ILFN are satisfied because a finite number of categories in the hidden layer of the input subsystem are generated and the prototype vectors are updated to include new information to move only toward the centroids of the categories. The prototypes always move toward the centroids of the categories since the statistical mean of data is used. For every new input data satisfying the criterion to have the same category, a new statistical mean will be calculated to include the new input to the category. These prototype vectors constitute the *Voronoi tessellation* making the system to classify a given pattern to a correct class in the *Voronoi set.*

Four benchmark data sets: the Fisher's Iris data set, a vowel data set, the two-spiral problem, and the Westland vibration data set, were used in simulation experiments to demonstrate the performance of the ILFN classifier. Comparisons between the ILFN and some existing methods were made. The results show that, in terms of classification performance, the ILFN is competitive with or even better than many well-known classifiers, including the MLP, the RBFN, the LVQ, and the Fuzzy ARTMAP classifier. For example, in the classification performance comparison between the ILFN and the Fuzzy ARTMAP on the Fisher's Iris data set, the ILFN achieved 96.268% correct classification while the Fuzzy ARTMAP achieved 93.467%, on average of 20 trials. Moreover, in the comparisons among the MLP, RBFN, LVQ, and ILFN using the Westland data set with 100% torque level for training and testing, 96.5%, 100%, 100%, and 100% correct classification were achieved by the MLP, RBFN, LVQ, and ILFN, respectively. When using 100% torque level for training and 80% for testing, the MLP, RBFN, LVQ, and ILFN achieved 58.71%, 4.57%, 71.43%, and 71.43%, respectively.

Additionally, in terms of training time, the ILFN is superior to those classifiers. Furthermore, the on-line, real-time, one-pass, incremental learning behavior supports ILFN in its ability to detect new classes and update its parameters without using old data to retrain the network. The ILFN classifier, acting as a component in a monitoring system, was used extensively to investigate the Westland vibration data. The results from the simulation studies have shown that the real-time and on-line ILFN classifier is efficient for fault classification and identification in machine condition monitoring.

5.2 Recommendation of Possible Future Work

A disadvantage of the ILFN is that there is no mathematical proof of the convergence. This network seems less sensitive to the order of presentation of the sample feature vectors than does the Fuzzy ARTMAP (as shown from the experiment); however, it is still dependent upon the order of presentation as do other one-pass learning algorithms. These problems are left for future research projects.

Another possible future work on the ILFN system is to add a capability of knowledge representation, ability to translate knowledge induced by the ILFN network into linguistic rules that can be easily comprehended by human system operators. In addition, research to merge an adaptive feature extraction system into the ILFN classifier to enable the ILFN system to higher performance is needed. In this study, a simple FFT technique was used for feature extraction. It was found that the FFT technique was very sensitive to the torque level; i.e., different torque levels generated significantly different patterns for the same faults resulting in incorrect classification.

Implementation of the ILFN into hardware architecture is another area for additional research. In order to apply for real-time, on-board and transportable machine health monitoring, the ILFN classifier should be designed into suitable electronic hardware architecture.

REFERENCES

[1] Alguindigue, I. E., Buczak, A. L., and Uhrig, R. E. "Monitoring and Diagnosis of Rolling Element Bearings Using Artificial Neural Networks," *IEEE Transactions on Industrial Electronics*, Vol. 40, No. 2, pp.209-217, April 1993.

[2] Alampalli, S., Fu, G., and Dillon, E. W. "Signal Versus Noise in Damage Detection by Experimental Modal-Analysis," *Journal Of Structural Engineering-ASCE*, Vol. 123, No. 2, pp. 237-245, February 1997.

[3] Andrews, H. C. *Mathematical Techniques in Pattern Recognition*, New York: John Wiley & Sons, 1972.

[4] Ashton, E. A. "Detection of Subpixel Anomalies in Multispectral Infrared Imagery Using an Adaptive Bayesian Classifier," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 36, No. 2, pp. 506-517, March 1998.

[5] Asim Roy, S. G. and Miranda, R. "An Algorithm to Generate Radial Basis Function (RBF)-Like Nets for Classification Problems. *Neural Networks*, Vol. 8, No. 2, pp. 179-201, 1995.

[6] Ball, P. G. "Machine Wear Analysis a Rational Approach to Methods Integration for Maximum Benefits," *Lubrication Engineering*, Vol. 54, No. 3, pp. 18-22, March 1998.

[7] Bahr, B., Motavalli, S., and Arfi T. "Sensor Fusion for Monitoring Machine-Tool Conditions," *International Journal of Computer Integrated Manufacturing*, Vol. 10, No. 5, pp. 314-323, September-October 1997.

[8] Battiti, R. "First- and Second-Order Methodes for Learning: Between Steepest Descent and Newton's Method," *Neural Computation*, Vol. 4, No. 2, pp. 141-166, 1992.

[9] Bezdek, J. C. *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.

[10] Bezdek, J. C., Chuah, S. K., and Leep, D. "Generalized k-Nearest Neighbor Rules," *Fuzzy Sets Systems*, Vol. 18, No. 3, pp. 237-256, April 1986.

[11] Broch, J. T. *Mechnanical Vibrations and Shock Measurements*, Bruel & Kjaer, 1984.

[12] Broomhead, D. S. and Lowe, D. "Multivariable Function Interpolation and Adaptive Networks," *Complex Systems*, Vol. 2, pp. 321-355, 1988.

[13] Bullock, T. H., Achimowicz, J. Z., Duckrow, R. B., Spencer, S. S., and Iraguimadoz, V. J. "Bicoherence of Intracranial EEG in Sleep, Wakefulness and Seizures," *Electroencephalography and Clinical Neurophysiology*, Vol. 103, No. 6, pp. 661-678, December 1997.

[14] Cabell, R. H., Fuller, C. R., and Obrien, W. F. "Neural-Network Modeling of Oscillatory Loads and Fatigue Damage Estimation of Helicopter Components," *Journal of Sound and Vibration*, Vol. 209, No. 2, pp. 329-342, January 1998.

[15] Cai, L. Y. L. and Kwan, H. K. "Fuzzy Classifications Using Fuzzy Inference Networks," *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, Vol. 28, No. 3, pp. 334-347, June 1998.

[16] Cameron, B. G. "Final Report on CH-46 Aft Transmission Seeded Fault Testing," *Westland Research Paper RP907, Westland Helicopters*, 1 September 1993. Now accessible via WWW at http://wisdom.arl.psu.edu/Westland/

[17] Carpenter, G. A. and Grossberg, S. "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Stephen Recognition Machine", Computer Vision, Graphics and Image Processing, Vol. 37, pp.54-115, 1987.

[18] Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H., and Rosen, D. B. "Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps." *IEEE Transactions on Neural Networks*, Vol. 3, No. 5, pp. 698-713, September 1992.

[19] Carpenter, G. A., Grossberg, S., and Rosen, D. B. "Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System," *Neural Networks*, Vol. 4, pp. 759-771, 1991.

[20] Carpenter, G. A. and Grossberg, S. "A self-Organizing Neural Network For Supervised Learning, Recognition, and Prediction," *IEEE Communications Magazine*, pp. 38-49, September 1992.

[21] Carpenter, G. A., Grossberg, S., and Reynolds, J. H. "A Fuzzy ARTMAP Nonparametric Probability Estimator for Nonstationary Pattern Recognition Problems," *IEEE Transactions on Neural Networks*, Vol. 6, No. 6, pp. 1330-1336, November 1995.

[22] Chang, R. L. P. and Pavlidis, T. "Fuzzy Decision Tree Algorithms," *IEEE Transactions Systems, Man, and Cybernetics*, Vol. 7, No. 1, pp. 28-35, January 1977.

[23] Charalambous, C. "Conjugate Gradient Algorithm for Efficient Training of Artificial Neural Networks," *IEE Proceedings*, Vol. 139, No. 3, pp. 301-310, 1992.

[24] Chen, J. C. and Black, J. T. "A Fuzzy-Nets In-Process (FNIP) System for Tool-Breakage Monitoring in End-Milling Operations," *International Journal of Machine Tools & Manufacture*, Vol.37, No. 6, pp. 783-800, June 1997.

[25] Chen, K. H. and Tsai, W. H. "Vision-Base Autonomous Land Vehicle Guidance in Outdoor Road Environments using Combined Line and Road Following Technique," *Journal of Robotic Systems*, Vol. 14, No. 10, pp. 711-728, October 1997.

[26] Chen, S., Cowan, C. F. N., and Grant, P. M. "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks," *IEEE Transactions on Neural Networks*, Vol. 2, No. 2, pp. 302-309, March 1991.

[27] Cheng, H. D., Wu, C. Y., and Hung, D.L. "VLSI for Moment Computation and its Application to Breast-Cancer Detection," *Pattern Recognition*, Vol. 31, No. 9, pp. 1391-1406, September 1998.

[28] Coker, S. A., Oh, S. J., and Shin, Y. C. "In-Process Monitoring of Surface-Roughness Utilizing Ultrasound," *Journal of Manufacturing Science and Engineering-Transactions of the ASME* , Vol. 120, No. 1, pp. 197-200, February 1998.

[29] Deterding, D. H. "Speaker Normalisation for Automatic Speech Recognition," University of Cambridge, Ph.D. Thesis, 1989.

[30] Devijver, P. and Kittler, J. *Pattern Recognition: a Statistical Approach*, Englewood Cliffs, N.J.: Prentice-Hall, 1982.

[31] Dimartino, M., Fanelli, S., and Protasi, M. "Exploring and Comparing the Best Direct-Methods for the Efficient Training of MLP-Networks," *IEEE Transactions on Neural Networks*, Vol. 7, No. 6, pp. 1497-1502, November 1996.

[32] Dorrell, D. G., Thomson, W. T., and Roach S. "Analysis of Airgap Flux, Current, and Vibration Signals as a Function of the Combination of Static and Dynamic Airgap Eccentricity In 3-Phase Induction-Motors," *IEEE Transactions on Industry Applications*, Vol. 33, No. 1, pp. 24-34, January-February 1997.

[33] Dubois, D., Prade, H., and Testermale, C. "Weighted Fuzzy Pattern Matching," *Fuzzy Sets Systems*, Vol. 28, No. 3, pp. 313-331, December 1988.

[34] Duda, R. O. and Hart, P. E. *Pattern Classification and Scene Analysis*, New York: John Wieley and Sons, 1973.

[35] Echanobe, J., Demendivil, J. R. G., Garitagoitia, J. R., and Alastruey, C. F. "Deformed Systems for Contextual Postprocessing," *Fuzzy Sets and Systems*, Vol. 96, No. 3, pp. 335-341, June 1998.

[36] Fahlman, S. E. "Faster-Learning Variations on Back-Propagation: An Empirical Study," *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann, 1988.

[37] Fahlman, S. E. and Lebiere, C. "The Cascade-Correlation Learning Architecture," *Touretzky (ed.) Advances in Neural Information Processing Systems 2, Morgan-Kaufmann*, 1990. A slightly more detailed version of the paper is available as CMU tech report CMU-CS-90-100.

[38] Fisher, R. A. "The Use of Multiple Measurements in Axonomic Problems," *Annals of Eugenics,* Vol. 7, pp. 179-188, 1936.

[39] Forouraghi, B. "Inductive Learning of Materials Performance Indexes from a Material Properties Database," *Materials Evaluation*, Vol. 55, No. 9, pp. 1007-1012, September 1997.

[40] Fu, P., Hope, A. D., and Javed, M. A. "Fuzzy Classification of Milling Tool Wear," *Insight*, Vol. 39, No. 8, pp. 553-557, August 1997.

[41] Fukunaga, K. *Introduction to Statistical Pattern Recognition*, San Diego, CA: Academic Press, Inc., 2nd Edition, 1990.

[42] Fukunaga, K. and Koontz W. L. G. "A Criterion and an Algorithm for Grouping Data," *Transaction of the IEEE in Computers*, C-19, pp. 917-923, 1970.

[43] Gauthier, J. Krause, T. W., and Atherton, D. L. "Measurement of Residual-Stress in Steel using the Magnetic Barkhausen Noise Technique," *NDT&E International*, Vol. 31, No. 1, pp. 23-31, February 1998.

[44] Goldberg, L. O. "Eddy-Current Testing, an Emerging NDT Method for Ferritic Weld Inspection," *Materials Evaluation*, Vol. 56, No. 2, pp. 149ff, February 1998.

[45] Govindaraju, M. R., Kaminski D. A., Devine M. K., Biner S. B., and Jiles D. C. "Nondestructive Evaluation of Creep Damage in Power-Plant Steam-Generators and Piping by Magnetic Measurements," *NDT & E International*, Vol. 30, No. 1, pp. 11-17, February 1997.

[46] Grabisch, M. and Sugeno, M. "Multi-Attribute Classification Using Fuzzy Integral," *Proceedings FUZZ-IEEE*, San Diego, CA, pp. 47-54, March 1992.

[47] Grossberg, S. "Adaptive Pattern Classification and Universal Recording. II: Feedback, Expectation. Olfaction, and Illusions," *Biological Cybernetics*, Vol. 23 pp. 187-202, 1976.

[48] Hagan, M. T. and Menhaj, M. "Training Feedforward Networks with the Marquardt Algorithm," *IEEE Transactions on Neural Networks*, Vol. 5, No. 6, 1994.

[49] Hall, D. J. and Ball, G. B. "ISODATA: A Novel Method of Data Analysis and Pattern Classification," *Technical Report, Standard Research Institute, Menlo Park, CA*, 1965.

[50] Hall D. L., Hansen, R. J., and Lang, D. C. "The Negative Information Problem in Mechanical Diagnostics," *Journal of Engineering for Gas Turbines and Power-Transactions of the ASME*, Vol. 119, No. 2, pp. 370-377, April 1997.

[51] Halgamuge S. K. "Self-Evolving Neural Networks for Rule-Based Data Processing," *IEEE Transactions on Signal Processing*, Vol. 45, No. 11, pp. 2766-2773, November 1997.

[52] Haykin, S. *Neural Networks: A Comprehensive Foundation*, New York: MacMillan, 1994.

[53] Hinton, G. E. and McClelland, J. L. "Learning Representations by Recirculation," *Proceeding of the IEEE Conference on Neural Information Processing Systems*, November 1988.

[54] Heng, R. B. W. and Nor, M. J. M. "Statistical-Analysis of Sound and Vibration Signals for Monitoring Rolling Element Bearing Condition," *Applied Acoustics*, Vol. 53, No. 1-3, pp. 211-226, January-March 1998.

[55] Honlet, M. "Nondestructive Testing of Complex Materials and Structures using Optical Techniques," *Insight*, Vol. 40, No. 3, pp. 176-182, March 1998.

[56] Huh, K. and Kwak, B. "Evaluation Of Discrete-Time Well-Conditioned State Observers," *KSME International Journal*, Vol. 11, No. 5, pp. 505-512, October 1997.

[57] Hurdle, J. F. "The Synthesis of Compact Fuzzy Neural Circuits," *IEEE Transactions on Fuzzy Systems*, Vol. 5, No. 1, pp. 44-55, February 1997.

[58] Hush, D. R. and Horne, B. G. "Progress in Supervised Neural Networks," *IEEE Signal Processing Magazine*, pp. 8-39, January, 1993.

[59] Hwang, Y.-S. and Bang, S.-Y. "A Neural Network Model APC-III and Its Application to Unconstrained Handwritten Digit Recognition," *Proceeding of International Conference on Neural Information Processing, Seoul: Korean Association for Intelligent Information Systems*, pp. 1500-1505, 1994.

[60] Hwang, Y.-S. and Bang, S.-Y. "An Efficient Method to Construct a Radial Basis Function *Neural Network Classifier*," *Neural Networks*, Vol. 10, No. 8, pp. 1495-1503, 1997.

[61] Ishibuchi, H., Nozaki, K., and Tanaka, H. "Distributed Representation of Fuzzy Rules and Its Application to Pattern Classification," *Fuzzy Sets Systems*, Vol. 52, pp. 21-32, November 1992.

[62] Jang, J. S. "ANFIS: Adaptive-Network-Based Fuzzy Inference Systems," *IEEE Transactions on Systems, Man, and Cybernatics*, Vol. 23, No. 3, 1993.

[63] Jang, J. S. R. and Sun, C. T. "Functional Equivalence Between Radial Basis Function Networks and Fuzzy Inference Systems," *IEEE Transactions on Neural Networks*, Vol. 4, No. 1, pp. 156-158, January 1993.

[64] Juang, C. F. and Lin, C. T. "An On-Line Self-Constructing Neural Fuzzy Inference Network and Its Applications," *IEEE Transactions on Fuzzy Systems*, Vol. 6, No. 1, pp. 12-32, February 1998.

[65] Kaminski, W. and Strumillo, P. "Kernel Orthonormalization in Radial Basis Function Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 8, No. 5, pp. 1177-1183, September 1997.

[66] Kang, S. J. and Chien, S. I. "Domain-Specific Design of a Nondestructive Health Monitoring Expert-System," *Expert Systems with Applications*, Vol. 14, No. 3, pp. 385-397, April 1998.

[67] Karayiannis, N. B. and Mi, G. W. "Growing Radial Basis Neural Networks: Merging Supervised and Unsupervised Learning with Network Growth Techniques," *IEEE Transactions on Neural Networks*, Vol. 8, No. 6, pp. 1492-1506, November 1997.

[68] Karayiannis, N. B. and Pai, P.-I. "A Fuzzy Algorithm for Learning Vector Quantization," *Proceeding of IEEE International Conference on Systems, Man, and Cybernetics*, San Antonio, Texas, pp. 126-131, October 1994.

[69] Keller, J. M., Gray, M. R., and Givens, J. A. "A Fuzzy k-Nearest Neighbor Algorithm," *IEEE Transactions Systems, Man, and Cybernetics*, Vol. SMC-15, pp. 580-585, July/August 1985.

[70] Kiddy, J. and Pines, D. "Eigenstructure Assignment Technique for Damage Detection in Rotating Structures," *AIAA Journal*, Vol. 36, No. 9, pp. 1680-1685, September 1998.

[71] Kohonen T. *Self-Organizing Maps*, Berlin: Springer-Brt;sh, 2nd Edition 1997.

[72] Krzyzak, A. and Linder, T. "Radial Basis Function Networks and Complexity Regularization in Function Learning," *IEEE Transactions on Neural Networks*, Vol. 9, No. 2, pp. 247-256, March 1998.

[73] Kumar, S. A., Ravindra, H. V., and Srinvasa, Y. G. "In-Process Tool Wear Monitoring Through Time-Series Modeling and Pattern-Recongition," *International Journal of Production Research*, Vol. 35, No. 3, pp. 739-751, March 1997.

[74] Kuo, Y.-H., Kao, C.-I., and Chen, J.-J. "A Fuzzy Neural Network Model and Its Hardware Implementation," *IEEE Transactions on Fuzzy Systems*, Vol. 1, No. 3, pp. 171-183, August 1993.

[75] Langari, R., Wang, L., and Yen, J. "Radial Basis Functin Networks, Regression Weights, and the Expectation-Maximization Algorithm," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, Vol. 27, No. 5, pp. 613-623, September 1997.

[76] Lannes, W. And Schneider, H. "Pollution Severity Performance Chart Key y\to Just-In-Time Insulator Maintenance," *IEEE Transactions on Power Delivery*, Vol. 12, No. 4, October 1997. pp. 1493-1500.

[77] Li, S. and Elbestawi, M. A. "Tool Condition Monitoring in Machining by Fuzzy Neural Networks," *Journal of Dynamic Systems, Measuremen,t and Control-Transactions of the ASME*, Vol. 118, No. 4, pp. 665-672, December 1996.

[78] Lin, C. C. D. and Benwang, H. P. "Performance Analysis of Rotating Machinery using Enhanced Cerebellar Model Articulation Controller (E-CMAC) Neural Networks," *Computers & Industrial Engineering*, Vol. 30, No. 2, pp. 227-242, April 1996.

[79] Linkens, D. A. and Nie, J. "Learning Control Using Fuzzified Self-Organizing Radial Basis Function Neural Network," *IEEE Transactions on Fuzzy Systems*, Vol. 1, No. 4, pp.280-287, 1993.

[80] Lippmann, R. P. "Pattern Classification using Neural Networks," *IEEE Communications Magazine*, pp. 47-64, November 1989.

[81] Looney, C. G. *Pattern Recognition using Neural Network: Theory and Algorithms for Engineers and Scientists*, Oxford, New York: Oxford University Press, 1996.

[82] Lu, Y. and Yamaoka, F. "Fuzzy Integration of Classification Results," *Pattern Recognition*, Vol. 30, No. 11, pp. 1877-1891, November 1997.

[83] Lucas, M. and Thomson, W. T. "A Study of the Natural Vibratory Response of Stator Structures to Improve Condition Monitoring Strategies for Induction

Motors," *Proceedings of the Institution of Mechanical Engineers Part C-Journal of Mechanical Engineering Science*, Vol. 212, No. 1, pp. 57-68, 1998.

[84] Luo, Z.-Q. "On the Convergence of the LMS Algorithm with Adaptive Learning Rate for Linear Feedforward Networks," *Neural Computation*, Vol. 2, No. 3, pp. 226-245, 1991.

[85] Mandayam, S., Udpa, L., Udpa, S. S., and Lord, W. "Wavelet-Based Permeability Compensation Technique for Characterizing Magnetic-Flux Leakage Images," *NDT&E International*, Vol. 30, No. 5, pp. 297-303, October 1997.

[86] McCormick, A. C. and Nandi A. K. "Classification of the Rotating Machine Condition using Artificial Neural Networks," *Proceedings of the Institution of Mechanical Engineers Part C-Journal of Mechanical Engineering Science*, Vol. 211, No. 6, pp. 439-450, 1997.

[87] McCormick, A. C., Nandi, A. K., and Jack, L. B. "Application of Periodic Time-Varying Autoregressive Models to the Detection of Bearing Faults," *Proceedings of the Institution of Mechanical Engineers Part C-Journal of Mechanical Engineering Science*, Vol. 212, No. 6, pp. 417-428, 1998.

[88] McLoone, S. and Irwin, G. "Nonlinear Optimization of RBF Networks," *International Journal of Systems Science*, Vol. 29, No. 2, pp. 179-189, 1998.

[89] Mitra, S., De, R. K., and Pal, S. K. "Knowledge-Based Fuzzy MLP for Classification and Rule Generation," *IEEE Transactions on Neural Networks*, Vol. 8, No. 6, pp. 1338-1350, November 1997.

[90] Monica Bianchini, P. F., and Gori, M. "Learning without Local Minima in Radial Basis Function Networks," *IEEE Transactions on Neural Networks*, Vol. 6, No. 3, pp. 749-756, 1995.

[91] Moody, J. and Darken, C. J. "Fast Learning in Networks of Locally-Tuned Processing Units," *Neural Computation*, Vol. 1, No. 2, pp. 281-294, 1989.

[92] Morassi, A. and Rovere, N. "Localizing a Notch in a Steel Frame from Frequency Measurements," *Journal of Engineering Mechanics-ASCE*, Vol. 123, No. 5, pp. 422-432, May 1997.

[93] Musavi, M. Ahmed, W., Chan, K., Faris, K., and Hummels, D. "On the Training of Radial Basis Function Classifiers," *Neural Networks*, Vol. 5, pp. 595-603, 1992.

[94] Nadler, M. and Smith E. P. *Pattern Recognition Engineering*, New York: John Wiley & Sons Inc., 1992.

[95] Narendra, K. G., Sood, V. K., Khorasani, K., and Patel, R. "Application of a Radial Basis Function (RBF) Neural-Network for Fault-Diagnosis in a HVDC System," *IEEE Transactions on Power Systems,* Vol. 13, No. 1, pp.177-183, February 1998.

[96] Olmez, T. "Classification of ECG Wave-Forms by using RCE Neural-Network and Genetic Algorithms," *Electronics Letters*, Vol. 33, No. 18, pp. 1561-1562, August 1997.

[97] Owsley, L. M. D., Atlas, L. E., and Bernard, G. D. "Self-Organizing Feature Maps and Hidden Markov-Models for Machine-Tool Monitoring," *IEEE Transactions on Signal Processing*, Vol. 45, No. 11, pp. 2787-2798, November 1997.

[98] Pal, S. K. and Mandal, D. P. "Linguistic Recognition System Based on Approximate Reasoning," *Informat. Sci.*, Vol. 61, pp. 135-161, April 1992.

[99] Pappa, R. S., James, G. H., and Zimmerman, D. C. "Autonomous Modal Identification of the Space-Shuttle Tail Rudder," *Journal of Spacecraft and Rockets*, Vol. 35, No. 2, pp. 163-169, March-April 1998.

[100] Park, J. and Sandberg, I. W. "Approximation and Radial-Basis-Function Networks," *Neural Computation*, Vol. 5, pp. 305-316, 1993.

[101] Paya, B. A., Esat, I. I, and Badi, M. N. M. "Artificial Neural-Network-Based Fault Diagnostics of Rotating Machinery using Wavelet Transforms as a Preprocessor," *Mechanical Systems and Signal Processing*, Vol. 11, No. 5, pp. 751-765, September 1997.

[102] Rahman, A. F. R. and Fairhurst, M. C. "A New Hybrid Approach in Combining Multiple Experts to Recognize Handwritten Numerals," *Pattern Recognition Letters*, Vol. 18, No. 8, pp. 781-790, August 1997.

[103] Robinson, A. J. "Dynamic Error Propagation Networks," Cambridge University Engineering Department, 1989.

[104] Rumelhart, D. E. and McClelland, J. L. *Parallel Distributed Processing*, MIT Press Cambridge, MA, 1986.

[105] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. "Learning Representations by Back-Propagating Errors," *Nature*, 323, pp. 533-536, 1986.

[106] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. "Learning Internal Representations by Error Propagating," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland (Eds.), MIT Press, Cambridge, Amsterdam, 1986.

[107] Russo, M. "FUGENSYS: a Fuzzy Genetic Neural System for Fuzzy Modeling," *IEEE Transactions on Fuzzy Systems*, Vol. 6, No. 3, pp. 373-388, August 1998.

[108] Sarkar, M., Yegnanarayana, B., and Khemani, D. "Backpropagation Learning Algorithms for Classification with Fuzzy Mean-Square Error," *Pattern Recognition Letters*, Vol. 19, No. 1, pp. 43-51, January 1998.

[109] Saxena, N., Khatri, P. K., Baheti, G. L., Meghwal, L. R., Songara, K. C., and Meena, V. L. "Adapting A Fluoroscopy System For Tomographic Visualization," *Insight*, Vol. 40, No. 8, pp. 570-572, August 1998.

[110] Sekhar, A. S. and Prabhu, B. S. "Condition Monitoring of Cracked Rotors through Transient-Response," *Mechanism and Machine Theory*, Vol. 33, No. 8, pp. 1167-1175, November 1998.

[111] Shen, L. C., Chang, S. K., and Hsu, P. L. "Robust Fault-Detection and Isolation with Unstructured Uncertainty Using Eigenstructure Assignment," *Journal Of Guidance Control And Dynamics*, Vol. 21, No. 1, pp. 50-57, Jan-Feb 1998.

[112] Simpson, P. K. "Fuzzy Min-Max Neural Networks-Part 1: Classification," *IEEE Transactions on Neural Networks*, Vol. 3, No. 5, pp. 776-786, September 1992.

[113] Simpson, P. K. "Fuzzy Min-Max Neural Networks-Part 2: Clustering," *IEEE Transactions on Fuzzy Systems*, Vol. 1, No. 1, February 1993.

[114] Smith R. A., Zeqiri B., Jones L. D., and Hodnett M. "Nonlinear Propagation in Water and its Effect on Ultrasonic C-Scanning," *Insight*, Vol. 40, No. 1, pp. 12-19, January 1998.

[115] Specht, D. F. "Probabilistic Neural Networks for Classification, Mapping, or Associative Memory," *Proceeding of the IEEE International Conference on Neural Networks*, San Diego, CA, Vol. 1, pp. 525-532, July 1988.

[116] Specht, D. F. "Probabilistic Neural Networks and the Polynomial Adaline as Complementary Techniques for Classification," *IEEE Transactions on Neural Networks*, Vol. 1, No. 1, pp. 111-121, March 1990.

[117] Specht, D. F. "Enhacements to Probabilistic Neural Networks," *Proceeding of the IEEE International Joint Conference on Neural Networks (IJCNN)*, Baltimore, MD, Vol. 1, pp. 761-768. June 1992.

[118] Specht, D. F. and Romsdahl, H. "Experience with Adaptive Probabilistic Neural Networks and Adaptive General Regression Neural Networks," *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, Orlando, FL, Vol. 2, pp. 1203-1208, June-July 1994.

[119] Spoerre, J. K. "Application of the Cascade Correlation Algorithm (CCA) to Bearing Fault Classification Problems," *Computers in Industry*, Vol. 32, No. 3, pp. 295-304, March 1997.

[120] Staszewski, W. J. and Worden, K. "Classification of Faults in Gearboxes Preprocessing Algorithms and Neural Networks," *Neural Computing & Applications*, Vol. 5, No. 3, pp. 160-183, 1997.

[121] Stein, J. L. and Wang, C. H. "Estimation Of Gear Backlash Theory And Simulation," *Journal of Dynamic Systems, Measurement, and Control-Transactions of the ASME*, Vol. 120, No. 1, pp. 74-82, March 1998.

[122] Sun, C. T. "Rule-Base Structure Identification in an Adaptive-Network-Based Fuzzy Inference System," *IEEE Transactions System, Man, and Cybernatics*, Vol. SMC-15, pp. 116-132. Jan./Feb. 1985.

[123] Tontini, G. and De Queiroz, A. A. "RBF Fuzzy-ARTMAP: a New Fuzzy Neural Network for Robust On-Line Learning and Identification of Patterns," *Proceeding of the IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 2, pp. 1364-1369, October 1996.

[124] Tse, P., Wang, D. D., and Atherton, D. "Harmony Theory Yields Robust Machine Fault-Diagnostic Systems Based on Learning Vector Quantization Classifiers," *Engineering Applications of Artificial Intelligence*, Vol. 9, No. 5, pp. 487-498, October 1996.

[125] Tsoukalas, L H. and Uhrig, R. E. *Fuzzy and Neural Approaches in Engineering*, New York: John Wiley & Sons, 1997.

[126] Uebele, V. Abe, S., and Lan, M. S. "A Neural-Network-Based Fuzzy Classifier," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 25, No. 2, pp. 353-361, September 1996.

[127] Vuorimaa, P., Jukarainen, T., and Karpanoja, E. "A Neuro-Fuzzy System for Chemical Agent Detection," *IEEE Transactions on Fuzzy Systems*, Vol. 3, No. 4, pp. 415-424, November 1995.

[128] Wevers, M. " Listening to the Sound of Materials Acoustic-Emission for the Analysis of Material Behavior," *NDT&E International*, Vol. 30, No. 2, pp. 99-106, April 1997.

[129] White, M. *Carnegie Mellon University collection of neural net benchmarks*, accessible via WWW at http://www.boltz.cs.cmu.edu/, 1997.

[130] Williamson, J. R. "Gaussian ARTMAP: A neural Network for Fast Incremental Learning of Noisy Multidimensional Maps," *Neural Networks*, Vol. 9, No. 5, pp. 881-897, 1996.

[131] Yacamini, R., Smith, K. S., and Ran, L. "Monitoring Torsional Vibrations of Electromechanical Systems Using Stator Currents," *Journal of Vibration and Acoustics-Transactions of the*, Vol. 120, No. 1, pp. 72-79, January 1998.

[132] Yen, C. W. V. and Liu, T. Z. "A Competitive Learning-Process for Hyperspherical Classifiers," *Neurocomputing*, Vol. 17, No. 2, pp. 99-110, 1997.

[133] Yen, G. G. "Health Monitoring of Vibration Signatures in Rotorcraft Wings," *Neural Processing Letters*, pp. 127-137, 1996.

[134] Zadeh, L. "Fuzzy Sets," *Inform. Contr.*, Vol. 8, pp. 338-353, 1965.

[135] Zagyapan, M., Bensalem, A., Fairfeld, C. A., and Sibald, A. "Monitoring Vibro-Tamper Compaction by the Analysis of Ground-Borne Vibrations," *Insight*, Vol. 40, No. 7, pp. 490-495, July 1998.

APPENDICES

APPENDIX A

MATLAB Source Code of the ILFN Classifier

```
function [Wa,Wb,count,stand_A]=ilfn(p,t,wa,wb,c,stdv)
% [Wa,Wb,count,stand_A]=ilfn(p,t,wa,wb,c,stdv)
%
% ILFN (Incremental Learning Fuzzy Neural) is an
%   on-line one-pass incremental learning classifier with capability of both hard
%   and/or soft decision (assuming that data is normal distribution).
%
%    INPUTS ARGUMENTS:
%   p   = A vector or a matrix of training data (each column
%          represent a vector of each datum).
%   t   = A vector or a matrix of training target (each column
%          represent a vector of each target).
%   wa  = A eeight matrix in the hidden layer of the Input Subsystem.
%   wb  = A eeight matrix in the hidden layer of the Input Subsystem.
%   c   = Count of patterns that have been added in prototype in wa.
%   stdv = Standard deviation at each prototype in wa.
%
% WRITEN BY: PHAYUNG MEESAD  10/11/98  10:24am

sigma_0 = 0.001;          % Initial standard deviation
new_cate_cutoff = 0.3;    % The threshold (epsilon)
maxpop = 100;                % The maximum number allow to add in
                             %   a prototype in operation mode
J=1;
if nargin == 1,
  Ptr=p; Wa = Ptr(:,1)'; Wb = ones(1,1);
  count=1; start=2;train=0;
  stand_A = sigma_0*ones(size(Wa));
  fprintf('TRAINING . . . \n');
  tic
elseif nargin == 2,
  Ptr=p; Ttr=t; Wa = Ptr(:,1)'; Wb=Ttr(:,1)';
  count=1; start=2;train=1;
  stand_A = sigma_0*ones(size(Wa));
  fprintf('TRAINING . . . \n');
  tic
```

```
elseif nargin == 5,
  Ptr=p; Wa=t; Wb=wa; count=wb;
  stand_A=c; train=0;start=1;
  fprintf('PREDICTING . . . \n');
  fprintf(['Number of nodes = ',num2str(size(Wb,1)),'\n']);
elseif nargin == 6,
  Ptr=p; Ttr=t; Wa = wa; Wb = wb;
  count = c; stand_A = stdv; start=1; train=1;
  fprintf('TRAINING . . . \n');
  tic
else
  error('Invalid arguments.');
end

if train==1,
  for i=start:size(Ptr,2),
    A = Ptr(:,i)';
    B = Ttr(:,i)';
    Ta = dist(A,Wa');
    st=norm(stand_A(J,:));
    if st<0.001, st=0.001; end
    net = Ta./st;
    member = exp(-(net.^2)/2);
    [winner, J] = max(member);
    if (winner>=new_cate_cutoff)&(dist(Wb(J,:),B')==0),
        count(J) = count(J)+1;
        Wa(J,:) = (Wa(J,:)*(count(J)-1) + A)/count(J);
        stand_A(J,:) = sqrt( (1-(1/count(J)))*(stand_A(J,:).^2)...
          + (Wa(J,:)-A).^2);
    else %if (winner<new_cate_cutoff)
        Wa = [Wa; A];
        Wb = [Wb; B];
        count = [count; 1];
        stand_A = [stand_A; sigma_0*ones(size(A))];
    end
  end
else
  for i=start:size(Ptr,2),
    A = Ptr(:,i)';
    Ta = dist(A,Wa');
    st=norm(stand_A(J,:));
    if st<0.001, st=0.001; end
    net = Ta./st;
    member = exp(-(net.^2)/2);
    [winner, J] = max(member);
    if (winner>new_cate_cutoff)&(count(J)<maxpop),
  count(J) = count(J)+1;
  Wa(J,:) = (Wa(J,:)*(count(J)-1) + A)/count(J);
  stand_A(J,:) = sqrt( (1-(1/count(J)))*(stand_A(J,:).^2)...
     + (Wa(J,:)-A).^2);
elseif (winner<=new_cate_cutoff),
  Wa = [Wa; A];
  if size(Wb,2)>1,
    dummy=[];
    for loop1=1:size(Wb,1),
      bin=[];
      for loop2=1:size(Wb,2),
        bin=[bin,num2str(Wb(loop1,loop2))]
      end
      dec=bin2dec(bin);
      dummy=[dummy; dec];
    end
    maxd=max(dummy);
    newWb=maxd+1;
    newWb=dec2bin(newWb,size(Wb,2));
    new=[];
    for loop2=1:size(Wb,2),
      new=[new,str2num(newWb(loop2))];
    end
  else
    new=max(Wb)+1;
  end

  Wb = [Wb; new];
  count = [count; 1];
  stand_A = [stand_A; sigma_0*ones(size(A))];
  Ta = dist(A,Wa');
  st=norm(stand_A(J,:));
  if st<0.001, st=0.001; end
  net = Ta./st;
  member = exp(-(net.^2)/2);
  member = member/sum(member);
end

%pruning module
y=Wb(1,:);
mvy=[member(1)];
l=1; k=2;
while k<=size(Wb,1),
  if dist(y(l,:),Wb(k,:)')==0,
    maxMV=max(mvy(l),member(k));
    mvy(l)=maxMV;
```

```
      else
        y=[y; Wb(k,:)];
        maxMV=member(k);
        mvy=[mvy; maxMV];
        l=l+1;
      end
      k=k+1;
    end;
    % end pruning module

    Wb_=y;
    MV=mvy/sum(mvy);
    [winner_, J_] = max(MV);

    fprintf('\nHARD DECISION \n');
    fprintf(['Predict Class : ',num2str(Wb_(J_,:)),'\n']);
    fprintf('\nSOFT DECISION \n');
    for c=1:size(Wb_,1),
      fprintf(['Class ',num2str(Wb_(c,:))]);
      fprintf([', MV = ',num2str(mvy(c),2),'\n']);
    end
    pause(1)
  end
end
if train==1,
  fprintf(['Training time   = ',num2str(toc), ' seconds\n']);
end
return
```

APPENDIX B

MATLAB Source Code for Testing the ILFN Classifier

```
function checknet(P,T,Wa,Wb,stdv)
% checknet(p,t,wa,wb,std)
%
% Use to test the performance of the ILFN classifier
%     p = input patterns
%     t = target patterns
%     wa = weigth centers of Gaussian kernel
%     wb = weigth target
%     std = standard deviation of Gaussian kernel
% WRITTEN BY: PHAYUNG MEESAD OCTOBER 11, 1998 1:13PM

wrong_class=0;
percent_correct=0;
total=0;
J=1;
fprintf('\nTESTING: Please wait . . . \n');
for i=1:size(P,2),
  A=P(:,i)';
  B=T(:,i)';
  Ta = dist(A,Wa');
  st=norm(stdv(J,:));
  net = Ta./st;
  member = exp(-(net.^2)/2);
  [winner, J] = max(member);

  %pruning module
  y=Wb(1,:);
  mvy=[member(1)];
  l=1; k=2;
  while k<=size(Wb,1),
    if dist(y(l,:),Wb(k,:)')==0,
      maxMV=max(mvy(l),member(k));
      mvy(l)=maxMV;
    else
      y=[y; Wb(k,:)];
      maxMV=member(k);
      mvy=[mvy; maxMV];
      l=l+1;
    end
    k=k+1;
  end;
  % end pruning module
  Wb_=y;
  MV=mvy/sum(mvy);
  [winner_, J_] = max(MV);
  if dist(Wb_(J_,:),B')~=0, wrong_class=wrong_class+1; end
  total=total+1;
end
percent_correct=(total-wrong_class)*100/total;
fprintf(['\nPercent correction     = ',num2str(percent_correct)]);
fprintf(['\nTotal data for testing = ',num2str(total)])
fprintf(['\nNumber of wrong prediction = ',num2str(wrong_class),'\n']);
```

APPENDIX C

An Example of MATLAB Source Code For Preprocessing of Westland Vibration Data

```
format short g;
load w20099ff.mat
d200=data;
load w30058ff.mat
d300=data;
load w40088ff.mat
d400=data;
load w50091ff.mat
d500=data;
load w60046ff.mat
d600=data;
load w70095ff.mat
d700=data;
load w80087ff.mat
d800=data;
load w90001ff.mat
d900=data;
data={d200 d300 d400 d500 d600 d700 d800 d900};
T=[0 0 1 0; 0 0 1 1;0 1 0 0;0 1 0 1;0 1 1 0;0 1 1 1;1 0 0 0;1 0 0 1]';
%T=[2 3 4 5 6 7 8 9];
fs=103116.08; MaxP = 9e8; w=hanning(1024); nfft=3072; novl=512;
P100=[]; T100=[]; index=1;
for i=1:novl:novl*100, % data 100 samples
  for j=1:size(T,2), % faults
    P_=[];  for k=1:8, % channels
      [p, f] = spectrum(data{j}(k,i:i+nfft-1),nfft,novl,w,fs);
      p = p(70:210,1)/MaxP;%/norm(p(70:210,1));
      P_=[P_; p];
    end  % channels
    P100=[P100 P_];
    T100=[T100 T(:,j)];
    index=index+1;
  end % faults
end
save patt100 P100 T100
```

2

VITA

Phayng Meesad

Candidate for the Degree of

Master of Science

Thesis: PATTERN CLASSIFICATION BY AN INCREMENTAL LEARNING FUZZY NEURAL NETWORK

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Nakornratchasima, Thailand, On October 19, 1968, the son of Phayom and Hem Meesad.

Education: Received a Diploma degree in Electrical Power Technology from the Institute of Technology and Vocational Education Nakornratchasima, Thailand in March 1989; Received a Bachelor of Science in Technical Education degree (Electrical Engineering) from King Mongkut's Institute of Technology North Bangkok, Bangkok, Thailand in May 1994. Completed the requirements for the Master of Science degree with a major in Electrical Engineering at Oklahoma State University in December, 1998.

Experience: Employed as a technician, 1990 to 1994, and a lecturer, 1994 to present by King Mongkut's Institute of Technology North Bangkok, Thailand; employed by Oklahoma State University, School of Electrical and Computer Engineering as a teaching assistant, Spring 1998.

Professional Memberships: The Institute of Electrical and Electronics Engineers.