Estimation of reliability measures for artificial neural networks

by

Sudnya Bhupendra Shroff

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Electrical Engineering

Major Professor: Dr. Lalita S. Udpa

Iowa State University

Ames, Iowa

1997

Graduate College
Iowa State University

This is to certify that the Master's thesis of

Sudnya Bhupendra Shroff

has met the thesis requirements of Iowa State University

To my parents
Bhupendra and Suchita
and my dearest brother
Shishir

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

First of all, my heartfelt gratitude to Dr. Lalita Udpa for giving me the opportunity to pursue graduate studies. Many thanks to Dr. S. S. Udpa for instilling tremendous motivation and inspiration in me through my very first class as a student at ISU; to Dr. W. Lord and Dr. S. Kothari for agreeing to be on my committee and patiently putting up with the change in schedule of my final exam.

Thanks to my room-mate, Mona, who gave a new meaning to the word friendship; Daya and Rishi for helping with my thesis corrections; and, friends Arun and Naveen, for making my stay in Ames so enjoyable.

Last but not the least, I would like to express my deepest affection and gratitude to my husband, Nickhil, without whom this thesis would not have been possible.

1. INTRODUCTION

1.1. Background of Non-destructive Testing

Non-destructive testing (NDT) techniques are means by which materials and structures may be inspected without impairment of their performance. NDT techniques play a vital role in a variety of industries for evaluating the integrity of critical components. Defects may occur in a material or component during manufacturing, where the location and size of the defect will influence subsequent performance of the component. Other defects, such as fatigue cracks or stress corrosion cracking, may be generated within a material or specimen during service. It is, therefore, necessary to have a reliable means for detecting the presence of defects at the manufacturing stage and monitoring the rate of growth of defects during the service life of a component or assembly. A typical NDT system (Figure 1.1) involves the use of a transducer for energizing the test specimen. The energy/specimen interaction is sampled by the receiving transducer. The received signal is then analyzed for evaluating the integrity of the test object. The diverse nature of NDT applications have led to the development of a variety of inspection techniques employing different forms of excitation energy. Electromagnetic, radiographic and ultrasonic techniques are some common examples.

Based on the frequency of the excitation source, electromagnetic NDT methods can span the entire electromagnetic spectrum. Magnetostatic flux leakage (MFL) and electrostatic potential drop NDT are zero frequency(dc) methods. Eddy current and remote field NDT are low frequency techniques and can be classified as quasi-static.

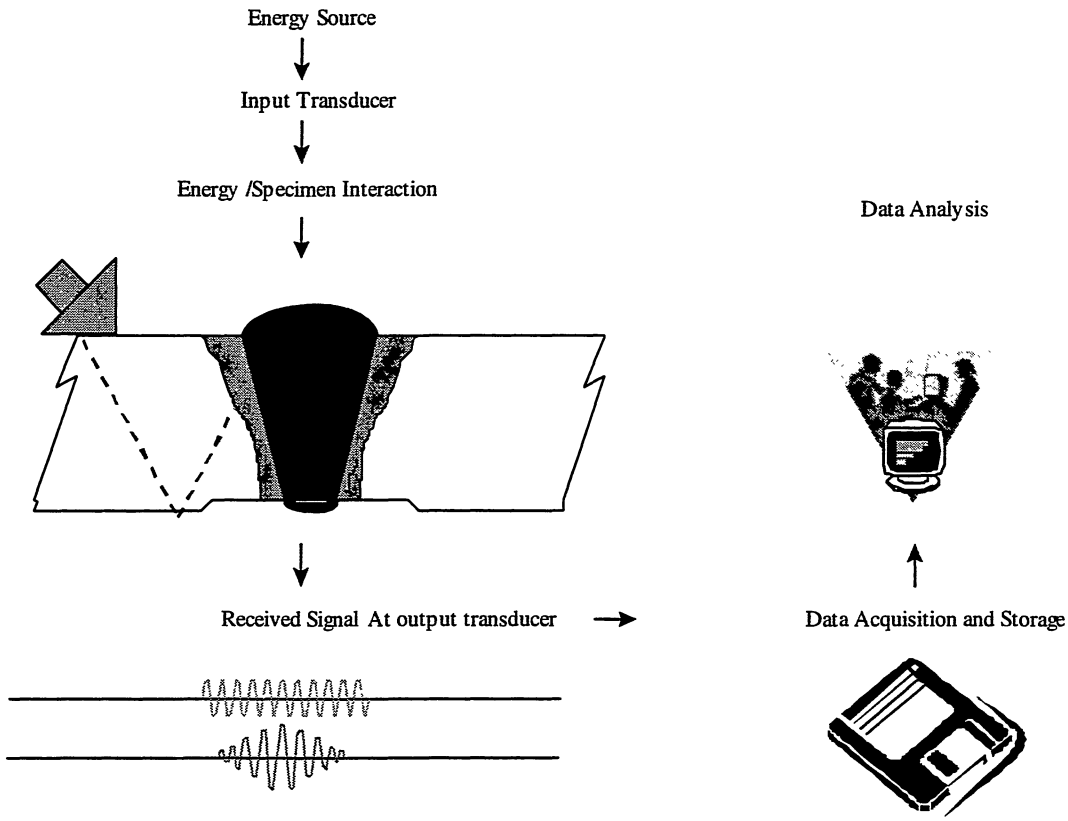

Figure 1.1 A generic NDT system

X-ray radiographic and microwave are wave-based methods. The underlying physics and the mathematical governing equations for each technique is different. Another commonly used NDT method relies on the propagation of ultrasonic waves in a material.

Ultrasonic techniques are capable of detecting internal defects in thick specimen. In this method, sound waves of short wavelength and high frequency are propagated through specimen and reflections or echoes from the backwall, discontinuities and anomalies are detected, and analyzed.

1.2. Inverse Problems in NDE

Characterization of test specimen on the basis of information in the response from energy/specimen interaction is referred to as the 'inverse problem' [4]. Inverse problems in NDT address the crucial issue of defect characterization wherein the information in signals is used to identify the shape, size and location of defects in materials. The strategy employed for solving these problems is a function of the underlying physical process. Based on the energy propagation process, the methods for solving inverse problems have to be appropriately tailored.

As the frequency increases from low to high, the amount of information contained in the signal increases. Static methods such as MFL generate signals containing information that can be extracted solely on the basis of amplitude measurements. Methods in which the interaction process between energy and specimen is diffusive, provide both amplitude and phase information. Finally, wave based methods provide amplitude, phase and range information to the signal analyst. The static methods, therefore, contain less information as compared to diffusion or wave based methods, making the inverse problem or signal characterization more difficult.

Thus, ultrasonic or microwave NDT data, being richer in content, lend themselves to procedures such as holography and tomography for obtaining solutions to inverse problems. Additionally, data intensive methods that are capable of utilizing vast amounts of information can be used.

Several methods have been proposed to solve inverse problems. A taxonomy of inverse problem solution methods [4] is described in Figure 1.2.

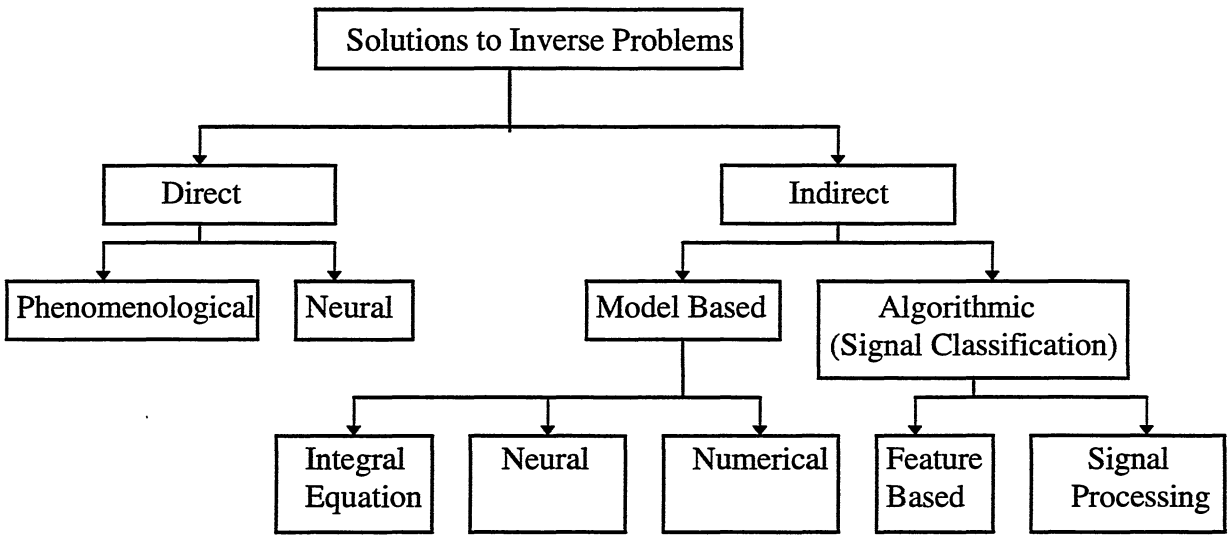

Figure 1.2 Taxonomy of inverse problem solution methods [4]

Of the different methods shown in Figure 1.2, direct analytical methods are more difficult particularly when the governing differential equations are nonlinear and crack shapes are arbitrary. These equations are, therefore, solved using numerical computational techniques. However, numerical models cannot be used directly for solving inverse problems. As a result, it is more common to see signal classification approaches which are being used increasingly in many commercial applications. In this class of techniques, the solutions to inverse problems are obtained as assignments of a measured signal to a known class of defects. These inversion techniques are typically independent of the underlying physics of the NDE process.

Matched filters [30] is an example of a signal classification method that has received considerable attention. Matched filters are linear filters whose transfer functions are matched to input signals. In their application to NDE, the technique of using matched filters requires a

data bank of input signals from the complete variety of defects that may be encountered. The technique consists of building an array of filters, each one of which is matched to a known signal. The unknown signal is then classified by observing the outputs of the filter array.

1.3. Feature Based Signal Classification

Feature based methods are more often used than matched filters because matched filters require storage of the entire signal whereas feature based methods reduce the input signal dimension by storing only the extracted features. Feature based methods involve two steps - feature extraction and classification. Feature extraction is defined as the reduction in dimensionality of the pattern vectors by means of a linear transformation that extracts certain invariant attributes of the pattern classes under consideration. There are different types of features - physical, structural and mathematical. Physical and structural features are extensively used by human beings for classification purposes as they are easily detected by eye or other sensory organs. Color, texture etc. are examples of physical features whereas shape is an example of a structural feature. Mathematical features are, however, more suitable for machine implementation and are more general in scope. Statistical means, correlation coefficients, eigenvalues and eigenvectors of covariance matrices, etc. are examples of mathematical features.

Once a feature vector is generated, it is input to a signal classifier for classification into a defect class. Numerous classification methods have been developed [29] and used for various applications. The simplest and most intuitive classification algorithm is the K-means clustering algorithm that maximizes the distance between clusters of different classes, while minimizing the variance of features within a given class. More recently, Artificial Neural

Networks (ANNs), another category of classifiers are being extensively used, largely due to their capability to model complex functions.

1.4. Reliability of Classification

A lot of work has been done in the areas of feature extraction and selection as well as pattern classification for NDE problems. However, attempts to quantify the confidence or the reliability of the classification decision by signal classifiers have been very few and far in between. This thesis addresses the issue of predicting the confidence associated with a classification decision. A neural network based inverse problem solution in ultrasonic NDE is considered where the inversion process is formulated as a pattern recognition problem and the signal is directly classified as belonging to one of a set of known defect classes. Various approaches for reflecting the certainty in a classification decision have been discussed and a new method has been proposed.

1.5. Organization and Scope

The organization of this thesis is as follows:

The details of ANNs and the need for confidence measure are described in the problem statement covered in Chapter 2. Chapter 3 gives a brief description of several methods for error estimation reported in literature along with an evaluation of their merits and demerits.

Chapter 4 covers *'fuzzy'* approaches taken to determine the confidence measure in a classification decision. The fuzzy K-means algorithm and the Multi-Layer Perceptron are also described in Chapter 4.

Chapter 5 outlines the details of the method used for determining the reliability of the Radial Basis Function network. A new method for resolving the shortcomings of the other existing methods is proposed and described in Chapter 6.

Results of implementations of the different methods are included following the description of each technique. Finally, the conclusions reached in course of this study along with a discussion of the potential future work are covered in Chapter 7.

2. PROBLEM STATEMENT

2.1. Background

2.1.1. Introduction

Ultrasonic NDT is most commonly used in the inspection of piping welds and pressure vessels in reactors. Pipes and vessels are designed to have an added amount of wall thickness or a specific corrosion allowance based on conditions such as pressure, acceptable corrosion rates and the expected life of the equipment. Extremely high corrosion rates could lead to wall thinning or stress related cracking (also called Intergranular Stress Corrosion Cracking - IGSCC) resulting in dangerous conditions with regard to pipe and pressure ratings. IGSCC occurs in the heat affected zones (HAZ) of stainless steel piping welds and propagates in a branch-like manner along the grain boundary. There are two phases of IGSCC, initiation and growth. IGSCC results mainly from following three factors:

- A tensile stress on the inner diameter of the weld region
- A sensitized grain structure
- A corrosive environment

All three conditions are required to produce IGSCC but removal of any one condition is sufficient to stop its progress. A tensile stress always exists on the inner diameter of the pipe weld region. The smaller the diameter and the thinner the pipe material, the greater the residual stress. Material sensitization occurs either during manufacturing due to improper heat treatment or milling, or during welding of austentic stainless steels at high temperatures (between 900-1600^{o} F). During large temperature changes, the carbon migrates to the grain

boundary and interacts with the chromium to create chromium carbides. A corrosive environment on the inner diameter of the weld region includes, among other highly corrosive mediums, substantial quantities of oxygen and boric acid. This corrosive medium attacks the low chrome areas formed due to sensitization at the grain boundary. IGSCCs tend to be extremely tight, and are often highly branched at the crack tip. IGSCC is thus, intimately related to the type of pipe material, environmental conditions, temperature and cyclic loads superimposed on a mean stress on the process pipe or vessel which may be in an aggressive environment.

Due to the combined effects of stress, environment and sensitization, if pipes and vessels become too thin, the cracks may propagate in them long enough resulting in considerable damage. For this reason, in the particular case of nuclear power plant tubing, the ability to distinguish the ultrasonic reflections from IGSCC is critical. To avoid any mishap, it is important to detect IGSCC as soon as it occurs. Since the cracking occurs on the insides, close to the weld region in the heat affected zones, inspection of welds contains reflections not only from IGSCC but also from other weld joint features, such as root welds and counterbores (ridges machined prior to welding to match unequal pipewall thickness). This makes it difficult to distinguish between the IGSCC and other benign signals.

Figure 2.1 shows the 3-D geometry of pipes in the weld region. Figure 2.2 is the cross section taken along the axis.

The inverse problem considered in this thesis is the analysis and classification of ultrasonic weld inspection signals into cracks, counterbores and rootwelds.

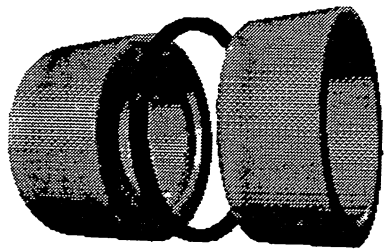

Figure 2.1 3-D geometry of the weld region of pipe

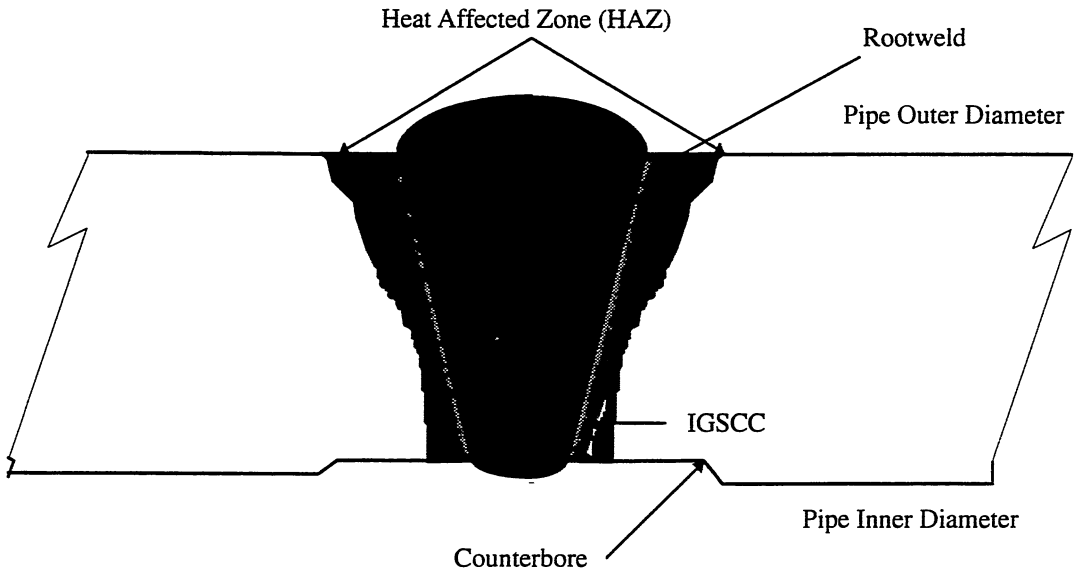

Figure 2.2 Cross-section of the pipe along the axis

The approach used is a feature based method for solving the classification problem which consists of the following steps:

1) *Feature extraction*, where characteristic features (physical or mathematical) in the signal that carry the discriminatory information are extracted.

2) *Classification*, which involves the use of a clustering algorithm or neural network to assign each feature vector to one of the known classes.

3) *Confidence Measure determination*, where the reliability of the classification decision is determined, quantitatively.

Each of these steps are described below.

2.2. Feature Extraction

A variety of algorithms have been developed over the years for identifying features in ultrasonic waveforms. Features have been derived from time, spatial and frequency domains such as wavelet coefficients [39], principle components [9] etc.

The research in this thesis focuses more on the concept of reliability of the classification performance and hence the optimality of features is not central to this work. Consequently, the features used for the classification are very simple, intuitive quantities. The simple two dimensional feature vector is easier to implement for reliability calculation. Also it is easier to understand and interpret the results obtained.

Typical signals reflected from a crack, a counterbore and a root weld are shown in Figure 2.3.

It is seen (from Figure 2.3) that there are considerable similarities between the different classes of signals making signal classification a challenging task. The differences in signals are used to derive discriminating features that reduce the dimensionality of the input. Three physical features are considered based on the differences observed in the signals. For instance, the number of significant peaks and the peak amplitude are seen to be different for each of the three classes.

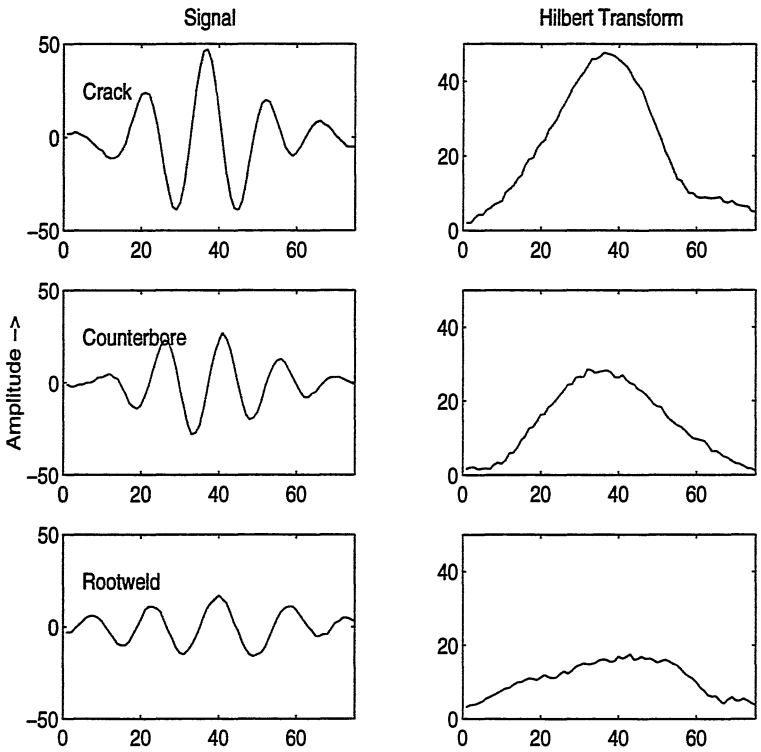

Figure 2.3 Typical Signals

The crack signals have the largest amplitude, followed by those of the counterbore, with the rootweld having the smallest amplitude. The third feature is the pulsewidth, computed using the Hilbert transform. The support of the Hilbert transform decreases from rootweld to counterbore to crack. Thus, the features selected are number of peaks, peak value of the amplitude and the pulse width and are calculated as follows:

Number of peaks: Let P represent the peak value of the signal. All peaks having an amplitude greater than or equal to $k*P$ are retained where $0 < k \leq 1$. The rest of the peaks are discarded. For example, if k = 0.25, all peaks with amplitudes upto 25% of the peak amplitude (P) are counted. Let $I(i)$ be defined as follows and x be an N point long A-scan:

$$I(i)=\begin{cases}1, & if \quad x(i)\geq k*P \\ 0, & otherwise\end{cases} \qquad i=1,2,...N \tag{2.1}$$

$$Number\_of\_peaks=\sum_{i=1}^{N} I(i) \tag{2.2}$$

Peak Value: The peak value (P) is the largest amplitude value of the signal under consideration, defined simply as

$$P=\max_{1\leq i\leq N}[x(i)] \qquad 1\leq i\leq N \tag{2.3}$$

Pulsewidth: To evaluate the pulsewidth, the Hilbert transform, $\hat{x}$, of the A-scan, x is first calculated to get

$$\hat{x}[n]=\sum_{m=-\infty}^{\infty} h[n-m]x[m] \qquad n=1,2,...N \tag{2.4}$$

where

$$h[n]=\begin{cases}\dfrac{2\sin^2(\pi n/2),}{\pi n} & n\neq 0, \\ 0, & n=0.\end{cases} \tag{2.5}$$

The magnitude of $\left(\hat{x}^2[n]+x^2[n]\right)^{1/2}$ is calculated and plotted as shown in Figure 2.3. The pulse width is then calculated as the difference between two points on the curve at 10% of the peak amplitude.

Thus the three features identified above reduce the dimension of the input signal from 1800 (length of the normal A-scan) to 3. These features then form components of a feature vector which describes the signal under consideration. The feature vectors of the signals are then classified by a selected classifier.

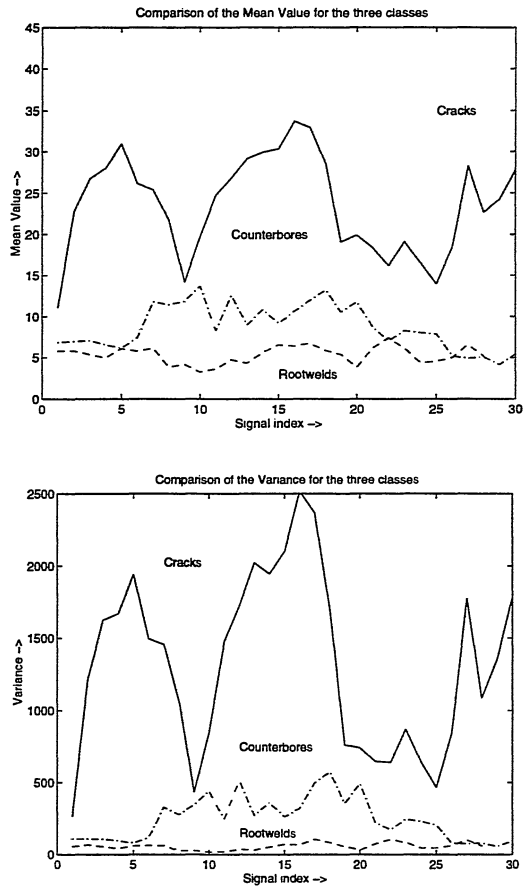

Figure 2.4 Mean and Covariance plots of the ultrasonic data

In addition to the above features selected by observing the raw signal, *mean* and *covariance* values for the different classes were plotted in Figure 2.4. They were estimated as:

$$\hat{m} = \frac{1}{N}\sum_{i=1}^{N} x(i) \tag{2.6}$$

$$\hat{\sigma} = (y^t * y) / (N-1), \tag{2.7}$$

where $\boldsymbol{y}$ is defined as

$$y(i) = x(i) - \frac{1}{N}\sum_{j=1}^{N} x(j), \tag{2.8}$$

where $N = 1800$.

These parameters (Figure 2.4) are also seen to contain discriminatory information. An equal number of cracks, counterbores and rootwelds were taken and their mean and covariance values were estimated using Equations (2.6) and (2.7) respectively.

The plots show that cracks have the largest mean and covariance values. Counterbores have lower and rootwelds have the lowest mean and covariance values. The plot of mean values shows that there exist some signals from both counterbore and rootweld classes that have values that are not completely discriminatory. Thus, the mean $\hat{m}$ by itself is not sufficient to help in classifying a signal.

2.3. Classification

Classification or pattern recognition is one of the basic attributes of human beings. All objects around us have a certain description which includes discriminating features that distinguish them from other objects. A human brain, being an extremely sophisticated information processing system, can perform pattern recognition of concrete items as well as abstract views. Recognition of concrete patterns may be considered as a psychophysiological problem [14], where a person receives a physical stimulus and makes an inference based on past experience. Thus, classification may be regarded as a problem of discriminating the input data with the help of features or invariant attributes among its members. In most cases, a human being far exceeds any automatic recognition system in its classification capabilities. However, there is still a necessity for automatic classifiers in areas of hazardous environment

or where recognition efficiency may be degraded by human factors such as fatigue. In other situations, the discriminating features of the input stimulus may be too cryptic. Consequently, there has been a considerable development of theory and techniques for the design of computer algorithms for simulating pattern recognition capabilities of a human brain. This has resulted in the emergence of Artificial Neural Network (ANN). ANNs are computer algorithms which attempt to simulate the performance of neurons in the human brain.

2.3.1. Modeling Capabilities of Artificial Neural Networks

ANNs are widely used in pattern classification applications largely due to their capability for modeling complex functions. There are many different kinds of ANNs. Rosenblatt's Perceptron Model [7], the Hopfield Network [7], Multi-Layer Perceptron [9], Radial Basis Function Network [9], etc. are some examples. Neural networks especially find extensive application in industry in modeling processes which are inherently difficult to understand. In general, physical systems are characterized with the help of mathematical models. Very accurate models can be built when the physics underlying the system being modeled is known. However, in many cases, the physics underlying the system is not well understood or too complex and intractable. In such cases, empirical methods are used to develop approximate mathematical models. Generalizers can be used to generate such empirical models automatically because they infer the parent function from available sets of data. Most generalizers perform very well if the systems they model are well behaved. For example, data which is linear and normally distributed can be modeled successfully using statistical methods. ANNs have shown to provide good approximating functions for nonlinear models with high computation speeds even with large dimensionality of problem.

This is largely due to their highly parallel structure. ANNs also have a powerful representational capacity. ANNs are used extensively in this capacity.

It has been shown that a Multi-Layer Perceptron (MLP) with a single hidden layer trained using backpropagation learning algorithm is sufficient to approximate any function, provided that enough nodes are present in the hidden layer [31]. The Radial Basis Function Network (RBFN) also has similar capabilities. An overview of the MLP and RBFN which are more commonly used in NDE applications is given in the following sections.

2.3.2. Multi-Layer Perceptron (MLP)

MLPs are a class of feedforward neural networks that typically consist of three types of layers, namely , the *input layer*, the *hidden layers* and the *output layer*. In this sense they are a generalization of the single layer perceptrons [9].

Notation used for MLP:

L	Number of hidden layers
S_h	Number of nodes in the h^{th} layer, $h=1,2,...L$
x_{hi}	Input to the i^{th} node of the h^{th} layer
y_{hj}	Output from j^{th} node of h^{th} layer
w_{ji}	Weight connecting the i^{th} node of $(h-1)^{th}$ layer to j^{th} node of h^{th} layer
θ_{hj}	Bias for the j^{th} node

Nodes in different layers are connected to each other via weights. The input to the i^{th} node of the h^{th} layer is the weighted sum of all the outputs from the $h\text{-}1^{th}$ layer. The model of each neuron in the network is associated with a continuously differentiable transfer function. The most commonly used form satisfying this condition is the *sigmoidal* transfer function.

This is mathematically described as follows: Let x_{hi} be the input to the i^{th} node of the h^{th} layer and y_{hi} be the corresponding outputs. Then,

$$y_{hj} = \frac{1}{1+e^{-\phi_{hj}}}, \tag{2.9}$$

where

$$\phi_{hj} = \sum_{i=1}^{S_{h-1}} w_{ji} x_{hi} + \theta_{hj} \qquad for \quad j = 1,2,...S_h \; and \quad h = 1,2,....L \tag{2.10}$$

The θ_{hj} term is the bias for the j^{th} node and S_h is the number of neurons in the h^{th} layer. A conventional MLP structure is shown in Figure 2.5.

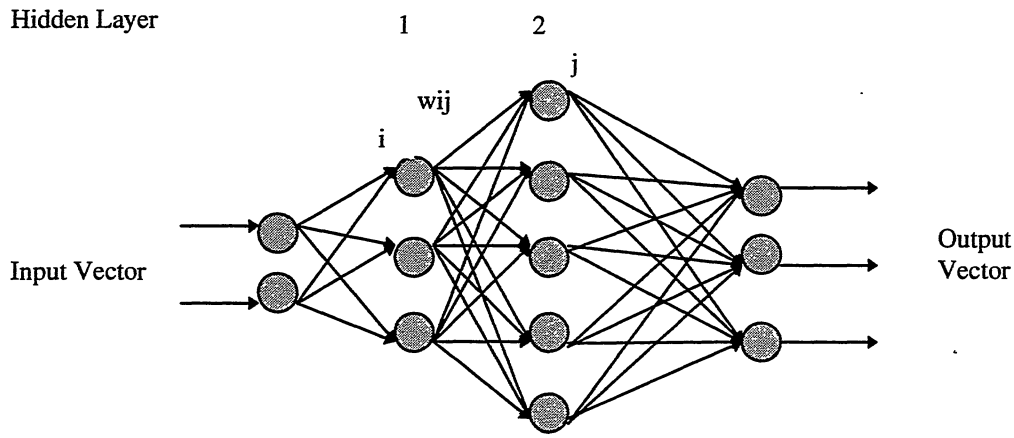

Figure 2.5 Architecture of an MLP

Typically, a neural network operates in two phases, namely training and testing. In the training phase of the MLP, the desired outputs are clamped to the output nodes for the corresponding inputs. The network 'learns' these input-output mapping by iteratively minimizing an error function. In this case, the error function, *E*, is the sum of the square of

the difference between the calculated (y_j) and the desired output. This is given by the following equation:

$$E = \sum_{j=1}^{N} (\hat{y}_j - y_j)^2 , \tag{2.11}$$

where N is the number of output nodes.

MLPs have been successfully applied to solve complex problems by training them in a supervised manner using the popular backpropagation algorithm. Since this algorithm is based on the *error correction rule*, it can also be considered as a generalization of the Least Means Square (LMS) [11] algorithm. The backpropagation performs a stochastic gradient descent in the weight space. Basically, the error backpropagation process consists of two passes through the different layers of the network. In the forward pass, an input vector is applied to the *input layer* and its effect is propagated forward to the *output layer* to provide the response of the network to the input stimulus. The weights of the connections in the network remain fixed. In the backward pass, error is propagated backwards from the output layer and the weights are adjusted using an error correction rule so as to make the actual response move closer to the desired response.

2.3.3. Radial Basis Function Network (RBFN)

Unlike Multi-Layer Perceptrons (MLPs), RBFNs use a distance metric in the input space to determine the hidden layer activations. As a result, the contours of constant activation of the hidden layer are hyperspheres instead of hyperplanes as with MLPs. The contours are finite in length and form closed regions of significant activation, as opposed to

MLPs where the contours are infinite in length and form semi-infinite regions of significant activation. This feature of RBFNs is exploited to produce a local reliability measure.

The RBFN consists of three layers of nodes as shown in Figure 2.6. Each successive layer is exhaustively interconnected by feedforward arcs.

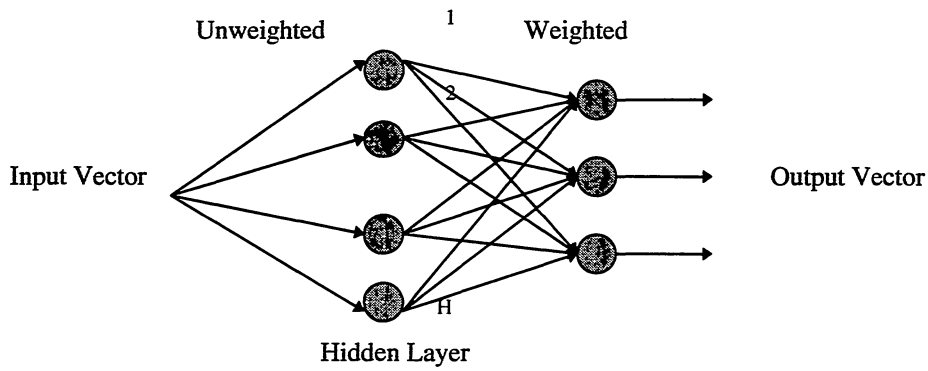

Figure 2.6 Architecture of an RBFN

Notations used for RBFN:

H	Number of hidden nodes
ϕ	Transfer function of the hidden node
$\boldsymbol{x_h}$	Center of the basis function at h^{th} hidden node
σ_h	Width of the basis function at h^{th} hidden node
a_h	Activation at the output of h^{th} node
y_i	Output from j^{th} node of output layer

1) The first layer is simply a fanout of the inputs to the hidden layer and are not weighted connections.

2) The hidden layer consists of H radial units plus one bias node with a constant activation of one. The transfer function of the hidden node is computed using a basis function ϕ,

$$a_h = \phi(- \| \mathbf{x} - \boldsymbol{x}_h \|^2 / \sigma_h^2) \tag{2.12}$$

where a_h is the output of the unit h in the hidden layer for a given input $\mathbf{x}$.

Each RBF node is characterized by two internal parameters, namely $\boldsymbol{x}_h$ and σ_h: $\boldsymbol{x}_h$ is the position of the basis center in the N-dimensional feature space and σ_h is a distance scaling parameter which is the width in the input space over which the unit will have a significant influence. The connections in the second layer of the RBFN represent weights of the linear combination.

3) The output layer has nodes which are linear summation units. The value of the i^{th} output node y_i is given by

$$y_i = \sum_{h=1}^{H+1} w_{ih} a_h = \sum_{h=1}^{H+1} w_{ih} \phi\left(\left\| x - x_h \right\|^2 / \sigma_h^2 \right) \tag{2.13}$$

where w_{ih} are the interconnection weights from the hidden nodes to the i^{th} output node. The $(H+1)^{th}$ node is the bias node with $a_{H+1}=1$.

2.3.3.1. Training the RBFN

There are several variations in the techniques for training the RBFN. The most commonly used technique is based on the algorithm suggested in [45]. This method trains the RBFN in three sequential stages:

1) The first stage consists of determining the number of unit centers H and position of the unit centers x_h by the K-means clustering algorithm, an unsupervised technique that places unit centers centrally among clusters of training points.

2) Next the unit widths are determined using a nearest neighbor heuristic that ensures the smoothness and continuity of the fitted function. The width of any hidden unit is taken as the RMS (root mean square) distance to the P nearest unit centers, where P is a design parameter.
3) Finally, the weights of the second layer of connections are determined by linear regression, the objective function to be minimized being the sum of the squared error as given in Equation (2.11).

2.3.3.2. Adaptive K-Means Algorithm to Determine H and x_h

The optimality of an RBFN for a particular application is largely dependent on the number of nodes in the hidden layer. By taking an excess number of nodes we may overfit the function being approximated by a higher order function. In this case, the training points may give acceptable error but the test points would give unsatisfactory results. Similarly, taking too few hidden nodes would result in a sub-optimal model.

The conventional K-Means algorithm is largely dependent on the number of clusters, K, the choice of the initial cluster centers and the order in which the data is presented. Linearly separable data are reasonably clustered by the K-means algorithm depending on the spatial properties of the training data. In training RBFNs, adaptive forms of the K- Means algorithms have been used to obtain optimum results. The method used for determining the clusters automatically is shown in the flowchart in Figure 2.7 [17].

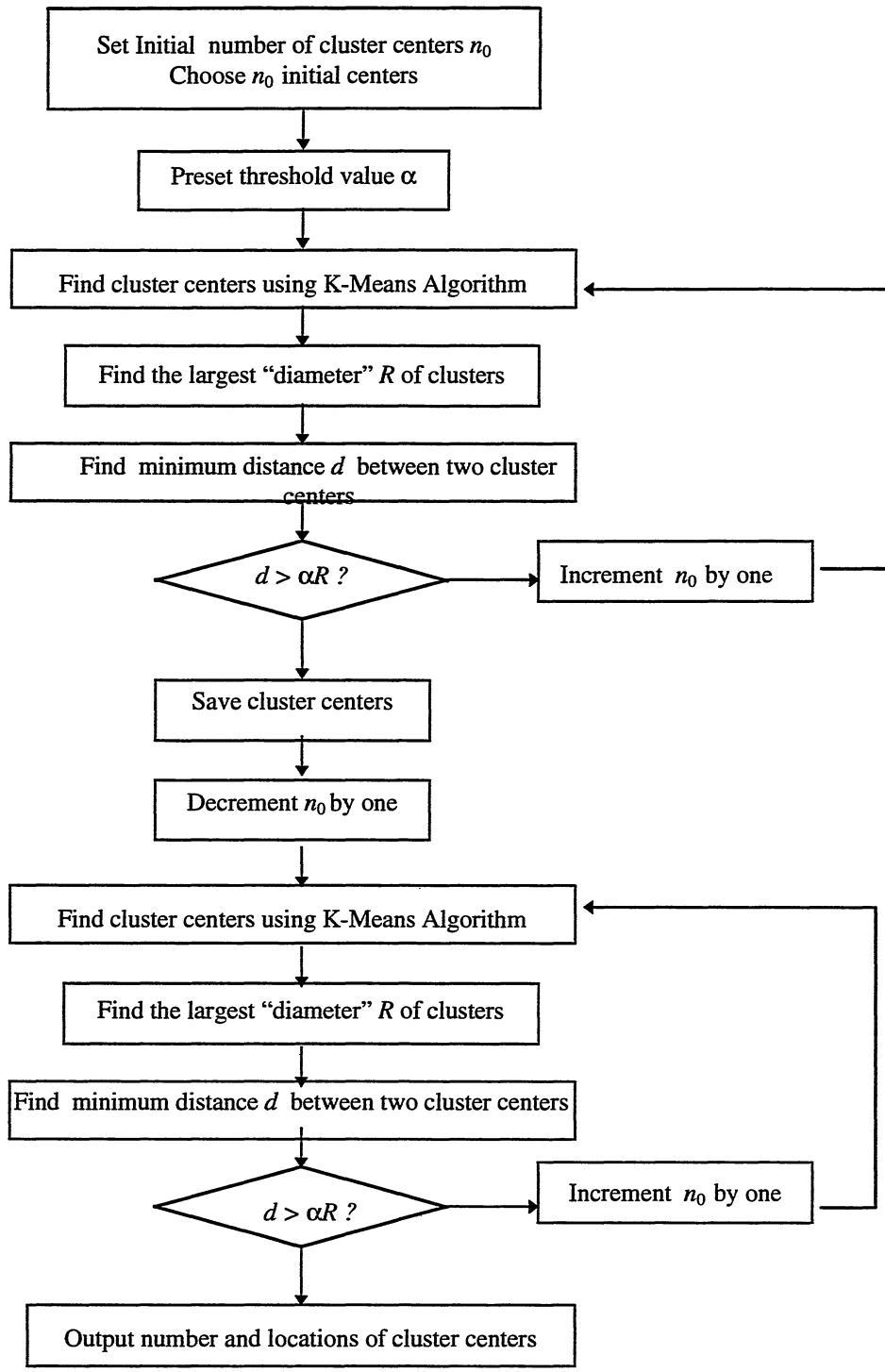

Figure 2.7 Flowchart for adaptive K-Means Algorithm [17]

In this algorithm, the number of clusters is automatically adjusted on the basis of spatial distribution of the samples. The K-Means algorithm is first applied by arbitrarily selecting the cluster centers, n_0. The minimum intercluster distance (d) is then calculated.

$$d = \min_{1 \le i, j \le n_0, i \ne j} \left\{ dist\left(X_i - X_j\right) \right\} \; for \quad i, j = 1,2,\ldots.n_0 \quad . \tag{2.14}$$

where X's are the n_0 cluster centers and *dist* is the Euclidean distance given by

$$dist(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 \ldots + (a_m - b_m)^2} \tag{2.15}$$

in an m-dimensional space.

The diameter (D_k) of the k^{th} cluster is defined as the maximum distance between two samples in cluster k. The largest diameter (R) is computed next. If x's are the points in the cluster k, the intracluster distance D_k is given by

$$D_k = \max_{i \ne j} \left\{ dist\left(x_i, x_j\right) \right\} \; for \quad i, j = 1,2,\ldots. N_k \tag{2.16}$$

where N_k is the number of points in cluster k and,

$$R = \max_{1 \le k \le n_0} \left(D_k\right) \tag{2.17}$$

When $d > \alpha R$, (where α is a preset threshold value) it means that the scatter plot of the points belonging to the largest cluster exceeds the threshold value that has been preset as a fraction of the largest diameter R. This intracluster distance can be reduced by increasing the number of clusters, n_0. Therefore, if $d > \alpha R$, the number of cluster centers is incremented. Otherwise, it is decremented. The K-Means algorithm is iterated to obtain the new cluster centers. The algorithm converges when the number of clusters does not change.

2.4. Why Do We Need a Confidence Measure ?

As discussed earlier, ANNs are used extensively in pattern recognition due to their capability of forming highly nonlinear boundaries between different classes of patterns. ANNs have many useful characteristics such as learning capability, generalization, noise and fault tolerance, etc. However, there has been a general lack of theory that allows for calculation of estimated errors on ANN solutions for validation and verification purposes.

Besides, different types of networks have different generalization properties and these generalization capabilities of ANNs have not been completely understood [2]. Due to the inherently abstruse nature of their highly parallel, dense interconnections, the reliability of a network decision has not yet been characterized.

Assumptions, whether implicit or explicit, that lead to network outputs being considered reliable may also be inappropriate, especially if the input presented to the ANN has not been covered by the training data. In general, the training data that the ANN 'learns' from, may or may not cover all possible inputs that the network may encounter in the future. Consequently, when the decision of an ANN is required critical (as in the case of nuclear power plants), the users of ANNs have to be aware of the uncertainty associated with the ANN output.

Providing a figure of merit for the ANN performance is necessary when addressing its reliability particularly with respect to test data not similar to the training data. A figure of merit in the form of an error bound on the ANN solution can help interpret the overall relationship between the input signals and the network outputs. For example, if the estimated error is too large, it can be interpreted as lack of reliability of the ANN decision for a

particular input. Accurate classification is of importance in any classification problem. However, besides improving the classification accuracy it is also important to determine quantitatively the confidence the network has in its decision. Preventive actions would be taken only if the confidence expressed is high enough.

Therefore, error estimation is very important assurance of the diagnosis obtained from a network essential in a practical implementation of the automated signal classification systems based on ANNs.

2.5. Research Contribution

The objective of this study was to develop a technique for evaluating the classification performed by a neural network in a quantitative manner.

The contribution of this thesis is twofold - Two approaches for reliability estimation have been developed based on (i) fuzzy methods and (ii) error regression model.

Classification is implicitly a binary decision. However, such a binary approach can sometimes be misleading. Two different signals or patterns may not have the same degree of belongingness to a particular class. This quality of 'degree of membership in a class' is used by humans in daily life as it conveys more information. Fuzzy concepts attempt to quantify this multivalued possibilities of a decision by replacing the more commonly used binary decision by one which can have a spectrum of values. Motivation in this area has been mainly due to the fact that utility of fuzzy set theory lies in its capability to model ambiguous or uncertain data. Since the outputs of fuzzy systems are membership values, they can be directly used to determine the certainty measure of a classification. The methods proposed to determine the confidence measure estimation involve the application of fuzzy membership

sets to the conventional methods of pattern classification. Chapter 4 describes *'fuzzy'* approaches that have been investigated for reflecting confidence in a classification decision. A fuzzy c-means clustering algorithm is first described where the inputs to the system are feature values and the outputs are membership values. This is followed by a fuzzy version of the MLP. This method incorporates fuzzy theory into conventional multi-layer perceptron neural network. Here, both the inputs as well as the outputs are membership values. Thus, an attempt is made to fuse fuzzy logic and conventional classification algorithms to achieve results that will reflect the confidence in the classification decision.

A new method for error estimation of an ANN output has been proposed and discussed in Chapter 6. This approach builds a regression model between a set of inputs describing the sources of uncertainty in an ANN and the corresponding error in its predicted output. Results of this method are presented and compared with the results obtained from existing methods.

3. LITERATURE SURVEY ON NDE RELIABILITY

3.1. Statistical Nature of the NDE Process

In the application of an NDE method, there are many factors that influence the outcome of an inspection as to the absence or presence of a flaw. In general, as we have seen in the introduction, NDE comprises the application of a stimulus to a structure and the interpretation of the response to the stimulus. Repeated inspections of a specific flaw can produce different magnitudes of stimulus response because of variations in setup and calibration of the measuring instrument. This variability is inherent in any measurement process. In addition, differences in the material properties, flaw geometry and flaw orientation, all contribute to different response magnitudes even when the flaws are of the same size. When the signal interpretation is being done either manually or in an automated fashion, the interpretation of the response is further influenced by the capability of the interpreter, the mental acuity of the inspector due to fatigue or emotional outlook, and the ease of access and environment at the inspection site. Since many critical inspection decisions are being made by skilled operators, human factors are considered primary contributors to the unreliability in the interpretation process. In the case of automated inspectors, the most important factors introducing variation in response interpretation are:

- Differences in the physical properties of flaws of nominally identical sizes
- Basic repeatability of the magnitude of the NDE signal response when a specific flaw is independently inspected by an inspector using the same equipment

- Human factors associated with inspectors
- Differences introduced by changes in inspection hardware

These factors must be addressed explicitly or implicitly in every NDE reliability experiment.

This thesis addresses issues of reliability related to automated flaw classification systems using artificial neural networks. The capabilities of these systems are highly dependent on the quality of the training data as well as the variabilities in the measurements on which they have been trained. These factors affect the reliability of flaw detection and classification. The following sections briefly describe some probabilistic as well as deterministic methods that are used for determining NDE reliability. Probabilistic concepts related to NDE reliability analysis are discussed first and then applied to compute the reliability of a neural network decision.

3.2. Probability of Detection (POD) and Probability of False Alarm (PFA)

An engineering approach to the reliability issue is based on the application of probability of detection (POD) curve as a tool for assessing NDE capabilities. POD curves denote inspection capabilities as a function of flaw size. A typical POD curve is shown in Figure 3.1. Such a curve is obtained by generating flaws in components and passing the components repetitively through an inspection process. Flaws spanning a range of shapes and sizes are considered and the responses are ordered in terms of actual flaw size. Responses are then grouped into statistically significant samples to allow calculation of a point estimate of detection for the sample group. This means that if the flaw of size a was inspected n times and detected successfully p times, the point estimate for detection for size a flaw is p/n. The point estimate of detection is plotted as a function of flaw size for that sample group.

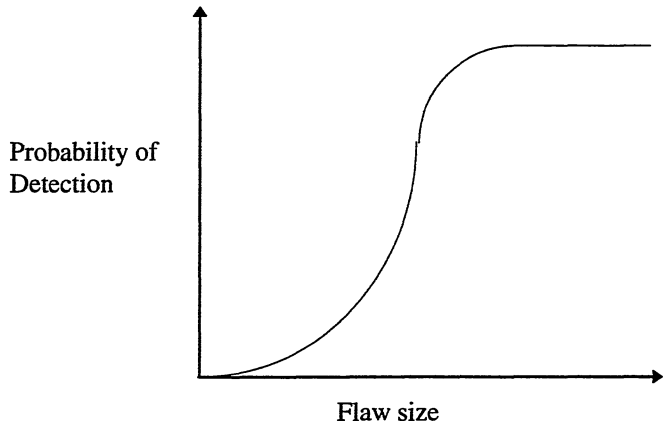

Figure 3.1 A Typical POD Curve

Successive grouping and plotting of the point estimate for detection is used to generate the POD curve. A large number of observations and a large quantity of data are required to generate an accurate POD curve, and hence to describe the capability and reliability of the NDE technique being assessed. In similar manner, POD curves may be generated to reflect other flaw parameters of interest, such as flaw depth [41]. It is also possible to extract additional information concerning the inspection and evaluation process from the POD curves. For instance, the shape of the POD curve provides a qualitative basis for assessment of the degree of control for a given data set and a criterion for grouping similar data sets [41]. Each POD curve is unique to the specificity of the inspection process, to the degrees of control in the inspection process and to the nature and distribution of the flaws. The cost of data generation, precision in data collection, and the discipline required for specific applications, have resulted in many attempts to generalize and model POD curve prediction. No satisfactory model has been generated so far.

Although the POD curve provides a graphical method for quantification of NDE reliability, all the contributing factors are not reflected by the curve. The POD curve reflects only the positive success for the inspection tool applied. There is an interdependence between the inspection stimuli and inspector responses which is schematically presented in Figure 3.2. The 'True positive' (TP) is defined as the conditional probability of the flaw being correctly identified when it was present. 'False positive' (FP) indicates the probability of a flaw being incorrectly indicated when it was actually absent. Similarly, 'False negative' (FN) misses a flaw and 'True negative' correctly indicates a safe condition. Therefore,

Total opportunities for positive calls = TP + FN

Total opportunities for negative calls = FP + TN

This then helps us to define POD and another independent probability called the probability of false alarm (PFA) in terms of the above conditional probabilities as,

$$POD = TP / (TP + FN), \text{ and} \tag{3.1}$$

$$PFA = TN / (FP + TN) \tag{3.2}$$

		Stimuli: POSITIVE	Stimuli: NEGATIVE
Response	POSITIVE	True Positive TP	False Positive FP
	NEGATIVE	False Negative FN	True Negative TN

Figure 3.2 Interdependence of response and stimuli

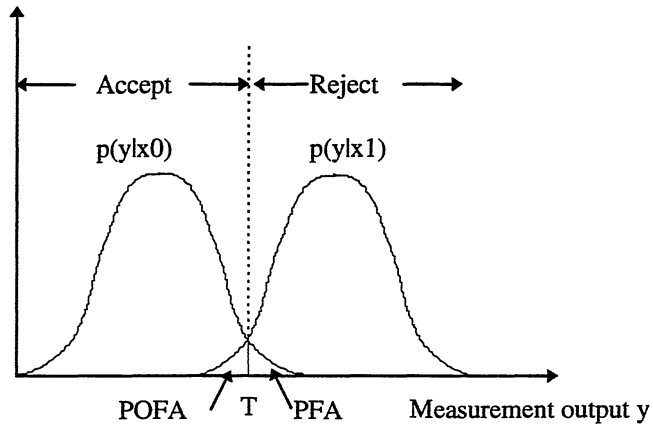

Figure 3.3 The pdfs of peak amplitude of signal with and without flaw

The probability of detection of a particular flaw of a given size using a given measurement system can also be determined by generating a conditional probability density function (pdf) [42] of the measurement signal as shown in Figure 3.3. This is given by the following equation,

$$POD = \int_{T}^{\infty} p(y|x1)dy \tag{3.3}$$

The figure shows the distributions of the peak amplitude of the signal in the absence of a flaw, p(y|x0), and in the presence of a flaw, p(y|x1). To decide whether the observed response is a flaw or not, a threshold T is chosen such that all signals above the threshold are considered flaws. When the flaw and noflaw pdfs overlap, two types of errors result. The first type of error is '*false alarm*' which is the FP region in Figure 3.2. The second type of error is '*false acceptance*' corresponding to FN in Figure 3.2. The probability of false alarm (PFA) is computed as

$$PFA = \int_{T}^{\infty} p(y|x0)dy \tag{3.4}$$

The probability of false acceptance (POFA) is defined as

$$POFA = \int_{-\infty}^{T} p(y|x1)dy \tag{3.5}$$

As the flaw size becomes smaller and smaller, the mean of p(ylx1) decreases and the p(ylx1) curve shifts towards the left. This results in an increase in the overlap between the two distributions and the corresponding POD is reduced. Similarly, as the flaw size increases, the p(ylx1) curve shifts towards the right, decreasing the overlap between the two distributions and consequently increasing the POD.

However, the POD curve does not directly reflect the nature of the calibration of the inspection tool or the decision criteria used in the generation of the curve. Also, there is a further need in the industry to classify the signal observed as belonging to one of several classes, where a simple accept-reject decision is not sufficient. Automated classification systems including neural networks are becoming increasingly popular for performing signal classification. The overall classification accuracy depends on both the accuracy of the classification tool and on the statistical nature of the measurement process.

3.3. Method for Error Estimation on the Outputs of ANNs

As discussed before, in an automated signal classification system based on an ANN, there are several sources of uncertainties. These include the distribution of the available data which strongly influences the function that the ANN approximates. If dense and uniformly distributed data is available, the neural network closely approximates the mapping between

the inputs and the outputs. However, if the data is not sufficient and not uniformly distributed, regions with scarce data will have a poorer fit than regions with dense data. In addition, it is possible that in the training process the network error converges to a local minimum instead of a global minimum. All these factors contribute to an unreliable network decision.

A method developed for estimating the confidence of a decision made by an ANN is briefly described in the following section.

3.3.1. Stacked Generalization

One of the earliest schemes, namely stacked generalization, was proposed by Wolpert in [34] for minimizing the generalization error rate of one or more generalizers. As the name suggests, it consists of stacking two generalizers together. The primary generalizer performs classification or function approximation while the second generalizer estimates the error in the output from the primary generalizer for a novel input.

Now how would this translate to an ANN? Under certain restrictions, a backpropagation ANN providing a function mapping can be considered as a generalizer. The restrictions are as follows [35]:

- Let L be a training data set in $\boldsymbol{R}^m \boldsymbol{X} \boldsymbol{R}^n$ space where m is the dimension of the input space and n is the dimension of the output space. The first restriction requires that the training of the generalizer be independent of the order in which data is presented.
- Subject to some training accuracy, when a generalizer is presented with an input from the training set L, the generalizer must produce an output which is the same as the one

corresponding to this input in L. This means that for an input which is not new to the network, the output should be the same as the one on which it was trained.

- The input-output mapping of the generalizer must be single valued. This means that two or more outputs corresponding to a single input vector are not allowed.
- To map the desired function, a generalizer requires at least k training pairs in the learning set, $k > m$, where m is the dimension of the input vector.
- The components of all input vectors in the training set lie on a hyperplane having a dimension which is the same as the dimension of the input of the generalizer.

Since the multi-layer perceptron with backpropagation can be made to satisfy these restrictions, it qualifies as a generalizer. Stacked generalization can, therefore, be implemented on ANNs. The essence of the method of stacked generalization is as follows: Let **G** be a generalizer (an ANN). A training set $\boldsymbol{L}$ of all input-output pairs is partitioned into two sets, namely the singleton set $\{(\boldsymbol{x}, \boldsymbol{y})\}$ and $\{\boldsymbol{L}\text{-}(\boldsymbol{x}, \boldsymbol{y})\}$, as shown in Figure 3.4. A novel input is represented by $\boldsymbol{q}$ lying outside of $\boldsymbol{L}$. Given this partition, the network **G** is trained using the data in $\{\boldsymbol{L}\text{-}(\boldsymbol{x},\boldsymbol{y})\}$. **G** is then presented with $\boldsymbol{x}$ as input. The output $\boldsymbol{g}$ of **G** and the difference vector from $\boldsymbol{x}$ to its nearest neighbor in $\{\boldsymbol{L}\text{-}(\boldsymbol{x},\boldsymbol{y})\}$, ξ, are recorded.

$$\xi = min\{dist(\boldsymbol{x}_i, \boldsymbol{x})\} \quad for\ i=1,2,.....\boldsymbol{N\text{-}1}, \tag{3.6}$$

where $\boldsymbol{N}$ is the total number of points in $\boldsymbol{L}$ and $dist(\boldsymbol{a}, \boldsymbol{b})$ is as defined in Equation (2.15).

In general, since **G** has not been trained with the pair $(\boldsymbol{x},\boldsymbol{y})$, the output $\boldsymbol{g}$ will differ from $\boldsymbol{y}$. Thus for a combination of $\boldsymbol{x}$ and ξ, there is an error of $(\boldsymbol{g} - \boldsymbol{y})$ in the network decision at the output.

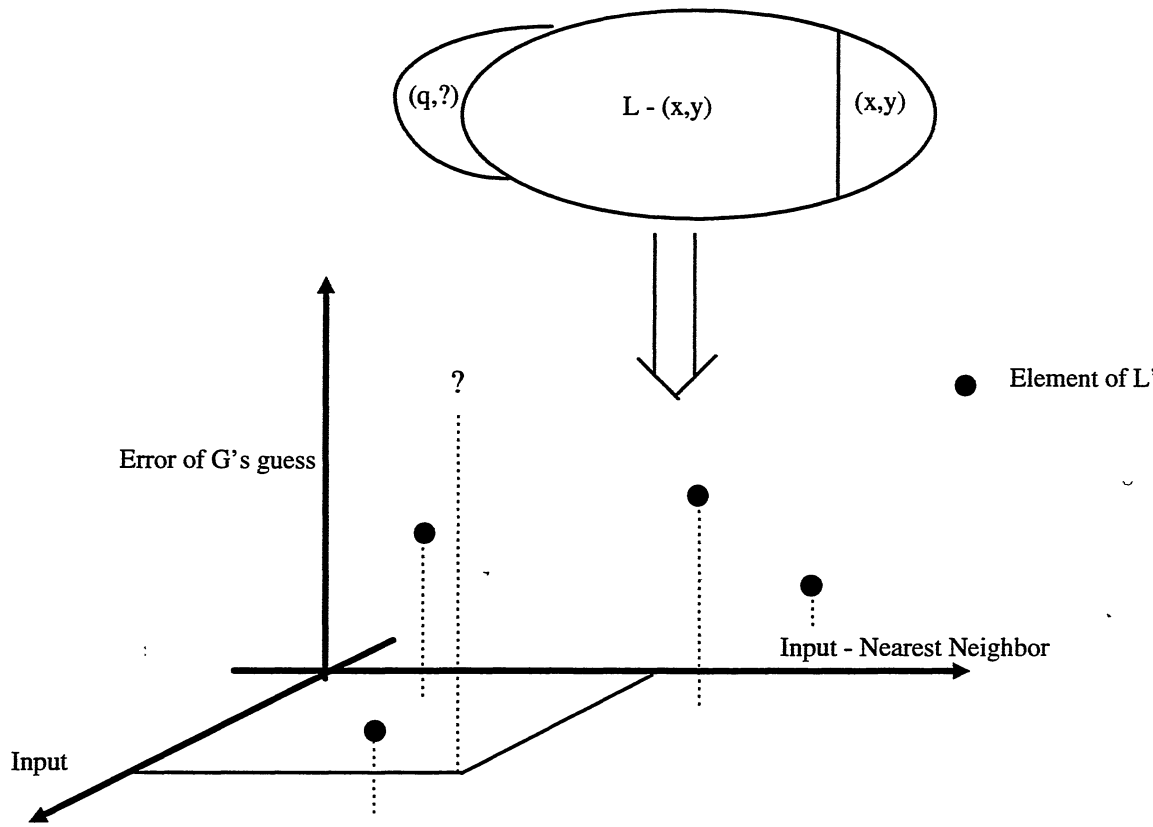

Figure 3.4 Stacked generalization for a single generalizer

This information can be cast in a new input-output domain where point (x, ξ) is the input and error (g - y) is the corresponding output. Choosing different partitions of L gives us other input-output pairs in the new domain. Taken together, these points constitute a training set L'. Another generalizer (neural network) **F** is trained on the training set L'.

Next, G is trained on the entire set L and then presented the new input q. Let p be the corresponding output of **G**. q and the distance from q to its nearest neighbor in L are input to **F**. The output of **F** represents the error in the output of **G** to the input q. Adding this estimated error (or fraction thereof) back to the output g gives the final output corresponding to x. A flowchart depicting this method is as shown in Figure 3.5.

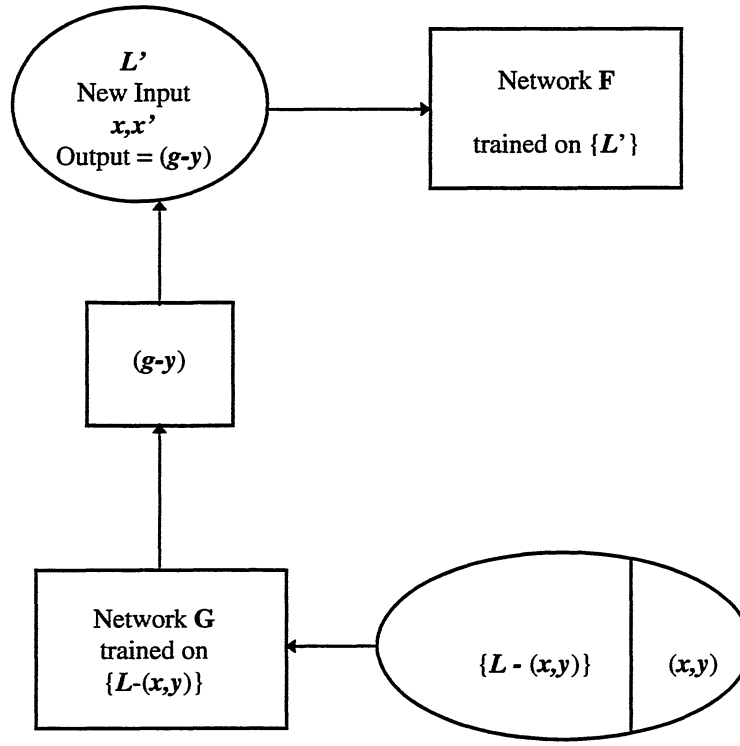

Figure 3.5 Flow Diagram for the different steps in stacked generalization

The results of implementing this method are presented at the end of this chapter. In literature, this method is also called the Cross Validation Partition Criterion (CVPC).

3.4. Implementation and Results for Stack Generation

The data used for implementation of the stacked generalization algorithm consists of ultrasonic weld inspection input signals to be classified into cracks, counterbores and rootwelds. A detailed description of this signal classification problem was given earlier.

The raw signals acquired by the ultrasonic data acquisition system were 1800 points long. Preprocessing the signals involved reducing its dimension by extracting discriminating

features. As mentioned in Section 2.2, the optimality of the features selected is not central to this work. Only two significant features were selected so that in a two dimensional feature space the analysis was tractable. The features used are the peak value and the pulse width of the raw ultrasonic signal. The training data used in the implementation included a total of 93 signals distributed as 46 crack signals, 21 counterbore signals and 26 rootweld signals. The test data for validation consists of 10 cracks, 10 counterbores and 10 rootwelds. The plots of the training and test data in the two dimensional feature space are shown in Figure 3.6 and Figure 3.7 respectively.

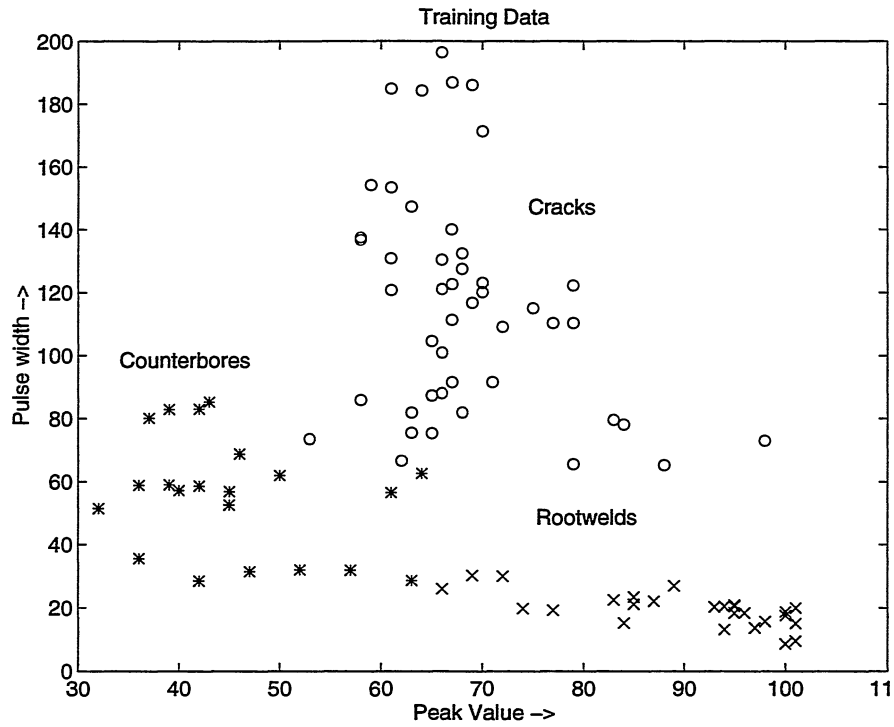

Figure 3.6 Training Data

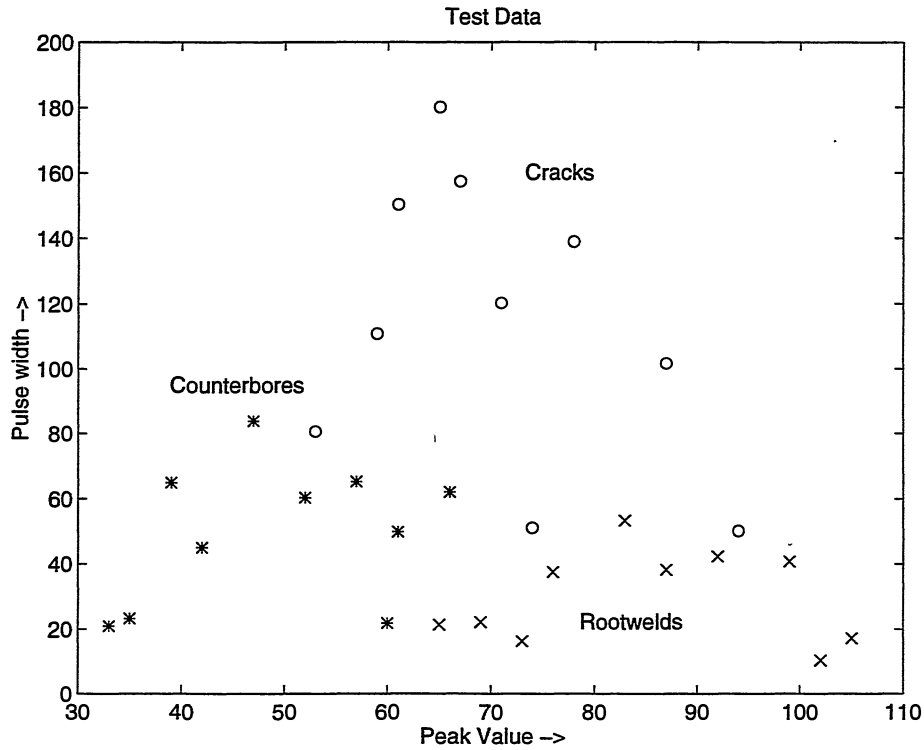

Figure 3.7 Test Data

The primary MLP neural network architecture had 2 input nodes, a single hidden layer consisting of 10 nodes and three output nodes each one representing one of the three classes. It was trained using backpropagation algorithm.

The steps involved in the implementation of stack generalization are as follows:

1. Partition L into L_1 and L_2 such that

   $L_2 = \{x_N\}$ and

   $L_1 = L - L_2 = \{x_i\}_{i=1}^{N-1}$.

2. Train the primary network G using L_1.

3. Test network with element x_N in L_2. Let d be the desired output and y be the network output.
4. Calculate the error $\varepsilon = |d-y|$.
5. Find the nearest neighbor of x in L_1.

   Let $X_1 = \min_{x_i \in L_1} dist\{x, x_i\}$.
6. Repeat steps 1 through 5 with $L_2=\{x_i\}$, $i=1,2,...N$
7. The set of input output pairs $\{(x, X_1), \varepsilon\}$ for the training set L' for the second network **F.**

The second network **F** was trained to predict the error for an input test data. The test data set consisting of 30 signals was presented to the primary network **G** which gave the classification result. The output of **F** predicted the error in the classification of **G**. The secondary network had two hidden layers with 22 and 8 nodes, respectively.

The results are presented in Table 3.1.

The first three columns indicate the three outputs of the primary network **G**. The first ten samples are crack signals and the desired output for this class is 0-1-0. The next ten samples are counterbores and the last ten are rootwelds with desired outputs 1-0-0 and 0-0-1 respectively. The output with the maximum activation is rounded off to 1 and the signal is assigned to the class represented by that node. The last column indicates the misclassifications with a '*'. The fourth column represents the output from the secondary MLP. In order to use this information for reliability calculation, the error estimated by the network **F** is applied as an error bound on each value obtained at the three output nodes of the primary network **G**. Let O_i, $i=1,2,3$ be the output of the three nodes of **G**. If the range of values $O_1 \pm E$ overlaps with either of the intervals $[O_2 - E, O_2 + E]$ or $[O_3 - E, O_3 + E]$, the

decision is said to be ambiguous. As seen in Table 3.1, column 4, in some of the misclassifications such as points 5 and 29, the error estimated by **F** was high. However, this trend did not appear for test points 10, 14, 21, 22, 27 which had also been misclassified. Also, points 23, 26 and 28 had a relatively higher error estimate inspite of correct classification.

Table 3.1: Results of Stack Generation

Signal no.	**Output from neural network G**			**Output from ANN F** *estimated error (E)*	
	node for counterbores	node for cracks	node for rootwelds		
1	0.0000	1.0000	0.0000	0.0004	
2	0.0000	1.0000	0.0000	0.0003	
3	0.0000	1.0000	0.0000	0.0004	
4	0.0000	1.0000	0.0000	0.0003	
5	0.0001	0.0316	0.6432	0.0024	*
6	0.0141	0.9980	0.0000	0.0008	
7	0.0000	1.0000	0.0000	0.0003	
8	0.0000	1.0000	0.0000	0.0003	
9	0.0000	1.0000	0.0000	0.0004	
10	0.9541	0.0002	0.0029	0.0008	*
11	1.0000	0.0000	0.0000	0.0000	
12	1.0000	0.0000	0.0000	0.0001	
13	0.8082	0.1100	0.0000	0.0001	
14	0.4545	0.6871	0.0000	0.0003	*
15	0.9998	0.0000	0.0000	0.0000	
16	1.0000	0.0000	0.0000	0.0000	
17	1.0000	0.0000	0.0086	0.0000	
18	1.0000	0.0000	0.0000	0.0000	
19	0.8621	0.0915	0.0000	0.0001	
20	1.0000	0.0000	0.0000	0.0000	
21	1.0000	0.0000	0.0787	0.0000	*
22	0.0510	0.0239	0.0262	0.0005	*
23	0.0000	0.0011	0.9874	0.0051	
24	0.0000	0.0000	0.9998	0.0003	
25	0.9996	0.0000	0.8735	0.0003	*
26	0.0059	0.0000	0.8662	0.0055	
27	1.0000	0.0000	0.2847	0.0001	*
28	0.0003	0.0008	0.9024	0.0033	
29	0.9702	0.0000	0.1479	0.0042	*
30	0.0000	0.0000	0.9999	0.0001	

'*' represents misclassified signals

The error estimate obtained from **F** indicates that the range of allowable values at the output nodes defined by the intervals $[O_i - E, O_i + E]$ did not consistently show an overlap in case of ambiguous signals as required. In many cases it failed to indicate ambiguous decisions completely. This implementation showed that the method of stack generation was not suited for the ultrasonic signal classification problem under consideration. More importantly, the drawback with this method is the use of a second neural network to evaluate the primary neural network which automatically raises the question of reliability of the second neural network.

4. FUZZY METHODS FOR CONFIDENCE MEASURE

4.1. Motivation to Use Fuzzy Logic

Utility of fuzzy set theory lies in its capability to model ambiguous or uncertain data. Since the outputs of fuzzy systems are values of membership or "*belongingness*" to a class, they are used to determine a measure of certainty. Two approaches have been taken and results have been presented. The first approach uses the fuzzy version of the intuitive K-nearest neighbor algorithm. In this algorithm, the inputs to the system are feature values whereas the outputs are membership values. The second method incorporates the fuzzy theory into the conventional multi-layer perception neural network. Here, both the inputs as well as the outputs are membership values. The results are compared and possible ways of improving their performance are discussed. This kind of a classification system finds an application in the development of a sophisticated inferencing expert system. This implementation is able to characterize, with a certain confidence, the different types of ultrasonic signals obtained as reflections from cracks, counterbores or rootwelds. Besides correct classification, the output of the system is also representative of its confidence in the classification decision.

4.2. Theory of Fuzzy Sets

Given a universe of objects $\boldsymbol{U}$, a conventional crisp subset $\boldsymbol{A}$ of $\boldsymbol{U}$ is defined by specifying the objects from $\boldsymbol{U}$ that are members of $\boldsymbol{A}$. This is characteristically written as

$$u_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases} \quad \text{or} \quad u_A: \boldsymbol{U} \rightarrow \{0,1\} \tag{4.1}$$

for all $x \in \boldsymbol{U}$. Fuzzy sets are derived by generalizing the concept of a characteristic function to a membership function u: $\boldsymbol{U} \rightarrow [0,1]$. Most crisp operations and set properties have analogs in fuzzy set theory [6].

The advantage provided by fuzzy sets is that the degree of membership in a set can be specified. This can be especially advantageous in pattern recognition, where frequently, objects may not clearly be members of one class or another. Using crisp techniques, an ambiguous object will be assigned to a single class with a definiteness may not be justified. On the other hand, fuzzy techniques will specify to what degree the object belongs to each class, offering information that is more useful in a practical implementation.

Given a sample of vectors $\{x_1, x_2, x_3, \ldots, x_n\}$, a fuzzy c partition of these vectors specifies the degree of membership of each vector in each of the c classes. This is represented by the matrix $\boldsymbol{U}$ of size $c \times n$, where each element of the matrix

$$u_{ik} = u_i(x_k) \qquad \textit{for } i=1,2,\ldots c \quad \textit{and } k=1,2,..n \tag{4.2}$$

is the degree of membership of x_k in class i. The following partitions must be true for $\boldsymbol{U}$ to be a fuzzy c partition:

1. $\sum_{i=1}^{c} u_{ik} = 1$

2. $0 < \sum_{k=1}^{m} u_{ik} < 1$

3. $u_{ik} \in [0,1]$

The first property implies that the vector's memberships in the c classes must sum to 1 for mathematical tractability. The second property implies that the total membership contribution of m samples assigned to the i^{th} partition should not exceed n, the number of vectors in the given set. The last property reiterates the fuzzy requirement for the membership values to have a continuous spectrum of values between 0 an 1, unlike the crisp case.

4.3. K-Nearest Neighbor (KNN) Algorithm

The K-means or KNN algorithm represents one of the most popular methods for clustering data. Several variations of this algorithm are found in literature. The common feature of all these algorithms is that they cluster samples based on the Euclidean distance. The conventional K-means algorithm forms K clusters of samples and assigns a class label to each sample. The fuzzy K-Nearest Neighbor algorithm [5] is different from its conventional version in that it assigns class membership to a sample vector rather than assigning the vector to a particular class. The advantage of this method is that no arbitrary assignments are made by the algorithm. In addition, the membership values provide a level of assurance to accompany the resultant classifications. For example, if a vector is assigned a membership value of 0.9 in one class and a membership value of 0.05 in the two remaining classes, it is reasonable to assign the vector to the class of membership value of 0.9. On the other hand, if the vector is assigned memberships 0.55, 0.44 and 0.01 in class one, two and three, respectively, then classification of the vector based on these results should be made with hesitation. However, it is certain that the vector does not belong to class three. In such a case, the vector must be examined further to determine its classification, since it exhibits almost equal degrees of membership to both classes one and two. Clearly, the membership

assignments can be useful in the classification process. The flowchart of this algorithm is presented in Figure 4.1

The first step in this technique is the assignment of membership values to the known samples. While assigning membership values to the known samples, each sample from the training set is considered one at a time. The value of k, (the number of neighbors considered during the assignment of membership values to the training samples) nearest neighbors to each sample $\mathbf{x}$ is then chosen and the membership is then assigned to $\mathbf{x}$ in all classes according to Equation (4.3) [5]. The membership sample vector $\mathbf{x}$ to the j^{th} class is given by

$$\mu_j(\mathbf{x}) = \begin{cases} 0.51+(n_j/k)*0.49 & if \quad j=i \\ (n_j/k)*0.49 & if \quad j\neq i \end{cases} \tag{4.3}$$

when $\mathbf{x}$ belongs to class i.

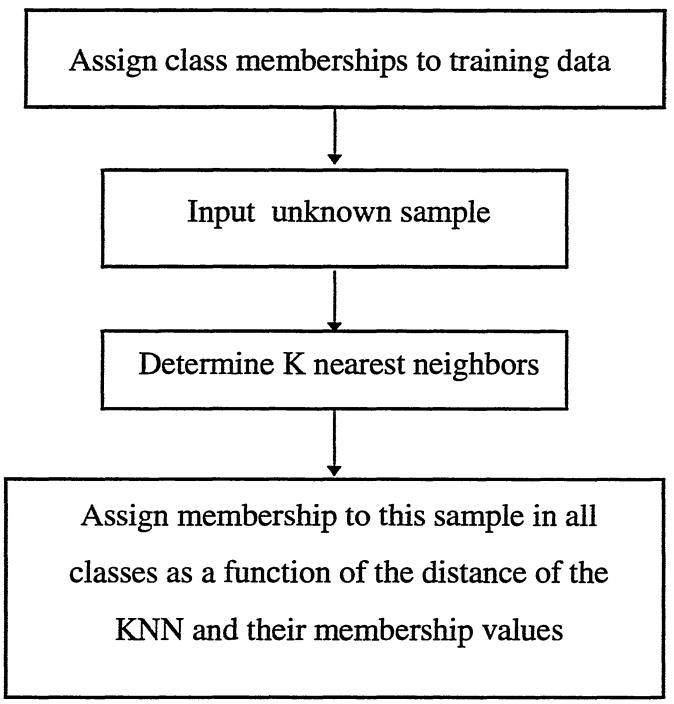

Figure 4.1 Flowchart of the fuzzy KNN algorithm

This value of k need not be the same as K, the number of nearest neighbors considered during the assignment of membership values to an unknown sample. In Equation (4.3), k is the number of neighbors that are considered and n_j is the number of neighbors from k belonging to the j^{th} class. This method attempts to *`fuzzify'* the memberships of the known samples, which are within the regions of intersection in the sample space, and leaves the samples that are well away from this area with a complete membership in the known class. As a result, the decision of an unknown sample lying close to the boundary will be influenced to a lesser extent by the known samples that are in the *`fuzzy'* area of the class boundary. The next step is to determine the membership values of the unknown sample, with respect to the training data. $\mu_i(\mathbf{x})$ is computed using Equation (4.4) [5].

$$\mu_i(\boldsymbol{x}) = \frac{\sum_{j=1}^{K} \mu_{ij} \left(1/\left\|x-x_j\right\|\right)^{\frac{2}{m-1}}}{\sum_{j=1}^{K} \left(1/\left\|x-x_j\right\|\right)^{\frac{2}{m-1}}} \tag{4.4}$$

where $\mu_i(\boldsymbol{x})$ is the membership value of vector $\boldsymbol{x}$ to i^{th} class, μ_{ij} is the membership value of the j^{th} nearest neighbor to the i^{th} class, $\boldsymbol{x_j}$ is the j^{th} nearest neighbor from the known samples and K is the number of neighbors. The variable m determines how heavily the distance is weighted when calculating each neighbor's contribution to the membership value. When m=2, the contribution of the neighboring point is weighted by the reciprocal of its distance from the point being measured. As m is increased, the neighbors are more evenly weighted, whereas when m is decreased, the neighbors are more heavily weighted.

4.3.1. Implementation of Fuzzy KNN Algorithm

The data used in this implementation are the same feature vectors obtained as explained in section 3.4.

The first step in the implementation of the fuzzy K-Nearest Neighbor algorithm was the assignment of class memberships to the training points. This was done using Equation (4.3) with k=8. Since, there were three classes under consideration, namely, cracks, counterbores and rootwelds, a set of μ_{ij}'s were obtained with i= 1,2,..... 93 and j=1,2,3 (number of classes). The test points were chosen such that at least some were in the regions close to the possible boundary between two classes. The test points were now assigned membership values to all three classes using Equation (4.4) with K=8 and m=2. A decision was said to be ambiguous if the most significant output value was around 0.5. The results of the application of this KNN algorithm for classifying ultrasonic NDE signals are tabulated in Table 4.1. The first column indicates the number of test points in each class. The second column represents percentage of accurate classification. The last column indicates percentage of the misclassified signals where the network decision indicated uncertainty.

Table 4.1: Results of the Fuzzy K-Nearest Neighbor Algorithm

Classes	Total number of test data	Correct Classification	Ambiguous misclassifications
Cracks	10	80%	100%
Counterbores	10	70%	66.67%
Rootwelds	10	70%	33.33%

The table shows that in the two-dimensional feature space (pulse width and peak value), the classification accuracy for cracks was 80%. 8 out of 10 cracks were correctly classified. Both the misclassified signals had the significant output membership values less than or equal to 0.5 ± 0.15. Thus, all the misclassified test points indicated ambiguity. Among the counterbore signals, three signals were assigned to the wrong class. These points occur close to class boundaries and 2 out of the 3 misclassifications indicated ambiguity. The rootwelds did not perform as well and only 1 of the 3 misclassifications reflected an uncertainty.

The fuzzy KNN method is only a simple demonstration of the use of a fuzzy approach for quantifying the confidence in a classification decision against the conventional binary decision process which can be incorporated in an automated signal classification system. A possible improvement in the results could be obtained by using features that are more discriminatory than the simple features considered here. The next section extends the fuzzy approach to the commonly used Multi-Layer Perceptron.

4.4. Introduction to Fuzzy MLP

As described in Chapter 2, ANNs have been used successfully in a large number of fields to model dynamic and non-linear systems, chaotic chemical systems and other chemical reactions, system identification and control, process fault diagnosis in NDT etc.

In the fuzzy MLP approach, signals are classified into one of the three desired classes, namely, cracks, counterbores and rootwelds, with a certainty measure reflected by a membership value to a class.

Each input feature vector ***Fj*** is expressed in terms of membership values in terms of low, medium and high.

$$F_j' = \{F_{j1}^{low}, F_{j1}^{medium}, F_{j1}^{high}, \ldots\ldots F_{jn}^{low}, F_{jn}^{medium}, F_{jn}^{high}\}$$

That is, an *n*-dimensional feature vector is converted into a *3n*-dimensional feature vector. In the simple case studied here, there are two features under consideration. After converting these feature values to membership values, a *6*-dimensional vector is obtained. The membership assignment is done using Gaussian membership sets:

$$F_{ji}^{set} = e^{-(F_{ji} - c_i^{set})^2 / 2(\sigma_i^{set})^2} \tag{4.5}$$

where $i = 1,2,\ldots n$ and c_i^{set} *and* σ_i^{set} are the center and width of the membership function.

The output membership values are then calculated. For a *3*-class problem, the membership of the *i*th pattern to class *k* is defined by Equation (4.6) [1].

$$\mu_k(\boldsymbol{F}_i) = \frac{1}{1+\left(\frac{z_{ik}}{f_d}\right)^{f_e}} \tag{4.6}$$

where

$$z_{ik} = \sqrt{\sum_{j=1}^{n}\left[\frac{F_{ij} - o_{kj}}{v_{kj}}\right]^2} \tag{4.7}$$

is the weighted distance of the pattern $\boldsymbol{F}_i$ to the center of the *k*th class, f_d and f_e are the denominational and exponential constants, $\boldsymbol{O}_k$ is the mean vector, and $\boldsymbol{V}_k$ is the standard deviation of the features in the *k*th class. The value of f_d decides the effective contribution of z_{ik} to the calculation of $\mu_k(\boldsymbol{F}_i)$. This value of f_d is decided by the range of the values of z_{ik}

obtained. If z_{ik} has a range of values significantly greater than 1, f_d is kept closer to 1. On the other hand if the range of z_{ik} is around 1, f_d is made very small (closer to 0), thus, ensuring that the denominator of (4.6) contributes to the corresponding membership value. Similarly, the value of f_e decides the weightage that is given to z_{ik} in contributing to the output membership value.

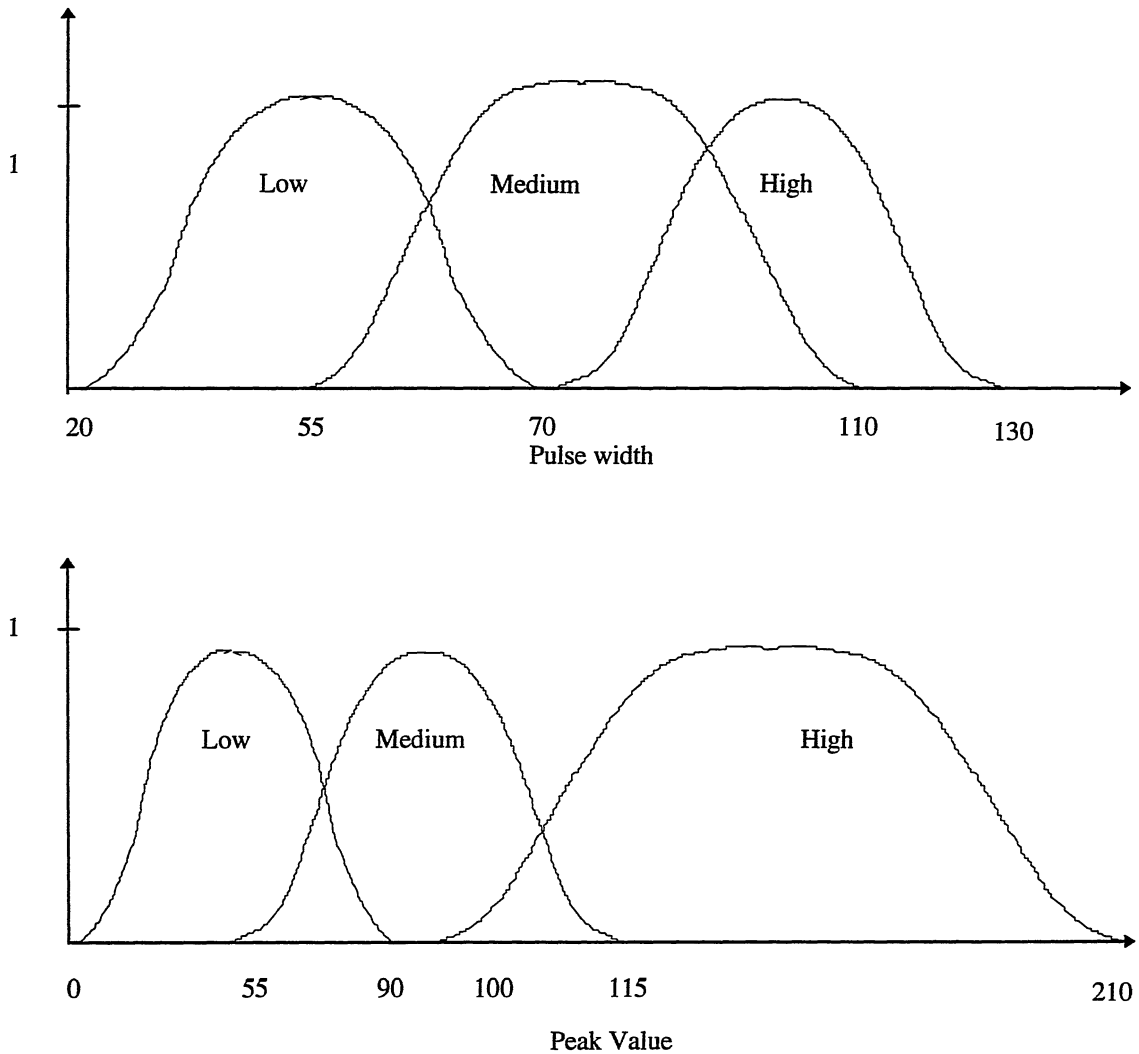

Figure 4.4 Membership sets for the inputs

This method of calculating the weighted distance is suitable for the application under consideration because the distance is inversely proportional to the variance of the features in a given class. This means that features having a larger variance are given less importance in influencing the output decision than features having smaller variance.

The multi-layer perceptron is trained with these input-output pairs before testing with unknown data.

4.4.1. Implementation

The data used in the implementation of the fuzzy KNN is used once again in the implementation of the fuzzy MLP. This implementation requires that each input feature vector, **x**=[peak_value, pulse_width] be expressed in terms of membership to each of the sets low, medium and high. The limits of these membership sets formed for the available training data are shown in Figure 4.4. Gaussian membership functions were used. Thus the input signal which was originally a two dimensional feature vector is now converted to a *3x2*-dimensional input vector. The output membership values for the corresponding training input points are obtained using equations (4.6) and (4.7) with f_d=1and f_e=2. Using these inputs and outputs, a single hidden layer MLP with 7 hidden nodes, was trained.

The results are presented in Table 4.2. They show an improved classification accuracy in the case of fuzzy MLP as compared to the fuzzy KNN. This improvement could be attributed to a neural network's capability of modeling non-linearities. However, the issue of importance is the classification reliability rather than classification accuracy.

Table 4.2: Results of the Fuzzy Multi-layer Perceptron

Classes	Total Number of Test Data	Correct Classification	Ambiguous Misclassifications
Cracks	10	80%	100%
Counterbores	10	70%	33%
Rootwelds	10	90%	100%

The output values are the confidence values since the network was trained to 'learn' an output that was a membership value, unlike conventional class labels. In the case of the misclassified signals, the output values were in the range 0.5 ± 0.1 for the crack and the rootweld signals. However, two counterbore signals were misclassified as cracks or rootwelds with a high degree of confidence, i.e. output at the respective nodes > 0.8. This can be explained as due to the fact that the misclassified counterbores were very close to both crack and rootweld training points in the given feature space.

The method of using fuzzy MLP gives us more information than a conventional MLP. The ambiguous points can then be further investigated more rigorously thereby reducing the effort and computation required to process large volumes of data.

4.5. Conclusion

In the KNN classifier with the nearest neighbor sample membership assignment, the number of misclassified vectors with high assigned memberships (greater than 0.8) in the wrong class is a small percentage of the total misclassified signals. In addition, the correctly classified signals were given relatively higher membership values in their own classes than in

other classes. Therefore, the nearest neighbor initialization technique does produce membership assignments that give an indication of the degree of correctness of classification. In the fuzzy MLP, output values are direct indications of the confidence of membership of input signals to the assigned class. Therefore, these reflect the certainty of the classification decision.

5. AN ANN ARCHITECTURE THAT COMPUTES ITS OWN RELIABILITY

5.1. Introduction

As mentioned in the previous chapter, it has been shown that an MLP with a single hidden layer and trained using backpropagation is sufficient to approximate any function given adequate number of nodes in the hidden layer. The Radial Basis Function Network (RBFN) has been shown to have a similar capability to represent arbitrary functions [32]. One of the earliest work in estimating the reliability of the RBFN was done by J. A. Leonard *et al* [3], where a method is proposed using an RBFN that computes its reliability. The underlying idea for determining the reliability of network decision is explained in detail in this chapter. This method is applicable to functional approximation problems. A suitable modification of the method for ultrasonic NDE classification problem is also described.

5.2. Why RBFNs ?

The characteristic feature of RBFNs is the concept of a local neighborhood. RBFNs partition data into local clusters and limit the distance over which data may influence its prediction. Empirical non-linear models of process data have been constructed using ANNs. Since network models are not based on any underlying physical theory and are highly non-linear, their predictions are not expected to be reliable when extrapolating beyond the range of the original training data. In addition, it is difficult to recognize when the network is extrapolating, especially if the inputs are correlated. The distribution of the training data may be non-uniform which implies that there exist areas of poor local fit. The aim is to extend the

network being used such that it has the capability of reflecting its reliability by indicating local regions of poor fit. Conventional methods for training ANNs use only global measures of goodness of fit, usually in terms of the sum squared training error given by Equation (5.1).

$$E = \sum_{j=1}^{N} (\hat{y}_j - y_j)^2 \tag{5.1}$$

where N is number of network outputs and y and $\hat{y}$ are the target and network output respectively.

This, however, may not strongly reflect local regions of poor fit. A separate goodness of fit is thus needed for each output, since different dependent variables may be fit with different accuracy. Thus, the inputs to the RBFN are still the same feature vectors as used in Chapter 4. The number of output nodes, however, increases. M outputs will require an additional M outputs to indicate the accuracy of each model prediction, and one output to indicate when the network is operating in a region of insufficient training data. The block diagram is shown in Figure 5.1.

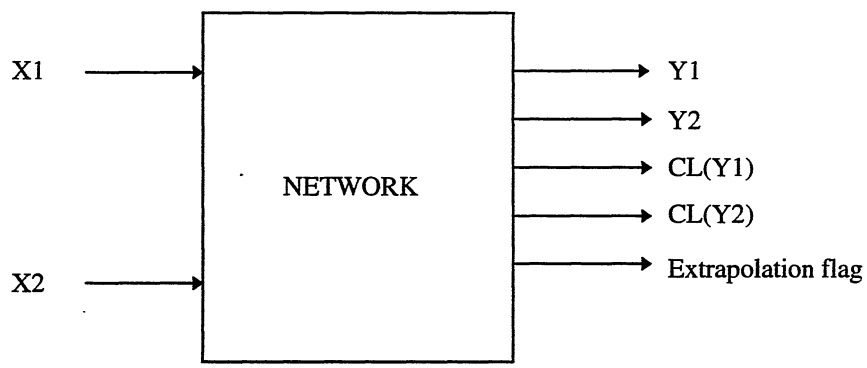

Figure 5.1 Block diagram of network determining reliability

5.3. Reliability Definition

Reliability is determined by two factors:

Extrapolation: Whether or not the model is being applied in a domain of the independent variables where training data were available.

Local goodness of fit: How accurate the is model for given independent variables.

In this approach, we can include extra outputs indicating extrapolation and confidence limits on network predictions. Important features of this approach are:

1) There is no assumption of specific distribution for the data. This is of particular interest since the data which is being processed has an unknown probability density function.

2) It is a non-parametric, non-convex method for computing reliability [3].

5.4. Reliability Measures

The determination of reliability consists of two steps:

1) The first step is to check if the model is extrapolating. If it is, the model output is unreliable since the network has not satisfactorily learned the output associated with the given input.

2) If the network is not extrapolating, model fit accuracy needs to be estimated. This is given as a confidence interval on each network output.

5.4.1. Extrapolation/Data Density Measure

Extrapolation is defined as any local region of input space with little or no training data to support a model prediction. This parameter is, therefore, linked with the estimation of the density of the training data in the region where the test data appears. The goal of this

measure is to determine whether there is sufficient training data in the vicinity of the test point to make a reliable prediction. A consequent warning is generated if the local density of the training points falls below a certain threshold.

The transfer function of an RBF hidden unit is like a multivariate Gaussian given by

$$a_h = \exp(- \| \mathbf{x} - \boldsymbol{x}_h \|^2 / \sigma_h^2) \tag{5.2}$$

where a_h is the output of the unit h in the hidden layer given the input $\mathbf{x}$. $\boldsymbol{x}_h$ and σ_h are the center and width, respectively, of the transfer function at the hidden node h. Since each RBF unit is centered on a subset of the training data by the clustering algorithm, as the test point moves away from the training data, the value of the maximum activation will decrease. A small value of the activation, say < 0.5, would then indicate extrapolation. The value of the optimum threshold is entirely problem dependent. While this is an obvious and intuitive definition for extrapolation, it indicates how far the test data is from a training data and does not indicate the amount/density of training data. Hence, it is not an ideal measure.

A more satisfactory measure for extrapolation that has been proposed is the local probability density function (pdf) of training data. If it is possible to calculate the local pdf for a given data set, it can be directly used to reflect the density of the training data around the test point. A well-known method of density estimation when the form of the pdf is not known *a priori* is Parzen windows [43]. The Parzen estimator is given by

$$\rho(\mathbf{x}) = 1/K \sum_{k=1}^{K} 1/\sigma^N \cdot \varphi(\mathbf{x} - \boldsymbol{x}_h/\sigma) \tag{5.3}$$

where $\rho(\mathbf{x})$ is called the Parzen density estimate for $\mathbf{x}$, K is the available number of training points, N is the dimension of the input, σ is the width of the hidden unit, $\boldsymbol{x}_h$ is the location of its center and φ is a window function satisfying

$$\int_{-\infty}^{\infty} \varphi(\mathbf{y})d\mathbf{y} = 1 \qquad \varphi(\mathbf{y}) \geq 0 \tag{5.4}$$

It has been shown by Specht [44] that Parzen windows can be implemented by RBFNs. Its direct implementation in network form requires radial units of identical width centered at every data point. However, in the RBFNs considered here,

1) there are fewer units than data points,

2) radial units are not centered at data points, and

3) widths of the activation functions of each hidden node are not identical.

The key idea in quantifying extrapolation is to estimate the data density using the unit centers and widths determined when the RBFN is trained for functional approximation. Towards this end, a two stage approach is used:

1) Probability density functions (pdf) are estimated at each hidden unit center using Parzen windows

2) Based on these pdfs, the probability at arbitrary test points are determined by means of an interpolation formula.

At each hidden unit h, the Parzen density estimate based on the unit width σ_h and unit activation function $a_h(x)$ is

$$\rho_h = \rho\ (x_h) = \ 1/KV_h \sum_{k=1}^{K}\ a_h(\boldsymbol{x}_k) \tag{5.5}$$

where K, as before, is the number of training points and,

$$V_h = \int_{-\infty}^{\infty} a_h(\mathbf{x})d\mathbf{x} \tag{5.6}$$

A simplified model has been proposed where activation functions are sharply defined hyperspheres rather than Gaussian distributions.

$$a_h(\mathbf{x}) = 1 \text{ if } \| \mathbf{x} - \boldsymbol{x}_h \| \leq c.\sigma_h$$

$$= 0 \text{ if } \| \mathbf{x} - \boldsymbol{x}_h \| > c.\sigma_h \tag{5.7}$$

where c is a constant defined as the smallest positive real number that allows the hyperspheres to cover the training data completely. Given this activation function, Equation (5.5) simplifies to,

$$\rho_h = (n_h / K) / V_h \tag{5.8}$$

where V_h is the volume of the hypersphere at node h, n_h is the number of training points falling inside V_h and K is the total number of training points. An important point to note here is that because the hidden units overlap, an arbitrary point $\mathbf{x}$ may be contained in more than one hidden unit and hence may be associated with more than one density estimate. This implies that $\sum_h n_h > K$. Therefore, an estimate of local density for any point can be obtained by averaging.

$$\rho(\mathbf{x}) = \frac{\sum_{h=1}^{H} a_h(x)\rho_h}{\sum_{h=1}^{H} a_h(x)} \tag{5.9}$$

If a point is contained in none of the hyperspheres, both the numerator and the denominator are equal to zero.

Let V_o to be the complement of the space covered by the hypersheres. The activation function for a test point falling in V_o is

$$a_o(\mathbf{x}) = 1 - \max(a_h) \tag{5.10}$$

i.e. a_o=1 if and only if the activation of all hyperspheres is zero. With this definition Equation (5.9) can be rewritten as

$$\rho(\mathbf{x}) = \frac{\sum_{h=0}^{H} a_h(x)\rho_h}{\sum_{h=0}^{H} a_h(x)} = \frac{\sum_{h=1}^{H} a_h(x)\rho_h}{\sum_{h=1}^{H} a_h(x) + 1 - \max(a_h)} \tag{5.11}$$

the right hand side of the equality stemming from the fact that $\rho_o = 0$.

In the case that a_h is a continuous variable as given by Equation (5.1), the Parzen window estimate of density [43] at unit centers using a Gaussian window of width σ_h is

$$\rho_h = \frac{\sum_{k=1}^{K} a_h(x_k)}{K(\prod^{1/2}\sigma_h)^N} \tag{5.12}$$

where all the terms in the above equation are the same as defined earlier. In this case, the effective number of points associated with any hidden unit h is

$$n_h = \sum_{k=1}^{K} a_h(x_k) \tag{5.13}$$

As indicated in Equation (5.11), the density contributions from active hidden units need to be combined to estimate the density at a test point **x**. With continuous activations, Equation (5.11) calculates a weighted average of unit densities, rather than a simple average.

If the test point is close to a unit center, the corresponding activation will be high. If the test point is far away from the unit center, activation and hence its weighted contribution will be small. If the test point is far away from all the units, the complementary activation function will approach unity while the numerator simultaneously approaches zero. The resulting density will consequently approach zero.

In order to interpret $\rho(\mathbf{x})$ as 'sufficient' amount of data, the threshold value is set such that it is equal to the minimum of all ρ_hs of the overall training set. This means that the network is *not* extrapolating if

$$\rho(\mathbf{x}) > \mathbf{T} \text{ where } \mathbf{T} = min\{\ \rho_h\ \} \text{ for } \boldsymbol{h} = 1,2,\ldots.\boldsymbol{H}.$$

provided the confidence interval local to the minimum ρ is acceptable. If not, a higher threshold level corresponding to an acceptable confidence interval is selected.

5.4.2. Confidence Limits

A confidence limit is used to indicate the regions that have a poor local fit to the function that the network is modeling. In this method the model for the accuracy of the fit for a given output in the region of the test point is similar to the confidence limits placed on a random variable. First of all, a local confidence limit for each RBF unit is developed. Then, the confidence limit for model prediction is obtained by taking a weighted average of the confidence limits over all contributing units. If the training points are associated unambiguously with hidden units (crisp case), the local estimate for variance of model residual output i within domain of hidden unit h is

$$s_{hi}^2 = \left[\sum_{j=1}^{n_h} E_{ij}^2\right] \Big/ (n_h - 1) \tag{5.14}$$

where n_h is the number of points inside the hypersphere and E_{ij} is the error on output i for sample j obtained in the training phase. For Gaussian RBF units, this can be generalized to

$$s_{hi}^2 = \left[\sum_{k=1}^{K} a_h(x_k) * E_{ik}^2\right] \Big/ (n_h - 1) \tag{5.15}$$

where $a_h(\mathbf{x}_k)$ is the activation of $\mathbf{x}_k$ in unit h and n_h is as defined by Equation (5.13).

The 95% confidence limit for expected value of the residual associated with output i for unit h is given by

$$CL_{hi} = t_{95} * s_{hi} * n_h^{-0.5} \tag{5.16}$$

where t_{95} is the critical value of Student's t-statistic for 95% confidence and n_h-1 degrees of freedom. Since the effective number of data points in a radial unit n_h is a continuous-valued function, the value of t_{95} is calculated by rounding n_h to the nearest higher integer. The final average confidence limit of the output node i is then calculated using the following equation,

$$CL_i(\mathbf{x}) = \left[\sum_{h=1}^{H} a_h(x) * CL_{hi}\right] \Big/ \left[\sum_{h=1}^{H} a_h(x)\right] \tag{5.17}$$

This is an average of the local confidence limits weighted by the contribution of each hidden unit.

This method assumes the residuals of the model to be independent and normally distributed with a zero mean and constant variance over the neighborhood defined by each hidden unit in the RBFN. This constant variance may vary from unit to unit. However, for this assumption to be accurate, all of the systematic variations of the dependent variable have

to be taken into consideration by the model implying that the model exactly matches the form of the function that is being approximated. Since, the true functional form is unknown and the model is empirical, the assumption of normally distributed errors may not be entirely accurate. Nonetheless, this method provides a good indication of the relative accuracy of the model prediction for the problem under consideration.

In order to be able to calculate the limits of dispersion of the data points, instead of local confidence limit of the expected value of model prediction, the variance of residuals is calculated as before. Assuming the residuals to be normally distributed, confidence limits on the residuals can be set based on the standard normal distribution. However, the true mean or variance of the residuals is not known and only a finite sample is available for their estimation. The mean may be non-zero because of the mismatch between empirical and true model forms. In order to account for this uncertainty variance of the mean and residuals are pooled. In deriving CL_{hi}, the total variance would be s_{hi}^2/n_h (variance of the mean) plus s_{hi}^2. Hence a more accurate estimate CL_{hi} is given by

$$CL_{hi} = t_{95} * s_{hi} * (1 + 1/n_h)^{0.5} \tag{5.18}$$

where all the terms are the same as defined in Equation (5.16).

5.5. Application of the Validity Index Network to Ultrasonic NDE

The problem under consideration is a classification problem. In a normal classification problem, the outputs are class labels. The target output vector to be learned has a value one for the output node corresponding to the correct class and zero for the other output nodes. Using this formulation, if the network has enough representational capacity and the training data are dense enough, the network output will represent the local relative

probability densities of the classes. The classification is given by the output node with the highest value.

The accuracy of the network outputs is largely dependent on the training data. In regions of sparse training data, variance in the network outputs increases and hence, the uncertainty increases. The reliability of classification can therefore be reflected by the density of training data in the vicinity of test data, by the extrapolation flag. Extrapolation measures are useful in such cases, but the corresponding measurements of uncertainty are not. This is due to the fact that in functional approximation problems the training targets represent the desired outputs and the residual errors represent the lack of fit.

5.6. Implementation and Results

In the NDE classification problem, given an input signal, the network classifies it as belonging to one of three classes - namely cracks, counterbores and rootwelds. For a network, this translates to three output nodes, each one representing one class. The training targets are, thus, an encoding of a symbolic variable while the desired network outputs are the pdfs. By minimizing the error during training, the classifier is made to 'learn' the pdfs. If the relative probabilities of membership of each class for each training sample were known, the problem could be reduced to one of functional approximation. However, this information is precisely the information that is unknown.

Certain modifications to the algorithm are necessary before this method can be applied to the classification problem. First, the classification problem needs to be reduced to that of functional approximation. To achieve this, class labels at the outputs are replaced by degrees of membership of each training sample to its corresponding class. Membership

values are obtained by calculating the weighted distance of each training sample in the feature space from the mean of that class scaled by the variance of the feature. Mathematically,

$$z_{ik} = \sqrt{\sum_{j=1}^{n}\left[\frac{F_{ij}-o_{kj}}{v_{kj}}\right]^2} \tag{5.19}$$

where z_{ik} is the weighted distance of the pattern $\boldsymbol{F}$ (pulse_width, peak_value)$_i$ to the center of the kth class, $\boldsymbol{O}_k$ is the mean of all the training vectors for class $\boldsymbol{k}$, and $\boldsymbol{V}_k$ is the standard deviation of the features in the kth class. The output membership value, u_{ik}, for k^{th} input vector to class i is then given by

$$u_{ik} = 1/\, z_{ik} \tag{5.20}$$

This is very similar to the Mahalanobis distance metric and in fact is the same metric used to assign output membership values in the implementation of the *'fuzzy'* MLP in Chapter 4.

Once the inputs and the outputs are obtained, the adaptive K-means algorithm (Section 2.3.3.2) can be used to obtain an optimal number of hidden nodes for the RBF. For the simulation carried out, the optimum number of hidden nodes was found to be 8. The width was taken as the RMS distance to P nearest cluster centers given by Equation (5.19). Let *dist*(**x** ,**y**) be the Euclidean distance between **x** and **y** in an n-dimensional space defined in Equation (2.15). In this same domain, let $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots\ldots \boldsymbol{x_p}$ be the P nearest cluster centers to $\boldsymbol{x}$. Then,

$$\sigma = \sqrt{\frac{\left(dist\{x,x_1\}\right)^2 + \left(dist\{x,x_2\}\right)^2 + \ldots . \left(dist\{x,x_p\}\right)^2}{P}} \tag{5.21}$$

The least error was obtained for P=2. After training the RBFN, the local densities associated with each hidden unit, ρ_h was calculated using Equation (5.12). The minimum density associated with the hidden nodes was 0.109 x 10^{-13}. The values for density have all been divided by 10^{-13} for normalization. To evaluate the local estimate for variance of the residual error on training data, Equation (5.15) was used and the result is substituted in Equation (5.16) to obtain confidence limits on the training data for each hidden node during training. The confidence limits for the test data are then obtained using Equation (5.17). Let O_1, O_2 and O_3 be the three output nodes of the RBFN. Let $\mathbf{CL_1}$, $\mathbf{CL_2}$ and $\mathbf{CL_3}$ be the confidence limits estimated by this method. If the intervals $O_1 \pm \mathbf{CL_1}$, $O_2 \pm \mathbf{CL_2}$, and $O_3 \pm \mathbf{CL_3}$ have any overlapping regions, the decision of the RBFN is said to be ambiguous.

The results obtained using this technique are shown in Table 5.1.

The misclassifications are marked by a '*' in the last column. In each of the Figures 5.2, 5.3 and 5.4, the training signals are represented by 'c', 'b' and 'r' for the crack, counterbore and rootweld class respectively. The test points are shown by a *.

As shown in Figure 5.2, points 5 and 10 occur in areas of low density of training data. This is reflected by the corresponding density flags having values lower than the minimum density of training data associated with the hidden nodes. The confidence intervals for outputs corresponding to point 5 are large and result in an overlap reflecting ambiguity in the classification. This, however, does not happen in the case of point 10.

Table 5.1: Results of the RBFN network

Index no.	Output from RBFN at node1 counterbores	node2 cracks	node3 rootwelds	*density flag* ρ_{min} = 0.109	Confidence limit at node1	node2	node3	
1	0.0000	1.0000	0.0000	0.4113	0.0002	0.0000	0.0100	
2	0.0000	1.0000	0.0000	0.3973	0.0102	0.0004	0.0023	
3	0.0000	1.0000	0.0000	0.2156	0.0219	0.0020	0.0111	
4	0.0000	1.0000	0.0000	0.2323	0.0010	0.0041	0.0023	
5	0.0021	0.1618	0.5432	**0.0123**	0.0013	0.3429	0.2351	*5
6	0.0141	0.9980	0.0000	0.1217	0.0043	0.0009	0.0564	
7	0.0000	0.9900	0.0000	0.3324	0.0006	0.0496	0.0045	
8	0.0000	0.8611	0.0000	0.3136	0.0345	0.0765	0.1858	
9	0.0000	1.0000	0.0000	0.2973	0.0384	0.0675	0.0089	
10	0.8573	0.0322	0.0059	**0.0001**	0.2945	0.0036	0.0575	*10
11	1.0000	0.0000	0.0000	0.4411	0.0004	0.0045	0.0047	
12	1.0000	0.0000	0.0000	0.3534	0.0013	0.0076	0.0056	
13	0.8586	0.0100	0.0001	0.2115	0.0687	0.0574	0.0253	
14	0.4900	0.5871	0.1300	**0.1109**	0.1000	0.2874	0.0039	*14
15	0.9998	0.0000	0.0000	0.2215	0.0376	0.0500	0.0520	
16	1.0000	0.0000	0.0000	0.2737	0.0385	0.0002	0.0000	
17	1.0000	0.0000	0.0000	0.2218	0.0024	0.0076	0.0200	
18	0.9020	0.0000	0.0000	0.1435	0.1089	0.0013	0.2006	
19	0.7059	0.1515	0.0076	**0.1976**	0.2545	0.3985	0.1008	*19
20	1.0000	0.2188	0.0000	0.2034	0.0000	0.1312	0.0013	
21	0.1787	0.1000	1.0000	0.3199	0.0987	0.0012	0.0453	
22	0.1518	0.1237	0.1247	**0.0000**	0.1022	0.3176	0.2997	*22
23	0.2857	0.0011	0.1000	**0.0105**	0.0000	0.0006	0.0001	*23
24	0.0000	0.0010	0.9795	0.3976	0.0001	0.0012	0.0043	
25	0.5769	0.0108	0.9995	0.1238	0.0005	0.0967	0.0007	
26	0.0678	0.0000	0.7845	0.1295	0.1000	0.1996	0.0200	
27	0.5847	0.0000	0.7800	0.1124	0.0001	0.0195	0.1010	
28	0.3792	0.0001	0.4099	**0.1048**	0.1184	0.0001	0.1099	#28
29	0.1024	0.0000	0.9496	0.1131	0.0124	0.1843	0.0078	
30	0.0000	0.0000	1.0000	0.1367	0.0073	0.0009	0.0012	

'*' represents misclassified signals

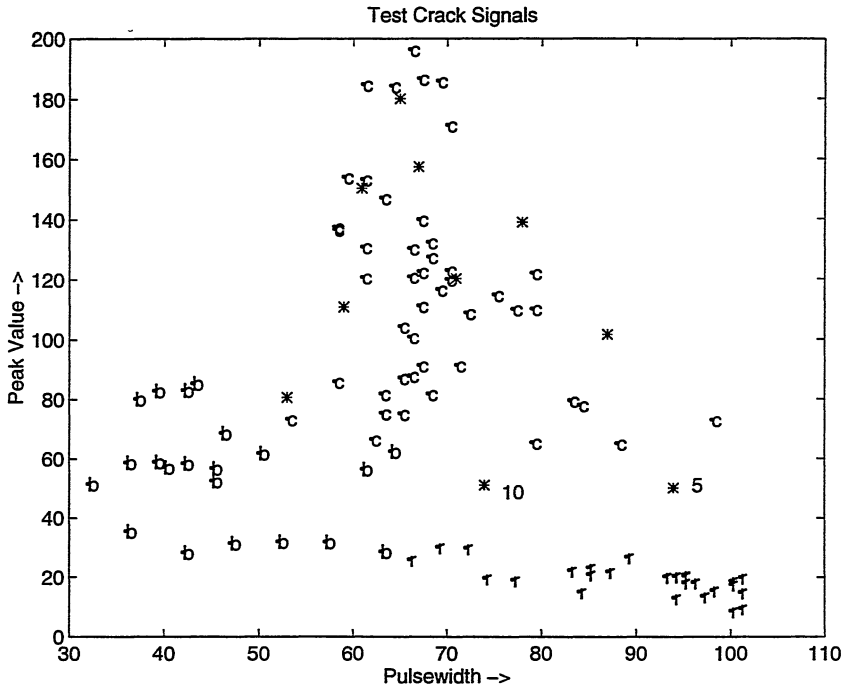

Figure 5.2 Test Signals for Cracks

Points 14 and 19, shown in Figure 5.3, have sufficient training data in their vicinity as indicated by their respective density flags. The corresponding confidence limits, when applied to the output nodes, indicate ambiguity of the test points as required. Points 22 and 23 shown in Figure 5.4, have very low activation values at all three outputs. This can be explained as due to the fact that the points occur in a region devoid of any training data. As such, since there is not sufficient density of training points in this region, the confidence limits have no meaning. Point 28 is an example of a case where the classification rule gives the right decision. However, the density flag indicates an area of low training data density reflecting poor reliability of the neural network decision.

5.7. Conclusion

Thus, we see from the implementation of the validity-index network, the confidence limits can be applied to the output nodes of an RBFN to indicate ambiguity in the classification decision. In addition, the extrapolation flag gives added information to the user regarding the density of the training data in the vicinity of the test point. Thus, in addition to classifying the input signals, this extended RBFN gives additional information regarding the quantity and the quality of the training data.

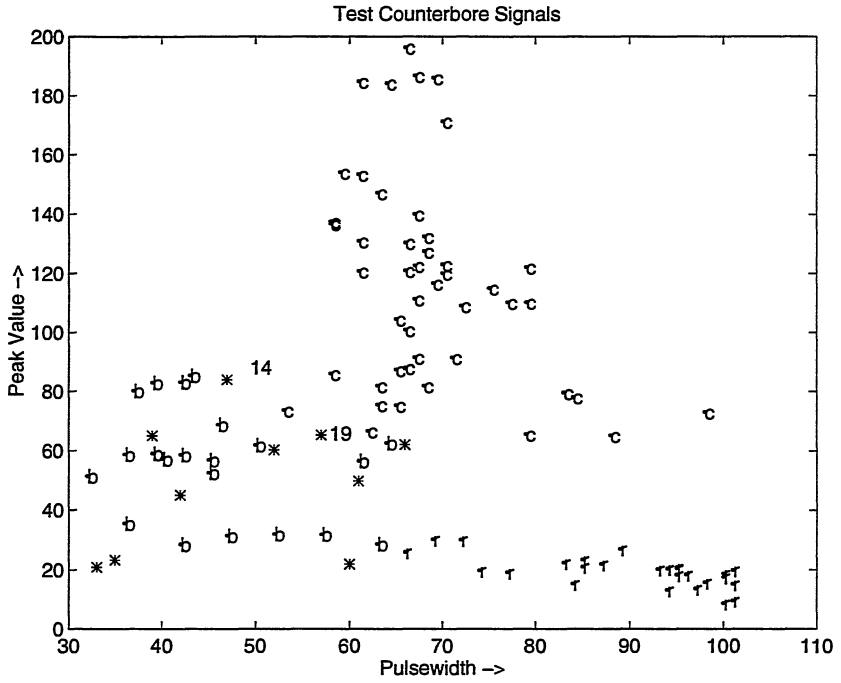

Figure 5.3 Test Signals for Counterbores

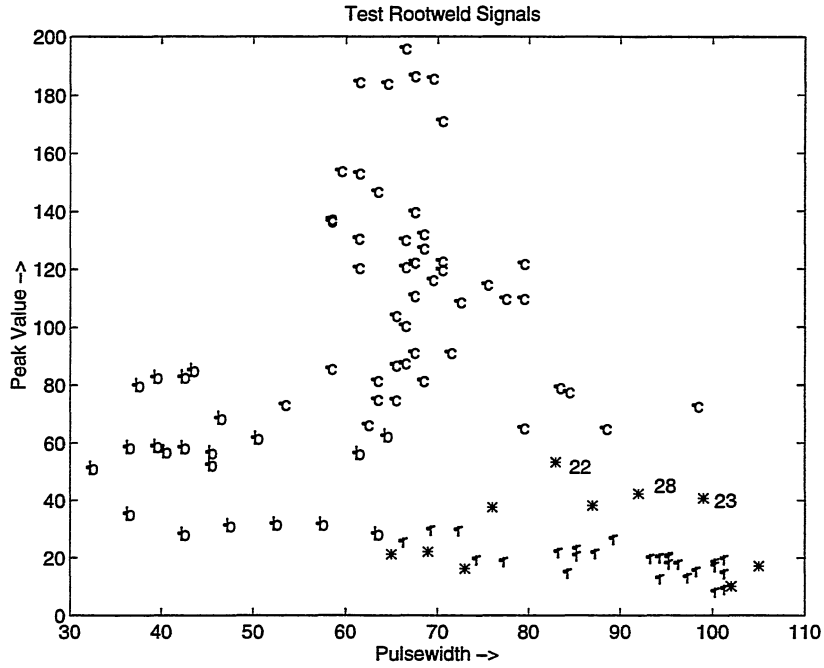

Figure 5.4 Test Signals for Rootwelds

6. A NEW METHOD FOR COMPUTING RELIABILITY OF ANN CLASSIFICATION

6.1. Motivation

Automated signal classification systems are becoming increasingly popular in many applications. This is especially the case in industries such as aerospace, nuclear power, etc., where large volumes of data are analyzed and the analysis is greatly influenced by operator fatigue.

In all these applications automated signal classification systems must perform with a high level of confidence. In general, a high classification accuracy can be attained at the cost of false positive (FP) signals as seen in Chapter 3. However this is not feasible in practice since a false alarm would imply unjustified plant shutdown, resulting in significant financial repercussions. Statistical techniques, involving estimation of POD curves, reflect the uncertainty in data acquisition processes. However, the POD plots as a function of the flaw size do not reflect the confidence of the classifier.

The existing method for computing reliability of a classification decision, described earlier in Chapter 3 (Section 3.3), involves the use of a secondary network to make predictions of error bounds on the outputs of the primary network. However, the use of a second ANN to predict the performance of the first ANN is not convincing, since this raises the issue of the error and confidence associated with the second network. In this chapter, a new method is described for quantifying the reliability of a neural network based signal classification system. The validity-index network, described in Chapter 5, is a reasonable

approach but its application is restricted to Radial Basis Function Networks (RBFNs). In this thesis, new approach which overcomes both the above mentioned problems has been proposed. First, the method does not involve the use of a second neural network to compute the reliability and second, the approach is independent of the type of neural network employed as a classifier. The approach is fairly general and is applicable to MLP and RBFN networks or other automated signal classification systems.

6.2. Description of the New Method

In any ANN based signal classification system, there are several factors that contribute to the lack of reliability of the network decision. The most significant factors are the quality of the training data and the accuracy of the training process. If these two factors are quantified, one can formulate an expression for calculating the error bound on the network output which in turn can be mapped onto the reliability of the decision. The quality of the training data is said to be poor when the training points are not diverse enough and they do not cover the entire input domain. Accuracy of the training process is related to the convergence of the network. Most of the training algorithms, especially those based on gradient descent methods, result in convergence of the ANN to a local minimum as against a global minimum of the error surface. This may greatly affect the reliability of the classification performance.

The new method, proposed here, attempts to quantify these sources of uncertainty in a neural network decision. In the implementation of this method, the training data is first partitioned into two sets for training and validation purposes. The training set is used for estimating the learning error of the network. This is followed in the second step by the

development of a regression model which relates the error in classification to the training error. This model also predicts the error bound on the output of the network based on the error measures computed. The details of the procedure are described below.

Notations used:

L	Entire training set
L_1	Training data
L_2	Validation data
N_1	Number of elements in L_1
N_2	Number of elements in L_2
G	Neural network
terr	Training error
derr	Input error
cerr	Classification error
x	Test data
y	Test output
d	Desired output
d_1	Distance from the nearest neighbor
d_2	Distance from the next nearest neighbor
δ	Density flag
x_n	Nearest input to x from training data
y_n	Training output from network **G** for input x_n

The training set $\boldsymbol{L}$ is partitioned into two sections $\boldsymbol{L_1}$ and $\boldsymbol{L_2}$ such that both the sections contain a uniform distribution of points from all the classes. Let $\boldsymbol{L_1}$ and $\boldsymbol{L_2}$ have $\boldsymbol{N_1}$ and $\boldsymbol{N_2}$ elements respectively. The classification network **G** first is trained using the signals in set $\boldsymbol{L_1}$. During validation with the signals in $\boldsymbol{L_2}$, three different types of errors are defined namely classification error - error at the output for test data, training error - error at the output for training data, and data error - error due to mismatch between test and training data at the input. Consider a test sample $\boldsymbol{x}$ in $\boldsymbol{L_2}$. Since the network was not trained with elements from $\boldsymbol{L_2}$, there will be some deviation of the output from the desired output. Let $\boldsymbol{y}$ be the output of **G** to the input $\boldsymbol{x}$. i.e.

$$\boldsymbol{y} = \mathbf{G}\{\boldsymbol{x}\} \tag{6.1}$$

Let the desired output corresponding to $\boldsymbol{x}$ be $\boldsymbol{d}$. We can define the output classification error to be

$$\boldsymbol{cerr} = dist(\mathbf{y}, \boldsymbol{d}) \tag{6.2}$$

where *dist* is the Euclidean distance as defined in Equation (2.15).

In order to correlate the classification error to the input data error, we find the element in $\boldsymbol{L_1}$, closest to $\boldsymbol{x}$. Let

$$\boldsymbol{derr} = \min_{x_i \in L_1} \{dist(x_i, x)\} \tag{6.3}$$

Let the minimum occur for $i=n$. This implies the nearest neighbor to $\boldsymbol{x}$ in the training set $\boldsymbol{L_1}$ is the data point $\boldsymbol{x_n}$. Let the output of **G** to the training data $\boldsymbol{x_n}$ be $\boldsymbol{y_n}$, i.e.

$$\boldsymbol{y_n} = \mathbf{G}\{\boldsymbol{x_n}\} \tag{6.4}$$

The training error for the point $\boldsymbol{x_n}$ is then given by

$$terr = dist(d_n, y_n) \tag{6.5}$$

Here ***derr*** is the distance of test pattern x from its nearest neighbor in L_1 and ***terr*** is the training error of the nearest training data. To quantify the density of the training data, the mean distance of x from the two nearest neighbors obtained. Let d_1 and d_2 be the distances from the two closest points. The density flag δ is given by

$$\delta = \frac{d_1 + d_2}{2} \tag{6.6}$$

As δ decreases, it indicates that the density of training data in the region of the test point is higher. This term can, therefore, be used as an indication of the density of the training data in the vicinity of the incoming test point. The average of all δ's for all test samples in the validation data is taken as the lower limit for the density threshold, T, i.e.

$$T = \frac{1}{N_2}\sum_{i=1}^{N_2}\delta_i \tag{6.7}$$

When the δ of a test point is greater than T, the point is interpreted as occurring in a region of relatively low training data density which in turn implies lower reliability of classification. On the contrary, if this distance is less that T, training data density and therefore the reliability is higher.

The distributions of ***derr*** and ***terr*** are plotted with respect to the output classification error, ***cerr***, of the test point. These plots can be used directly or after suitable transformation to develop a regression model.

A model that relates the classification error (***cerr***) to the training error (***terr***) and training diversity (***derr***) can be expressed as,

$$cerr = f(derr, terr) \tag{6.8}$$

For example, if the distribution of ***terr*** versus ***cerr*** is exponential, the plot of the natural logarithm of ***terr*** against the natural logarithm of ***cerr*** will be linear. A linear regression model can then be used to simplify further computation. In the simple case of a linear regression model,

$$f(derr, terr) = \beta_1\, derr + \beta_2 terr + c \tag{6.9}$$

where β_1 and β_2 are constants and c is an intercept.

Consider a test point x. Let the network produce an output

$$y = \mathbf{G}\{x\} \tag{6.10}$$

Let $x_n \in L$ be the nearest neighbor of x and

$$y_n = \mathbf{G}\{x_n\} \tag{6.11}$$

The input error is defined by

$$derr(x) = \{dist(x_n, x)\} \tag{6.12}$$

Let the training error of x_n be given by,

$$terr(x) = dist(y_n, d) \tag{6.13}$$

where d is the desired output for x_n.

The classification error for the test point x is obtained by substituting Equations (6.12) and (6.13) in Equation (6.8).

New test inputs presented to a network may or may not occur in regions of dense training data. Thus, the closest training data to the test point may or may not be from the same class as the one to which the signal is assigned by the network **G**. For example, consider a simple two class problem. Let x be assigned to class I. Let M_1 be the set of points

in L belonging to class I. Let $x_{sameclass}$ be the nearest neighbor of x in M_I. Also, let the nearest neighbor of x in the full set L be $x_{nearest}$. Then,

$$dist_{sameclass} = dist(x_{sameclass}, x) \tag{6.14}$$

and $$dist_{nearest} = dist(x_{nearest}, x) \tag{6.15}$$

Let γ be a ratio defined as

$$\gamma = dist_{sameclass} \,/\, dist_{nearest} \tag{6.16}$$

where $\gamma = 1$ if the nearest point belongs to class I and $\gamma > 1$ if the nearest point belongs to another class. In general, it would be expected that the value of γ be 1 in all cases. However, because of the arbitrary nature of the boundary placement by an MLP during training, test points lying very close to the boundary may have γ greater that 1.

In addition to the error, ***cerr***, obtained from the regression model, the ratio γ is also an important measure which needs to be taken into consideration. A function η quantifies this information and the error estimated by the regression model to get the final error, E.

$$E = \eta(\gamma, cerr) \tag{6.17}$$

6.3. Implementation and Results

The training data used here, is the same as that used in the previous implementations. The original ultrasonic signals consist of A-scans which are 1800 points long. For simplicity, two features namely, pulse width and peak value of the A-scan data are extracted. This reduces the dimension of the input signal from 1800 to 2. The two-dimensional feature vectors form the inputs for training the MLP. A total of 93 signals were used in the training

data set (46 cracks, 21 counterbores and 26 rootwelds). The test data set comprised 30 signals (10 cracks, 10 counterbores and 10 rootwelds).

The step by step procedure in the implementation of the new technique for determining the error estimate associated with the classification decision are as follows:

1. The training data of 93 elements was partitioned into two sections such that L_2 had 30 elements (10 cracks, 10 counterbores and 10 rootwelds) and L_1 had 63 elements.
2. The MLP consisted of 10 hidden nodes in its hidden layer, 2 input nodes and 3 output nodes.
3. This network was trained on the signals in L_1 using the backpropagation algorithm.
4. Elements in L_2 were then used for validation. Let

   $y_i = \mathbf{G}\{x_i\}$, for $x_i \in L_2$, $i=1,2,...30$.
5. The error $cerr_i$ was calculated using Equation (6.2) for each validation point (30 in this case).

   $cerr_i = dist(g_i, y_i)$ for $i=1,2...30$.
6. For each element x_i in L_2, the nearest point x_{ni} in L_1 was located and the distance between the two was calculated using Equation (6.3).

   $derr(i) = dist(x_{ni}, x_i)$ for $i=1,2...30$.

   The second closest point was similarly located and density flag δ_i was calculated using Equation (6.6). Equation (6.7) was used to evaluate density threshold T.
7. Let $y_{ni} = \mathbf{G}\{x_{ni}\}$.

   For the nearest point in L_1, the training error using Equation (6.5) was also calculated.

   $terr(i) = dist(d_i, y_{ni})$ for $i=1,2...30$.

8. The scatter plots of ***derr*** versus ***cerr*** and ***terr*** versus ***cerr*** are shown in Figures 6.1 and 6.2 respectively. It was found that the distance of the validation points from their closest neighbor in the training set (***derr***) was not a good parameter for determining the test error ***cerr***. As seen in Figure 6.1 there is no correlation between these two quantities. This may be due to the fact that, in case of an MLP, the assignment of a class label to a test point does not depend as much on the distance of this point from the training samples as it does on the distance from the class boundaries that the MLP '*learns*' during the training process. The farther the test point is from the boundary, the more confident is the classification. In fact, a test point may be equally far from both the training set and the boundary and yet be correctly classified. Hence, for an MLP the test error is not related to the distance from the nearest training point. However, this is not necessarily true for other types of networks.

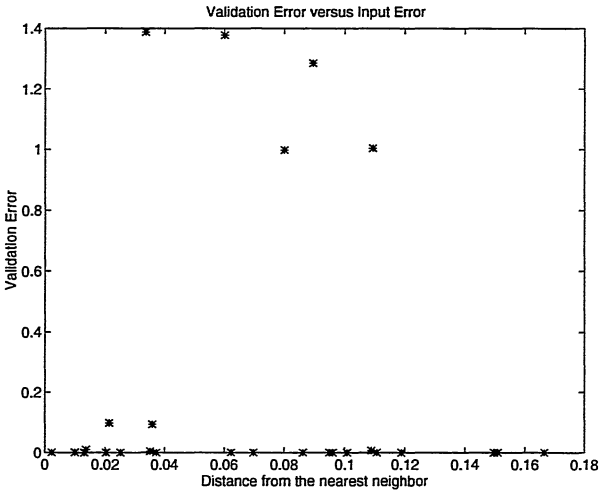

Figure 6.1 Plot of Validation Error versus Input Error

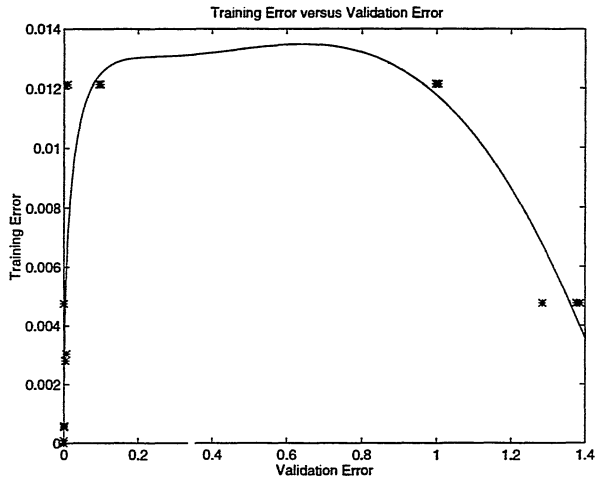

Figure 6.2 Plot of Training Error versus Validation Error

For example, class assignment in an RBFN involves a clustering process. In this case the distance of the test point from its nearest neighbor will definitely have a relationship with the test error. Since we are considering an MLP for this implementation, only ***terr*** was considered in this case.

9. Two approaches were taken in order to fit a function between ***terr*** and ***cerr***. The first approach was a simple one in which a logarithmic transformation was applied to both ***terr*** and ***cerr***. The points in the transformed domain are shown in Figure 6.3. A linear regression model was then fitted to this data. The second approach involved fitting the best possible function through the data shown in Figure 6.2. This function is shown as a solid line in Figure 6.2. The linear model that was obtained had the form

$$\ln cerr = 0.8634 \ln terr \tag{6.18}$$

The higher order function that best fit the data in Figure 6.2 (obtained using *TableCurve*^{TM} software) was of the form

$$terr = a + b * cerr + c * cerr^2 * ln(cerr) + d * cerr^{2.5} + e * cerr^{0.5} \quad (6.19)$$

where a=0.0001947178, b=-0.17850432, c=-0.13924689,d=0.10582571, e=0.084246128.

10. The test data was then used to evaluate the performance. In addition to the error predicted by the regression model (*cerr*), the ratio γ from Equation (6.13) was calculated. To calculate the final error using the results from the linear regression model, the following empirical relationship was used:

$$E_{lin} = 1.5 * \gamma^2 * cerr \quad (6.20)$$

Similarly, the overall error while using the higher order model was given by

$$E_{nonlin} = 3 * \gamma^2 * cerr \quad (6.21)$$

11. The mean distance from two nearest neighbors (δ) was also recorded.

These results are recorded in Table 6.1. The misclassified signals have been denoted using a '*' in the last column. The first column of the table is an index number for the test sample. The next three columns represent the outputs at the three output nodes of the MLP.

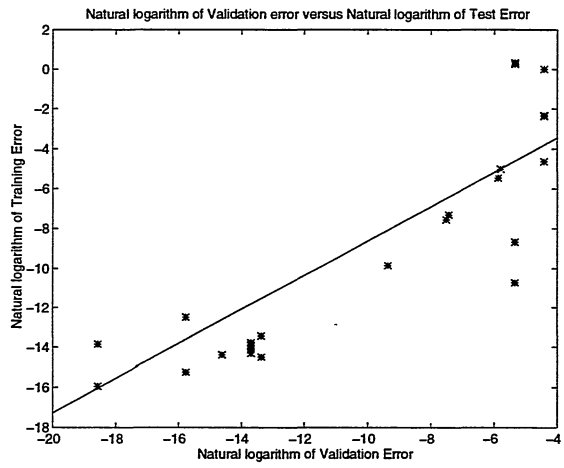

Figure 6.3 Plot of *ln*(Training Error) versus *ln*(Validation Error)

Table 6.1: Results of classification and reliability analysis parameters

No	Outputs O_1 node for counter-bores (1)	O_2 node for cracks (2)	O_3 node for rootwelds (3)	*density flag* δ T=0.085 (4)	γ (5)	*Linear regression* *cerr* (6)	*E* (7)	*Higher order function* *cerr* (8)	*E* (9)	
1	0.0000	1.0000	0.0000	0.0764	1.0000	0.0000	0.0000	0.0003	0.0008	
2	1.0000	0.0000	0.0000	0.0284	1.0000	0.0000	0.0000	0.0003	0.0008	
3	0.0000	1.0000	0.0000	0.1022	1.0000	0.0000	0.0000	0.0003	0.0010	
4	0.0000	1.0000	0.0000	0.1025	1.0000	0.0000	0.0000	0.0004	0.0011	
5	0.0001	0.0316	0.6432	0.1385	**1.6020**	0.0067	0.0259	0.0054	0.0414	*
6	0.0141	0.9980	0.0000	0.0656	1.0000	0.0000	0.0000	0.0003	0.0009	
7	0.0000	1.0000	0.0000	0.0546	1.0000	0.0000	0.0000	0.0003	0.0008	
8	0.0000	1.0000	0.0000	0.0146	1.0000	0.0000	0.0000	0.0003	0.0009	
9	0.0000	1.0000	0.0000	0.0631	1.0000	0.0000	0.0000	0.0003	0.0009	
10	0.9541	0.0002	0.0029	0.1125	**1.2230**	0.0011	0.0025	0.0019	0.0085	*
11	1.0000	0.0000	0.0000	0.0908	1.0000	0.0000	0.0000	0.0002	0.0006	
12	1.0000	0.0000	0.0000	0.0414	1.0000	0.0001	0.0002	0.0006	0.0019	
13	0.8082	0.1100	0.0000	0.0434	1.0000	0.0964	0.1446	0.0323	0.0968	
14	0.4545	0.6871	0.0000	0.0997	**4.0911**	0.0387	0.9727	0.0169	0.8475	*
15	0.9998	0.0000	0.0000	0.0613	1.0000	0.0000	0.0000	0.0002	0.0007	
16	1.0000	0.0000	0.0000	0.0213	1.0000	0.0000	0.0000	0.0002	0.0006	
17	1.0000	0.0000	0.0086	0.0801	1.0000	0.0099	0.0148	0.0068	0.0205	
18	1.0000	0.0000	0.0000	0.0793	1.0000	0.0000	0.0000	0.0002	0.0006	
19	0.8621	0.0915	0.0000	0.0800	**1.2454**	0.1000	0.4461	0.0331	0.1643	
20	1.0000	0.0000	0.0000	0.0829	1.0000	0.0000	0.0000	0.0002	0.0006	
21	1.0000	0.0000	0.0787	0.0312	**1.0000**	0.0099	0.0148	0.0068	0.0205	*
22	0.0510	0.0239	0.0262	0.0975	**2.3989**	0.0011	0.0095	0.0019	0.0327	*
23	0.0000	0.0011	0.9874	0.0915	1.0000	0.0015	0.0023	0.0023	0.0068	
24	0.0000	0.0000	0.9998	0.0528	1.0000	0.0006	0.0008	0.0013	0.0040	
25	0.9296	0.0000	0.8735	0.0346	**1.0048**	0.0222	0.0336	0.0116	0.0350	*
26	0.0059	0.0000	0.8662	0.0791	1.0000	0.0063	0.0094	0.0052	0.0155	
27	1.0000	0.0000	0.2847	0.0512	**1.0000**	0.0099	0.0148	0.0068	0.0205	*
28	0.0003	0.0008	0.9024	0.0891	1.0000	0.0015	0.0023	0.0023	0.0068	
29	0.9702	0.0000	0.1479	0.0873	**1.0588**	0.0222	0.0373	0.0116	0.0389	*
30	0.0000	0.0000	0.9999	0.0146	1.0000	0.0004	0.0006	0.0011	0.0034	

Numbers in parentheses indicate column numbers.
'*' represents misclassified signals

The output nodes correspond to the classes counterbores, cracks and rootwelds respectively. For example, when the signal is classified as a counterbore, the output from the MLP is expected to be 1-0-0 or as close to it as possible. The signal is said to be assigned to that class whose node has the highest activation. The density flag (δ) is shown in column

four. This is indicative of the density of the training data in the neighborhood of the test point. The fifth column in the table represents the γ ratio as defined in Equation (6.15). This column provides some interesting insights into the operation of the algorithm. The data points for which $\gamma>1$ are highlighted. The ratio γ indicates that the nearest neighbor of the test data lies in a class different from the one that it is assigned to. The sixth and the seventh columns represent the classification error estimated using the linear regression model (Equation (6.18)) and the final error (Equation (6.20)).

The eighth and the ninth columns represent results using the higher order function using Equations (6.19) and (6.21). Let O_i represent the value at the i^{th} output node. The classification error E is an error bound on outputs of the MLP. This means that the value of the output node can be anywhere in the range $O_i \pm E$. In this case, there are three output nodes O_1, O_2, and O_3. If $O_1 \pm E$, $O_2 \pm E$ and/or $O_3 \pm E$ have any intersecting regions, it indicates that the classification decision is ambiguous.

6.4. Discussion

A lot of information can be obtained from Table 5 as discussed below. The most important feature of the table is that the misclassifications have been identified automatically by the algorithm. This result is discussed in detail for three classes (cracks, counterbores and rootwelds) of signals.

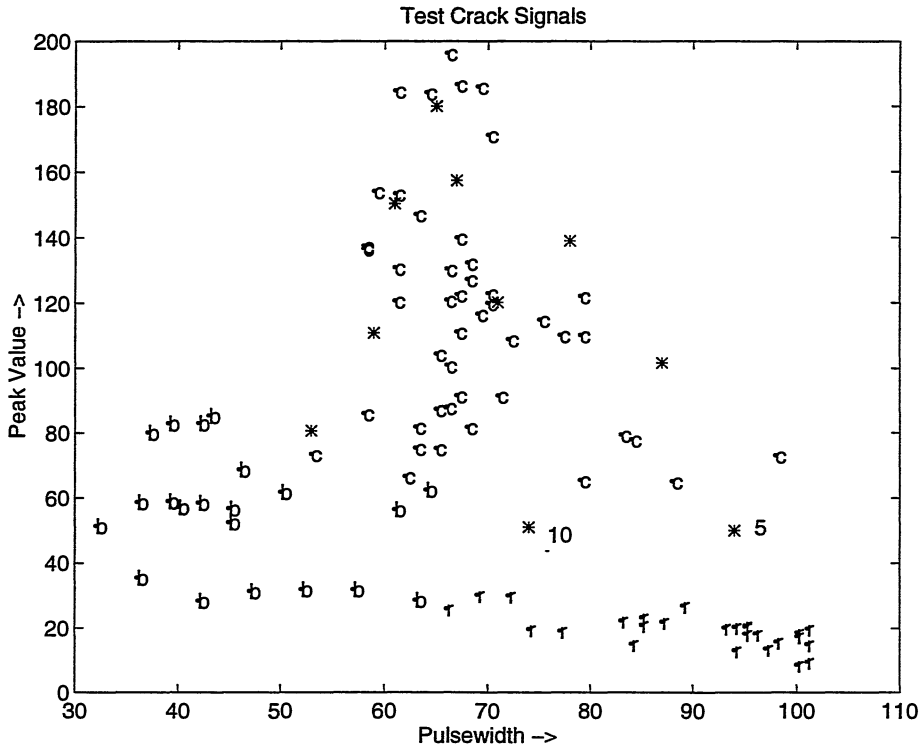

Figure 6.4 Test Signals for cracks

6.4.1. Cracks

The first ten signals in the table are crack signals. The two dimensional plot of these test points is shown in Figure 6.4. The test signals are denoted by a *. The training points are represented by the letters 'c', 'b' and 'r' for cracks, counterbores and rootwelds respectively. Two out of these ten signals have been misclassified. The misclassified points (5 and 10) have also been marked. For the correctly classified signals, the $\boldsymbol{E}$ value from the table shows that there is no ambiguity in the classification decision. Also, for these signals γ=1. For the misclassified points, $\boldsymbol{E}$ does not reflect any ambiguity. However, it is observed that γ>1 for both these points. The value of the corresponding γ's indicate that their nearest neighbors do

not belong to the class that they have been assigned. The density flag also indicates low training data density in the region of both the points 5 and 10. This would reflect the fact that during the training process of the MLP, the training boundary was placed in this region in such a way that both these points lie very close to the training boundary but on the wrong side.

6.4.2. Counterbores

In Figure 6.5, the test signals for counterbores are marked with a '*'. All signals except signal 14 are correctly classified as indicated by the corresponding error bounds and value of γ.

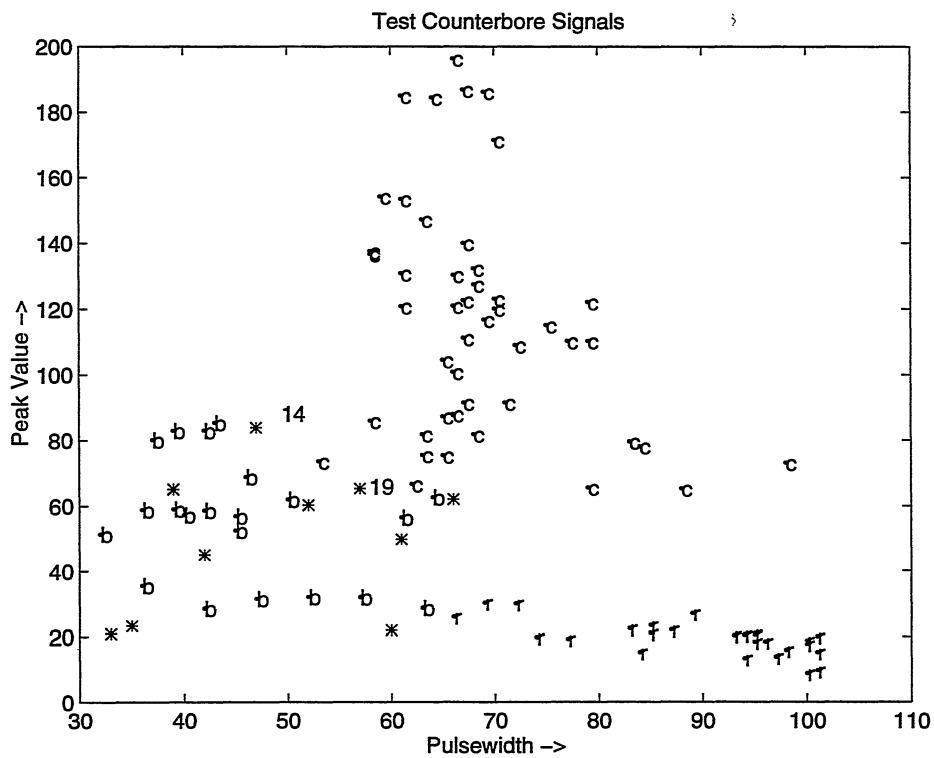

Figure 6.5 Test Signals for counterbores

Signal 14 has been wrongly assigned to the crack class. The nearest neighbor to this point (shown in Figure 6.5) is a counterbore, however. This is reflected in the large value for γ. When the error E is incorporated at the output nodes ($O_{i,14} \pm E$), there is a significant overlap in the allowable range of values that $O_{1,14}$ (node representing counterbore class) and $O_{2,14}$ (node representing crack class).

This overlap is indicative of an ambiguous decision. An interesting point to note is that signal #19 has been correctly classified as a counterbore signal with high confidence but the corresponding γ values is slightly higher than 1. In Figure 6.5, it can be seen that this point lies very close to the boundary between cracks and counterbores. It has a crack signal which is very slightly closer to it than the nearest counterbore signal which accounts for $\gamma>1$. In the case of all ten counterbore test points the density flag indicates sufficient training data in the neighborhood.

6.4.3. Rootwelds

Figure 6.6 shows the test rootweld signals denoted by a '*'. Points 22 and 25 have been misclassified but the error bound E indicates an ambiguous decision in each case. The density flag has a value greater than T, indicating insufficient data around point 22. Point 25, however, does not have this problem. Point 29 has been misclassified with high confidence. The value of γ, however, is greater than 1 and $\delta>T$. Points 21 and 27 from Table 5 are shown in Figure 6.6. They have been misclassified as counterbores. Note that they have a $\gamma=1$ as well as values of $\delta<T$.

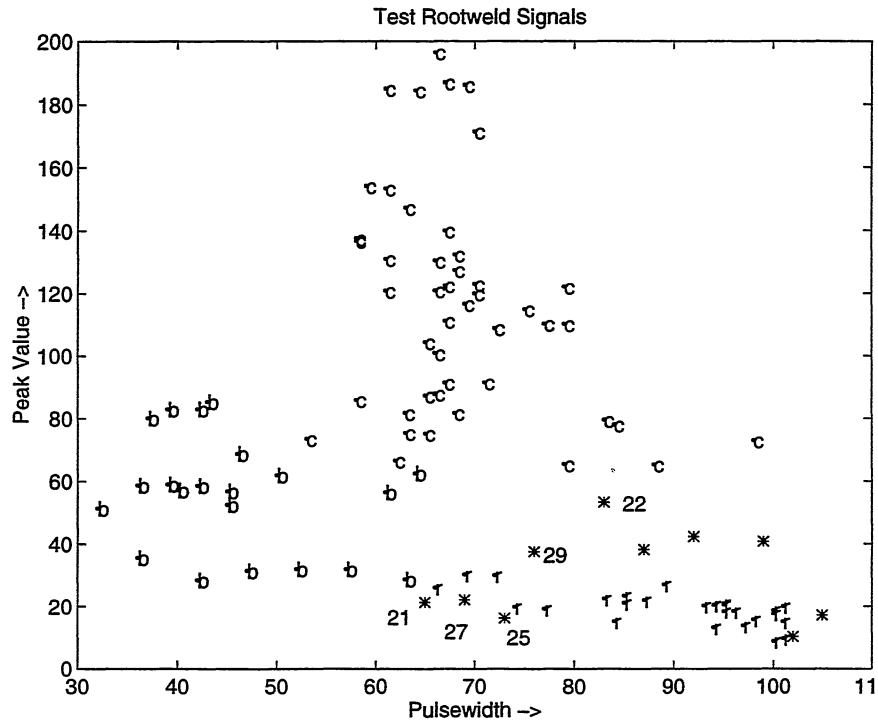

Figure 6.6 Test Signals for rootwelds

This indicates that for the training data used, the network identifies these test points as the counterbore class with sufficient confidence since neither the error term nor γ indicate any uncertainty. Also, there is sufficient training data close to these points. This means that, for features that were extracted from the available data bank, these test points lie in a region defined for another class which is different from its own. In this new technique, however, such points will go unnoticed.

When training data is unevenly scattered, there are empty regions in the input domain where the position of the decision boundaries is not known exactly. Whenever a test point occurs in this region, the network is extrapolating the decision is not reliable. Thus, whenever the value of $\gamma > 1$ and $\delta > T$, the corresponding input signals need further investigation.

7. CONCLUSION

A complete defect classification system has two major sources of uncertainty - one source is associated with the data acquisition system which can be quantified by experimental POD curves. The other source of uncertainty is associated with the classifier. The new technique attempts to quantify the latter case.

Error estimation for an ANN is important because, in addition to improving classification accuracy, the confidence in the classification obtained from an automated signal classification system is essential to flaw detection in a practical implementation.

The issue of reliability of neural networks in an ultrasonic NDE problem has been addressed in this thesis. Several existing approaches have been discussed and two new approaches have been proposed.

The stacked generalization approach as proposed by Wolpert in [34] was first implemented on the ultrasonic weld inspection data. Several problems were encountered. The training error for the secondary network was quite high. Secondly, the error values predicted by the second network was not able to reflect the ambiguity in the test points for the problem under consideration. Besides, the use of an additional ANN for reliability analysis of the primary ANN classifier brought up questions of the reliability of the second network.

The RBF validity index network was considered next with suitable modifications in the original method. The approach performed reasonably well for the ultrasonic NDE classification problem. Areas of sparse training data were flagged, thus, giving more

information to the operator about the reliability of the network decision. However, this method is applicable only to an RBFN and cannot be extended to other types of ANNs.

Two new approaches are suggested to overcome the disadvantages of the above methods. The first approach is based on *fuzzy* set theory for determining the confidence in a network decision. This approach required the input features to be assigned membership values which was done using Gaussian membership sets. The method gave good results but the robustness of the classification was extremely sensitive to the position and width of the Gaussian membership sets used in the input feature domain. Even a slight change in the width of the membership set results in very poor classification and reliability performance. In regions where sufficient training data was not available, there were more misclassifications. Also, this method does not give the operator a warning when the test points occurred in regions of low training data density.

Finally, a second method was proposed that used a functional model for estimating reliability. The new technique consisted of a regression model which gave an error estimate on an output decision based on the available training data and the training error during the training process of the ANN. This method does not use an additional network and is independent of the architecture of the ANN. The method also has the capability to flag low density regions in the input space. The results of the implementation of this new technique showed that it was successful in reflecting ambiguity. A more detailed analysis can be done only for the critical points flagged as uncertain by the reliability indication process.

BIBLIOGRAPHY

[1] S. Mitra and S. K. Pal, "Fuzzy multi-layer perceptron, inferencing and rule generation," *IEEE Transactions on Neural Networks*, Vol. 6, No. 1, pp. 51-63, January 1995.

[2] S. K. Pal and S. Mitra, "Multi-layer perceptron, fuzzy sets and classification," *IEEE Transactions on Neural Networks*, Vol. 3, pp. 683-697, 1992.

[3] J. A. Leonard, M. A. Kramer and L. H. Ungar, "A neural network architecture that computes its own reliability," *Computers and Chemical Engineering*, Vol. 16, No. 9, pp. 819-835, September 1992.

[4] L. Udpa and S. S. Udpa, "Application of signal processing and pattern recognition techniques to inverse problems in NDE," *Int. J. of Applied Electromagnetics and Mechanics*, Vol. 9, pp. 1-20, 1996.

[5] J. M. Keller, M. R. Gray and J. A. Givens, Jr., "A fuzzy k-nearest neighbor algorithm," *IEEE Transactions*, Vol. SMC 15, No. 4, pp. 580-585, July-August 1985.

[6] J. C. Bezdek and S. K. Pal, *Fuzzy Models for Pattern Recognition*, IEEE Press, 1991.

[7] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, Vol. 4, pp. 4-22, April 1987.

[8] D. R. Hush and B. O. Horne, "Progress in supervised neural networks," *ISP Magazine*, pp. 8-39, January 1993.

[9] S. Haykin, *Neural Networks A Comprehensive Foundation*, Macmillan, New York, 1994.

[10] B. Kosko, *Neural Networks and Fuzzy Systems*, Prentice-Hall, New Delhi, 1992.

[11] S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

[12] H. Nomuru, I. Hayashi and N. Wakami, "A learning method for fuzzy inference rules by descent method," *IEEE International Conference on Fuzzy Systems*, pp. 203-210, March 1992.

[13] Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley, Reading, Massachussetts, 1989.

[14] L. Udpa, S. A. Mandayam, S. S. Udpa, W. Lord and Y. Sun, *Magnetic Flux Leakage Inspection of Gas Pipelines: Neural Networks for Signal Characterization, Compensation and Identification*, GRI Topical Report, ISU, Ames, Iowa, February 1996.

[15] G. W. Snecdor and W. G. Cochran, *Statistical Methods*, Iowa State University Press, Ames, Iowa, Sixth edition, 1967.

[16] S. A. Mandayam, *Invariance Transformations for Processing NDE Signals*, PhD. Dissertation, Iowa State University, Ames, Iowa, 1996.

[17] G. Xie, *Characterization of Gas Pipeline Defects Using Optimal Radial Basis Function Networks*, M. S. Thesis, Iowa State University, Ames, Iowa, 1996.

[18] K. R. Shahani, *Evaluation of Various Preprocessing Techniques for Classification of Ultrasonic Signals*, M. S. Thesis, Iowa State University, Ames, Iowa, 1991.

[19] A. V. Oppenheim and R. W. Shafer, *Discrete-time Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

[20] L. Udpa, *Class notes for Pattern Recognition (EE 529G)*, Iowa State University, Fall 1995.

[21] R. Kuc, *Introduction to Digital Signal Processing*, McGraw-Hill, Singapore, 1982.

[22] J. M. Davis, *Ultrasonic Training for Detection Sizing*, Workshop by Davis NDE Inc., Durham Center, ISU, Ames, Iowa, 1995.

[23] R. W. Weeks, "Stress corrosion cracking in BWR and PWR piping," *Proceedings of the Intl. S. on Environmental Degradation of Materials in Nuclear Power Plants*, pp. 69-86, Myrtle Beach, Aug 22-25, 1983.

[24] M. O. Speidel, "Environmental degradation of steam turbine materials in nuclear power systems," *Proceedings of the Intl. S. on Environmental Degradation of Materials in Nuclear Power Plants*, pp. 113-131, Myrtle Beach, Aug 22-25, 1983.

[25] Ohio State University, "Environmental effects on IGSCC of sensitized steel in high temperature solutions," *EPRI Report*, August 1987.

[26] R. Thompson and D. O. Thompson, "Ultrasonics in NDE evaluation," *Proceedings of the IEEE*, Vol. 73, No. 12, pp. 1716-1755, December 1985.

[27] D. R. Hush and B. G. Horne, "Progress in neural networks," *IEEE Signal Processing Journal*, pp. 8-39, January 1993.

[28] L. Tarassenko, P Hayton, N. Cerneaz and M. Brady, "Novelty detection for the identification of masses in mammograms," University of Oxford, England, 1996.

[29] J.T. Tou and R. C. Gonzalez, *Pattern Recognition Principles*, Addison-Wesley, Reading, Massachussetts, 1974.

[30] L. Udpa and S. S. Udpa, "Application of signal processing and pattern recognition techniques to inverse problems," *Int. J. of Applied Electromagnetics and Mechanics*, Vol. 9, pp. 1-20, 1996.

[31] S. Mandayam, S. S. Udpa, L. Udpa and W. Lord, "Inverse problems in magnetostatic NDE," *14th World Conference on NDT*, pp. 1631-1634, New Delhi, December 1996.

[32] K. S. Narendra and K. Parthasarthy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, Vol. 1, pp. 4-27, 1990.

[33] L. Udpa and S. S. Udpa, "Solution of inverse problems in eddy-current NDE," *Journal of Nondestructive Evaluation*, Vol. 7, Nos.1 / 2, pp. 111-120, 1988.

[34] D. H. Wolpert, "Stacked generalization," *Neural Networks*, Vol. 5, pp. 241-259, 1992.

[35] K. Kim and E. B. Bartlett, "Error prediction for nuclear power plant fault-diagnostic advisor using neural networks," *Nuclear Technology*, Vol. 108, pp. 283-296, November 1994.

[36] K. Kim and E. B. Bartlett, "Error estimation by series association for neural network systems," *Neural Computation*, Vol. 7, pp. 799-808, 1995.

[37] R. B. Chinnam, W. J. Kolarik and C. V. Manne, "Performance reliability prediction of tools in mettle cutting using the validity index neural network," *International Journal of Modeling and Simulation*, Vol. 16, No. 4, pp. 210-217, 1996.

[38] D. K. Ranaweera, N. F. Hubele and A. D. Papalexopoulos, "Application of radial basis function neural network model for short-term load forecasting," *IEE Proceedings for General Transmission Distribution*, Vol. 142, No. 1, pp. 45-50, January 1995.

[39] R. Polikar, *Multiresolution wavelet analysis of EEG signals for the detection of Alzheimer's disease*, M. S. Thesis, Iowa State University, Ames, Iowa, 1995.

[40] A. P. Berens and P. W. Hovey, "Quantifying NDE capability for damage tolerance," *Flaw Detection Reliability Criteria - Methods and Results*, Vol. 1, pp. 25-35, April 1984.

[41] W. D. Rummel, P. H. Todd, Jr., S. A. Frecska and R. A. Rathke, "The detection of fatigue cracks by nondestructive testing methods," *NASA Cr - 2369*, February 1974.

[42] Subramanya G. K., *Probability of detection model for magnetostatic nondestructive evaluation*, M. S. Thesis, Iowa State University, Ames, Iowa, 1994.

[43] E. Parzen, "On Estimation of a probability density function and mode," *Ann. Math. Statist.*, Vol. 3, pp. 1065-1076, 1962.

[44] D. F. Specht, "Probabilistic neural networks," *Neural Networks*, Vol. 3, pp. 109-118, 1990.

[45] J. Moody and C. J. Darken, "Fast learning in networks of locally tuned processing units," *Neural Comput.*, Vol. 1, pp. 281-294, 1989.