THE UNIVERSITY *of* LIVERPOOL

APPLICATIONS OF ARTIFICIAL INTELLIGENCE IN MACHINE CONDITION MONITORING

A THESIS SUBMITTED TO THE UNIVERSITY OF LIVERPOOL
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF ENGINEERING

November 2000

By

Lindsay Burnett Jack

Department of Electrical Engineering and Electronics

Contents

List of Tables

List of Figures

Abstract

Machine Condition Monitoring (MCM) is an area that is gaining increasing importance in manufacturing industry. The increased drive for efficiency and responsiveness by companies means that their manufacturing plants must be available and reliable if the company is to be able to compete in the modern marketplace. The last fifty years have seen a gradual increase in the degree of importance which is attached to both predictive and preventative maintenance. Modern systems are seeing the introduction of artificial intelligence, remote sensing, and data abstraction and fusion on a very large scale. This thesis investigates some of the different techniques - both available and emerging, that are coming to the fore.

Artificial Neural Networks (ANNs) are an area that has seen much developmental work carried out in the MCM arena. Most works have been concerned with the use of a standard preprocessing technique - the FFT. Various different alternative methods of preprocessing have been advanced, and a comparison is carried out between different methods of feature extraction, and their impact on the classification performance of three different neural networks; the Multilayer Perceptron (MLP), Radial Basis Function (RBF) network, and the Probabilistic Neural Network (PNN). The performance of each with the different feature extraction techniques is compared, with good accuracy results (95-100%) for the experimental data used.

Support Vector Machines are a recent development in the field of statistical

learning theory; showing much promise on classification tasks, as yet they have seen little application in MCM type problems. One of the problems that faces the use of the SVMs in MCM is the lack of large amounts of training data. Two different modifications to the standard SVM are proposed, which yield very good results (99 - 100%) in the multiclass fault characterisation problem.

Genetic Algorithms are a technique which has gained acceptance in problems requiring optimisation with an extremely large search space. GAs are used in combination with the MLP and SVM classifiers to determine whether the performance of the classifiers can be improved further, and made more robust by the removal of those features which confuse the classifiers, causing performance to deteriorate. Using the GA with the ANN, the reduced feature set allows the MLP to reach 100% accuracy using only 6 inputs from a possible 156, while the SVM is capable of achieving 100% classification using only 4 inputs out of 156, significantly reducing the computational complexity of the problems while increasing the accuracy.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Copyright

Acknowledgements

There are many people who are due thanks as a result of their help throughout the last three years of this PhD. First and foremost, thanks must go to my supervisor, Professor Asoke Nandi, who has been a constant guiding hand throughout this period, including the move from Strathclyde University to Liverpool. Thanks must also be expressed to the University of Liverpool, the University of Strathclyde, Weir Pumps Ltd., of Cathcart, Glasgow, and Solartron Mobrey Ltd., of Farnborough, for financial and practical support at various stages of this work.

Strathclyde was a great place to work, with some great people. Probably the other person due the most thanks from this thesis is Dr. Andrew McCormick, who's earlier work in the area and technical advice was always useful, and his sense of humour helped him to tolerate me at my most trying. Strathclyde was certainly not all work, and thanks must go to Saul, Daniel, GCF, Harve, Druti, Moritz, Stoeps, Bob and Eugen for their sense of humour, joke emails, hill walking and frequent trips to the pub (but we won't talk about Inverarnan too much....). Mounir, Charles, Mark and JJS for their footballing prowess (?), even if two of them were Celtic supporters.....

Liverpool was the site of the last 18 months of the PhD. Thanks must be expressed to a lot of people there as well; firstly, to Jaqui Cowan, Brenda Lussey and David Turner for helping to make the move a (relatively) smooth changeover, and helping us to find our feet in the weeks before and immediately after the

move. This is truly appreciated, and made life much easier than it could have been. Socially, Liverpool has also been great - of the postgrads, special thanks must be made to Kev Keller for introducing us to many of the other postgrads, and the delights of Liverpool night life. Derek, Giles, and Lee are also due a big thankyou for their sense of humour, and nights out or in the pub - and how could I forget Alison, Louise, Jane and Tina, for inviting Frank and I out for the first night out "with the secretaries". Being able to go out and relax with all these people helped us settle, and has made Liverpool a great place to be working. Thanks guys (and girls!).

Due acknowledgement must also be made to my colleagues in the lab here at Liverpool. I would like to extend my thanks and appreciation to Frank Herrmann, Dr. Vicente Zarzoso, and Dr. Juneyule Lee, who came down from Glasgow with me - without their tolerance and sense of humour in the first months here, things would have been a much less friendly place (even when I hosed the system for a week...). Thanks to them, and also Dennis Wong, one of the newer students here who has made working together a pleasure rather than a chore. Cheers guys.

Lastly, thanks must also be made to those people who gave me their support and encouragement (and the occasional kick up the backside) during the writing of this work; my family, for encouraging me to go and do what I wanted to do in the first place, and also Hannah Leyland, for her support and encouragement during the writing process. It is truly appreciated.

Glossary of Abbreviations

ANN	Artificial Neural Network
BAB	Branch And Bound
CF	Crest Factor
DFT	Discrete Fourier Transform
ECG	ElectroCardioGram
FA	False Alarm
FFT	Fast Fourier Transform
FNR	Fault Not Recognised
GA	Genetic Algorithm
GSBS	Generalised Sequential Backwards Selection
GSFS	Generalised Sequential Forward Selection
MCM	Machine Condition Monitoring
MLP	MultiLayer Perceptron
PCA	Principle Components Analysis
PDF	Probability Density Function
PNN	Probabilistic Neural Network
RMS	Root Mean Square
SBS	Sequential Backwards Selection
SFS	Sequential Forward Selection
SP	Shock Pulse
STFT	Short Time Fourier Transform

Chapter 1

Introduction

1.1 Motivation

Machine Condition Monitoring (MCM) is an area which is gaining increasing acceptance within the manufacturing industry. Greater manufacturing efficiency has created a demand for more efficient, reliable, and effective techniques to monitor the health of machinery. In large scale activities such as power generation, or the petrochemical industry, unplanned downtime can be extremely costly, with extremely high losses being recorded on an hourly basis. Various techniques have been developed over the last fifty years, with varying degrees of complexity and success in the approach to problems posed by MCM.

One of the areas which has seen most interest recently is the development of unsupervised or partially autonomous systems which are capable of performing everyday monitoring tasks without requiring the presence of skilled personnel for supervision. This has created a demand for the integration of artificial intelligence into everyday systems. Artificial intelligence itself is a very popular area, with many different techniques, systems, and approaches being developed and investigated currently. There is a distinct need for evaluation of many of these

novel techniques in the MCM arena and for an assessment of their usefulness for further study and development. This has been the aim of the work carried out in this thesis, to investigate the possibility of certain approaches and techniques that have not been used widely or are currently under development, and determine the potential usefulness in this field.

The Probabilistic Neural Network (PNN) is an artificial neural network (ANN) which has been in existence for some time; however, its usage within MCM has been limited, with most researchers using multilayer perceptron (MLP) or radial basis function (RBF) networks for classification. It was decided to perform a comparison between the three network types in order to try and determine the usefulness of the PNN for MCM applications.

Support Vector Machines (SVMs) belong to a very new and vigorous area of research in the neural network community at present. They have been shown to achieve some of the highest results recorded on benchmark classification tests [1]. Much research has been carried out in pattern classification; however, as far as the author is aware, very little investigation has been carried out to test the usefulness of the SVM in MCM applications and to determine how robust the SVM is when faced with problems of small training sets. This is one of the main problems with all classifiers in the MCM arena, as training data can be extremely hard to come by, and a successful technique capable of handling these problems would be extremely useful.

One of the greatest challenges facing human operators of large and complex plant is the problem of data overload. Modern control and monitoring systems can have hundreds of different sensors and alarms. When a problem does arise, it can be very difficult to determine what is actually happening, as several, or even tens of alarms are triggered. Forms of data abstraction which can take large amounts of data, and refine the raw contents into a small quantity of useful *information*

are needed. The abstraction process allows the operators or supervisory system to understand better what has happened, and is attracting a lot of research interest at present. However, one of the problems of data abstraction is determining which data streams provide useful information, and which are erroneous or uncorrelated with the problems of interest. Research on feature selection has been carried out quite widely , and yet little application of this has been made in the MCM area, with system integrators usually using the experience and knowledge of human experts to select inputs. However, using human intuition to govern the selection procedure will only detect those relationships that the human is aware of. A form of automated feature selection may be able to find more accurate correlations in other data streams that had not been previously considered. This work attempts to address some of these areas, to try and determine the applicability of some of the novel techniques developed over the last few years in the MCM arena.

1.2 Original Contribution

The original contributions believed to be made in this work are:

- A comparison of the performance of the probabilistic neural network with the multilayer perceptron and radial basis function networks, with a variety of different statistical and spectrally based features.
- An evaluation of the performance of a conventional multiclass support vector machine with condition monitoring data, and assessment of the problems inherent in the use of the conventional SVM.
- The creation and evaluation of two modified kernels for the SVM, which take better account of the distribution of data within feature space, and the relative proportions of the data with regard to the normalised dataspace.

These show a marked improvement in the classification performance over that of the conventional kernel.

- An evaluation of GA based feature selection with ANNs, and an investigation of the impact on classification performance of different datasets before and after feature selection.

- An evaluation of the performance of the SVM after feature selection, and demonstration that all three SVMs used in this thesis are capable of performance equivalent to, and in some cases better than that of, the best ANN available after feature selection. Furthermore, it is also demonstrated that feature selection allows the SVM to deal successfully with small quantities of training data.

1.3 Organisation

This thesis is broken into seven separate chapters. Chapter 1 deals with the introduction, motivation and organisation of the thesis. Chapter 2 provides a short historical background of condition monitoring, and a brief overview of the different statistical, signal processing, classification and feature selection techniques that are used within the different areas that this thesis deals with. Chapter 3 provides an overview of the data sources used in the experiments, along with a brief examination of the characteristics of the different data. The pre-processing carried out on the data prior to being used for the experiments is also detailed. Chapter 4 is a comparison of the performance of three different ANN types with the data prepared earlier. Chapter 5 introduces the SVM, and introduces two modified kernels for use with multiclass data; these are then evaluated with the data prepared earlier. Chapter 6 examines the impact of feature selection on

the performance of both ANN and SVM based classifiers, and evaluates the impact of feature selection on the robustness of the classifiers. Chapter 7 presents conclusions and further work, on the basis of the work carried out in this thesis.

1.4 Publications

Journal Papers

- McCormick A C, Nandi A K, and Jack L B. Digital Signal Processing Algorithms in Condition Monitoring, International Journal of Comadem, Vol 1, No. 3, pp 5-14, 1998.
- McCormick A C, Nandi A K, and Jack L B. Application of Periodic Time Varying Autoregressive Models to the Detection of Bearing Faults. Proceedings of The Institute of Mechanical Engineers, Part C(6), Vol. 212, pp. 417-428, 1998.
- McCormick A C, Nandi A K, Jack L, Diagnosis of Rolling Element Bearing Faults using Radial Basis Function Networks, Applied Signal Processing (1999), 6:25-32.
- Jack L, Nandi A K, Genetic Algorithms for Feature Selection in Machine Condition Monitoring with Vibration Signals, IEE Proceedings on Vision, Image and Signal Processing, Vol 147, No. 3, pp. 205-212, 2000.

Conference Papers

- Jack L B and Nandi A K, Feature Selection for ANNs using Genetic Algorithms in Condition Monitoring, Proceeding of the 7th European Symposium on Neural Networks (ESANN99), D-Facto, Brussels, pp. 313-318,

1999.

- Jack L B and Nandi A K, Genetic Algorithms for Input Selection in Condition Monitoring, Proceedings of Comadem 99, Coxmoor Publishing, Oxford, pp. 381-388, 1999.

- Jack L B and Nandi A K, Genetic Algorithms for Input Selection in Condition Monitoring, Proceedings of QRM 2000, University of Oxford,pp. 145-150, March 30-31, 2000, Published by the Institute of Mechanical Engineers.

- Jack L B and Nandi A K, Comparison of Neural Networks and Support Vector Machines in Condition Monitoring Applications, accepted for Comadem 2000, Dec. 3-8, 2000, Houston, TX, USA.

- Jack L B and Nandi A K, Pattern Recognition of Rolling Element Bearing Faults using Support Vector Machines, accepted for International Conference on Communications, Computers and Devices, ICCCD 2000, Dec. 14-16, 2000, Kharagpur, India.

Papers Submitted for Publication

- Jack L B and Nandi A K, Support Vector Machines for detection and Characterisation of Rolling Element Bearing Faults, submitted to Proceedings of IMechE, Part C.

- Jack L B and Nandi A K, Using and Optimising Support Vector Machines for Detection of Fault Conditions, submitted for publication to Mechanical Systems and Signal Processing.

Chapter 2

Preliminaries

2.1 Machine Condition Monitoring

Machine Condition Monitoring (MCM) is an area of engineering that is gaining acceptance and importance in many parts of manufacturing industry. Modern day production plants are expected to run for 24 hours a day, seven days a week. Lost production due to unexpected failure of machinery is regarded as a serious and high cost problem. In order to ensure that production runs successfully, industry has created a demand for techniques that are capable of recognising both the development and severity of a fault condition within a machine system. Machine Condition Monitoring was developed to meet this need.

Condition monitoring itself is a fairly recent development, being able to trace its roots back to the late 1950s and early 1960s. However, the very earliest developmental work was carried out during the second world war.

2.1.1 Historical Background

Pre World War II

Prior to W.W. II, most manufacturing was done with relatively simple machines; this made the machines easy to repair and fairly reliable, as a result, downtime was a not a big issue as the level of automation that existed in many industries was relatively small, and so manufacturing yields were correspondingly lower. Generally speaking, there was no use of diagnostic or monitoring techniques for detecting faults. When a machine broke down it was repaired, with servicing being carried out on the basis of the number of hours running, or miles covered, etc. The overall emphasis was on fault *repair*, not prevention. This made for a very reactive system, forcing large plants to carry high stocks of spares as a contingency measure. This also meant that large quantities of money were tied up in material assets, such as spare parts for the machines.

World War II

World War II marks the beginning of condition monitoring [2]. The reduction in the skilled labour force during the war combined with the demand for higher production quantities, meant that attention was focused on the failure point of the manufacturing process, and also of the reliability of machines. The increasing complexity of machines - aero engines being one of the driving examples, led the military to investigate the causes of failures in the engines, as one of the big problems at the time was trying to keep as many warplanes in the air, and keep them as reliable as possible in long raids. This led to a demand for some type of monitoring in the systems, to try and provide warning of a failure, so that action could be taken to repair the damage.

The emphasis on detection was simple at this stage, with the use of techniques

that looked for physical signs of distress, such as wear debris, and led to the use of magnetic sump plugs, and other simple devices. Most of the work done here was concerned with examining the lubrication medium for signs of distress in the form of metallic fragments and chips that occur during fatigue. The use of boroscopes for in-depth examination of items like gearboxes was also introduced around this time. No attempt was made to examine simple physical manifestations of the problems, such as vibration, and detect problems that way. For the first time, engineers began to look for signs of failure and try to avoid catastrophic failure where possible, however the response was still a very much reactive system; mechanics would attempt to find the damaged components in order to replace them, without attempting to find the underlying cause of the problem.

Post World War II

In the aftermath of WW II, as the military demobilised many of the ideas that had been pioneered by organisations such as the US Air Force, the Office of Naval Research, and the British Ministry for Aircraft Production, began to find uses in the commercial and industrial worlds. Among the earliest of the "early adopters" were the US railways, who converted most of their fleets from steam to diesel haulage in the years following the war. This led to a drive to improve the reliabilty of the locomotives, which were expected to run on intense long haul schedules within the US. While the military had been mainly concerned with the detection of metallic particles in the lubrication oil, the railways were also interested in the condition of the lubricating oil itself, and added tests that checked the quality of the oil; this was a further move toward a more preventative agenda in maintenance, with more attention being focused on repair as the fault develops, rather than after a catastrophic failure.

At this time, many new techniques were developed for monitoring and diagnosis; however the emphasis on these was in increasing the sensitivity of detection, not with assessment of condition or in actually trying to determine the cause of the problem. As mechanisation increased, the cost of buying plant and also the cost of maintenance as a fraction of the total running cost of the plant increased significantly; as a result, management started to take a greater interest in finding ways of reducing the maintenance costs, and maximising the life of production plant.

The Modern Era

With the introduction of computers into manufacturing environments, this opened up many new areas for the analysis and monitoring of systems; suddenly, it became to monitor the systems automatically, and on a larger scale than had been hitherto possible. Increased computational capacity meant that new methods of analysis, which were computationally intensive (and up until then, impractical) were now feasible for use in a working environment. The development of the science of Tribology, studying the cause of mechanical fatigue and failure, contributed a lot towards better understanding of the root causes of many problems, and as a result, to find new techniques that were more capable of detecting and gauging the severity of faults. The 1960s and 1970s saw the introduction of vibration studies on a widespread basis, as engineers realised that they could monitor the health of a machine by examining its physical behaviour characteristics, and gain a lot of information from this. Perhaps one of the biggest contributions towards this end was the development of the Fast Fourier Transform (FFT), which for the first time, gave a computationally practical way of calculating the frequency spectrum of machine vibrations. Additionally, the new computers helped the development of artificial intelligence, and this began to to make an impact

in maintenance and condition monitoring, in the form of expert systems, which were pioneered by locomotive manufacturers to help maintenance fitters target problems from symptoms. The development of neural networks has also opened up many areas of research that are pertinent to condition monitoring, as pattern recognition algorithms readily transfer into the MCM arena. One of the biggest problems for MCM however, is the fact that most pattern recognition algorithms are based around the assumption that it is possible to provide examples of all the likely cases that a system will see. Unfortunately, this is not the case with MCM, and so specialised approaches have had to be developed.

New management techniques, such as Just In Time (JIT), gave rise to the situation where minimal spares were held in house in an attempt to reduce money. This however places a greater burden on the maintenance sections, as the lack of spares held in house means that a greater emphasis on the early detection of problems is required. The problem is further compounded by the fact that one JIT factory going down can stop the production in other factories, as they rely on the output of the first plant. Problems of this nature are pushing industry towards a drive for zero downtime, which in turn is causing condition monitoring to become ever more complex and widespread.

Modern MCM systems are moving towards a holistic approach; commonly, a manufacturing line will have a number of different sensors feeding back into a SCADA (Supervisory Control And Data Acquisition) system, which monitors the performance of the system, and looks for problems. Traditional methods have been based around the use of two basic approaches; the use of single feature threshold performance measures that give a very general indication of the existence of a fault but no indication of the nature of the fault, or alternatively the use of frequency derived indicators which, while often extremely reliable, can be time consuming to compute and require a detailed knowledge of the internal

components of a machine and their relative speeds in order to make a good classification of the fault. Figure 2.1 gives an approximate overview of the evolution of MCM techniques over the last 50 years.

The challenge is now to develop improved algorithms and supervisory functions that are capable of providing *information*, rather than data, to operators. Techniques such as neural networks, data fusion, genetic algorithms and expert systems all have their parts to play in this. Research has been carried out into the use of artificial neural networks for fault diagnosis, and the results are promising [3–8]; however many of the input features used require a significant computational effort to calculate, and work is ongoing to try and improve the performance of the systems. Work is, and has been ongoing in the areas of data fusion [8–10], genetic algorithms [11, 12] and expert systems [8, 13–16].

2.2 Feature Extraction Algorithms

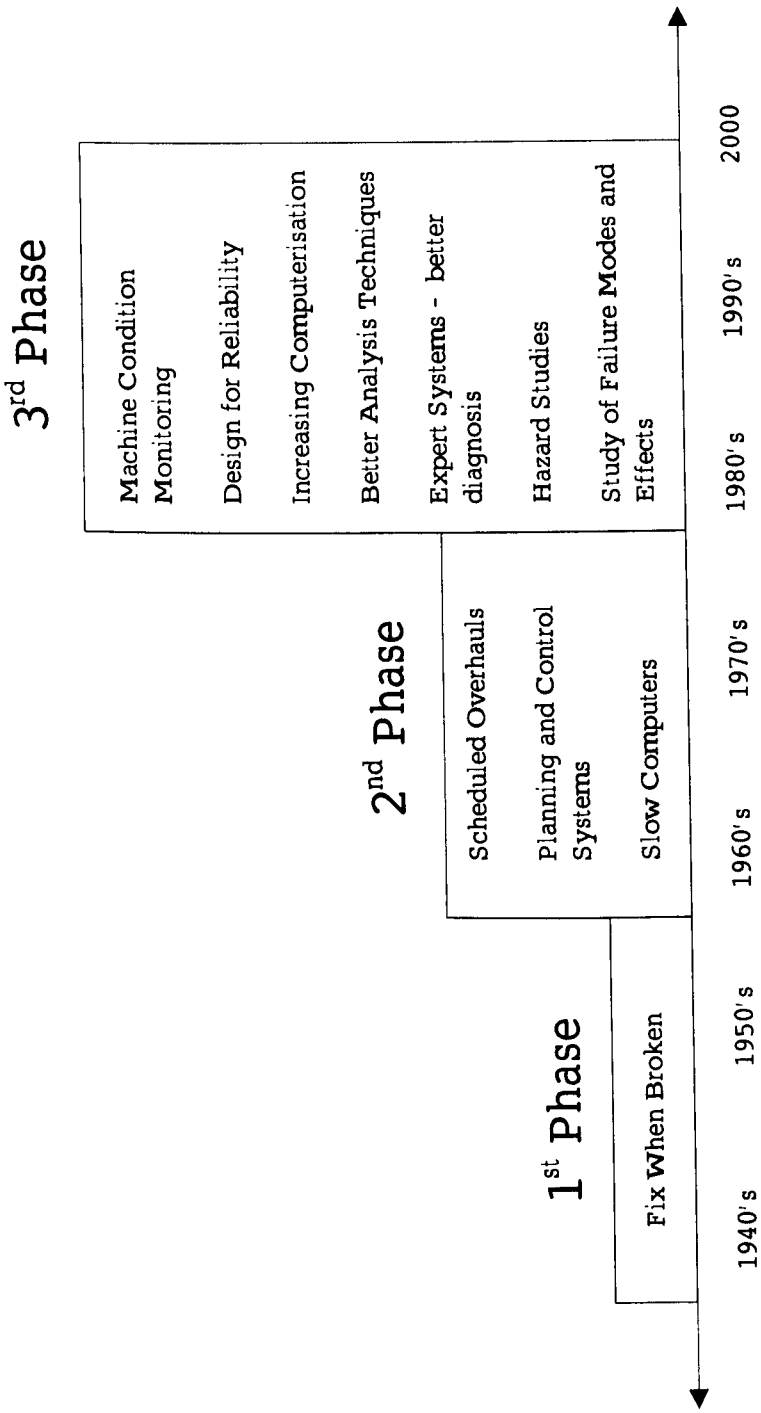
2.2.1 Statistical Features

In order to describe some of the statistical features used, it is necessary to introduce the basic concepts of probability and statistics. these are covered in much greater depth in many books (e.g. [17, 18]) on statistics and random processes; however for convenience the basics are summarised below.

Statistical Preliminaries

$P(X)$ is the probability of an event X occurring. This can also be expressed in experimental terms as:

$$P(X) = \lim_{N \to \infty} \frac{N_X}{N} \tag{2.1}$$


1st Phase
Fix When Broken
2nd Phase
Scheduled Overhauls
Planning and Control Systems
Slow Computers
3rd Phase
Machine Condition Monitoring
Design for Reliability
Increasing Computerisation
Better Analysis Techniques
Expert Systems - better diagnosis
Hazard Studies
Study of Failure Modes and Effects
1940's
1950's
1960's
1970's
1980's
1990's
2000

where N_X represents the number of occurrences of X, and N represents the total number of occurrences measured. The probability that the event X lies within the interval $[-\infty, \infty]$ is $P(X) = 1$; however the probability that X will lie below a threshold value x, defines a function, $F(x)$.

$$F(x) = P(X \leq x) \tag{2.2}$$

$F(x)$ is known as the cumulative probability distribution function. $F(\infty) = P(X \leq \infty) = 1$, while correspondingly $F(-\infty) = P(X \leq -\infty) = 0$. The cumulative probability distribution function shows how X is distributed over the range of possible values. A simpler definition of the distribution of X over it's range, is to take the derivative of $F(x)$, creating the Probability Density Function (PDF), $f(x)$:

$$f(x) = \frac{dF(x)}{dx} \tag{2.3}$$

The PDF shows how the random variable is distributed about the range of possible values, and indicating whether the values concentrate about one or more areas.

The PDF allows an expectation operator to be defined:

$$E\{\mathbf{x}\} = \int_{-\infty}^{\infty} x\, f(x) dx \tag{2.4}$$

This gives the expected, or mean value of the random variable $\mathbf{x}$. The expectation operator is used to define the statistical moments of a random variable. The expectation operator is the basis for several different statistical measures used here; most notably statistical moments and variance/standard deviation.

The n^{th} order moment of $\mathbf{x}$ is defined:

$$m_{\mathbf{x}}^{(n)} = E\{\mathbf{x}^n\} \tag{2.5}$$

This is useful when $\mathbf{x}$ is stationary and ergodic. For sampled data, the moments can more easily be estimated using:

$$m_{\mathbf{x}}^{(n)} = \frac{1}{N}\sum_{i=0}^{N} x_i^n \tag{2.6}$$

The variance of a random variable $\mathbf{x}$ can be calculated:

$$\sigma^2 = \int_{-\infty}^{\infty} (x - m_{\mathbf{x}}^{(1)})^2 f(x)\, dx \tag{2.7}$$

where σ is the standard deviation of $\mathbf{x}$. The standard deviation/the variance can be used as a measure of how "wide" a distribution is, and can be used as a characteristic of the signal.

The characteristic function of a random variable $\mathbf{x}$, $\Upsilon(s)$, can be expressed in terms of it's PDF, $f(x)$, by the equation:

$$\Upsilon(s) = \int_{-\infty}^{\infty} f(x) e^{sx}\, dx \tag{2.8}$$

Differentiating $\Upsilon(s)$ n times, it can be seen that

$$\Upsilon^n(s) = E\{\mathbf{x}^n e^{s\mathbf{x}}\} \tag{2.9}$$

and therefore

$$\Upsilon^n(0) = E\{\mathbf{x}^n\} = m_{\mathbf{x}}^{(n)} \tag{2.10}$$

So, the n^{th} derivative of $\Upsilon(0)$ gives the n^{th} moment of the random variable - which gives rise to the alternative name of the characteristic function - the *moment generating function.*

Taking the natural logarithm of the characteristic function generates the second characteristic equation of the random variable.

$$\Psi(s) = \ln \Upsilon(s) \tag{2.11}$$

The cumulants, $C_{\mathbf{x}}^n$, of a random variable $\mathbf{x}$ is defined as:

$$C_{\mathbf{x}}^{(n)} = \frac{d^n \Psi(0)}{ds^n} \tag{2.12}$$

These can also be expressed as combinations of some of the lower order moments. The first four moments are defined as follows:

$$C_{\mathbf{x}}^{(1)} = m_{\mathbf{x}}^{(1)} \tag{2.13}$$

$$C_{\mathbf{x}}^{(2)} = m_{\mathbf{x}}^{(2)} - (m_{\mathbf{x}}^{(1)})^2 \tag{2.14}$$

$$C_{\mathbf{x}}^{(3)} = m_{\mathbf{x}}^{(3)} - 3m_{\mathbf{x}}^{(2)}m_{\mathbf{x}}^{(1)} + 2(m_{\mathbf{x}}^{(1)})^3 \tag{2.15}$$

$$C_{\mathbf{x}}^{(4)} = m_{\mathbf{x}}^{(4)} - 3(m_{\mathbf{x}}^{(2)})^2 - 4m_{\mathbf{x}}^{(3)}m_{\mathbf{x}}^{(1)} + 12m_{\mathbf{x}}^{(2)}(m_{\mathbf{x}}^{(1)})^2 - 6(m_{\mathbf{x}}^{(1)})^4 \tag{2.16}$$

2.2.2 "Conventional" Indicators

Before the (relatively recent) widespread availability of significant computing power for use in condition monitoring, a number of statistically based performance measures were introduced, and these provided single figure assessments of the condition of rolling element bearings. These were comparatively simple to calculate, and gave an indication of whether a bearing was in a state of distress, or within normal operating parameters. These are not generally capable of discriminating between different failure types however, and can only show the degree of distress that a bearing is under. Probably the three most common measurements used are shock pulse, crest factor, and kurtosis.

Shock Pulse

Whenever there is metal to metal contact in the bearing, its causes a high frequency shock pulse to propagate through the bearing, activating the transducer. Shock Pulse (SP) analysis relies on a specialised transducer, which has a resonant frequency of 32-36kHz. This resonant frequency is too high to be excited by the usual low frequency energy contained within a normal signal from a roller bearing. This then produces a signal showing the impact of the balls against the defect. By examining the average or *carpet* values of the signal, and the peak values of the signal, information can be gleaned about the state of lubrication of the bearing, and also the likelihood of the existence of a defect. This classification is not capable of distinguishing between different fault conditions of the bearing however, just the existence of a fault condition.

Crest Factor

The Crest Factor (CF) is a commonly used performance index, intended specifically for the detection of bearing faults. Simply, the CF is the ratio of the peak magnitude within a time series, divided by the RMS value of the signal. Signals that have a relatively small high frequency content, but containing large spikes, will cause relatively high numbers to be generated. This works well in the early stages of a fault developing, as the degree of damage to the bearing does not generate a large high frequency component, however as the degree of damage increases, the high frequency component of a vibration signal increases, and as a result the RMS value increases, causing the CF value to decrease. If the bearing is only checked on a periodic basis, there is a risk that the actual warning provided by CF may be missed.

Kurtosis

Kurtosis is a commonly used measure of damage. Two different definitions are commonly used:

$$\gamma_4^{(m)} = \frac{m_{\mathbf{x}}^{(4)}}{\left(m_{\mathbf{x}}^{(2)}\right)^2} \tag{2.17}$$

$$\gamma_4^{(c)} = \frac{C_{\mathbf{x}}^{(4)}}{\left(C_{\mathbf{x}}^{(2)}\right)^2} = \gamma_4^{(m)} - 3 \tag{2.18}$$

These two definitions produce different values for the same signal. For a zero mean, Gaussian signal, equation 2.17 will give a value of three, while equation 2.18 will give a value of zero. The kurtosis value emphasises the length of the "tails" of a distribution. Signals which exhibit lots of sharp impacts - such as

those that occur when the rolling elements of a bearing strike a defect, will have high kurtosis values, while signals with little or no "spike" content will have low values.

2.2.3 Spectral Features

Many of the problems that occur in MCM have frequency components that can often identify them, as certain parts of machines will spin at speeds that are multiples of the base rotation frequency of the machine. If the construction of the machine is known, then it is possible to calculate the different rotational speeds of different parts of the machine, and as a result, by examining the frequency spectrum of the machine, the nature or location of the fault may be identified.

The Discrete Fourier Transform

The Discrete Fourier Transform (DFT) forms the cornerstone of spectral processing used commonly in signal processing. The transform provides information about any sampled signal, returning complex frequency spectra that can in turn be used to calculate both the phase and magnitude spectra of a signal. The DFT is defined by the equation 2.19. Each one of the frequency bins created as a result of this process has a resolution of $\frac{f_s}{N}$, where f_s is the sampling frequency. Applying a DFT directly to a time series causes large peaks to be generated at both ends of the spectrum; this is caused by the assumption that the signal is periodic, and any discontinuities in the signal cause the large peaks to be generated to compensate for this. In order to alleviate this problem, a window is applied to the data, that ensures that the beginning and end of the time sequence start at zero.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} \tag{2.19}$$

Calculating the DFT is a fairly computationally intensive job, however Cooley and Tukey's Fast Fourier Transform [19] (FFT), proposed in 1965, opened up the field of simple spectral analysis. This method, and its variants, are still found commonly in use today.

Power Spectrum

The power spectrum is a development of the DFT. The DFT returns a complex answer to a real valued problem, and examining the individual components it is hard to make sense of the what is actually happening within a signal. The power spectrum calculates the average of the magnitude of the DFT, and is defined:

$$S_{xx}(k) = E\{X(k)X^*(k)\} \tag{2.20}$$

where $X^*(k)$ is the complex conjugate of $X(k)$. This is probably the most common spectral technique used. By examining the peaks, harmonics can be spotted in the spectrum, and from this the frequencies of interest can be calculated, and then related back to the faulty components. However, in many cases it can be difficult to distinguish harmonic peaks within a spectrum, dependent upon the degree of noise present in the raw data, and the number of other vibration sources in the location where the data was sampled.

Cepstrum Analysis

Cepstrum Analysis was originally proposed by Bogert *et al* around 1963 [20], and used for work in seismic echo detection . Later applications developed in speech and ECG related areas [21,22], and over time, the technique came to be used in vibration analysis and monitoring.

Cepstrum analysis can be expressed as the inverse FFT of the logarithm of the power spectrum of the vibration signal, as shown in equation 2.21. This process creates another spectrum, which when read, gives an indication of the presence of harmonics in the power spectrum. Peaks in the Cepstrum correspond to the existence of harmonics in the power spectrum, with the *quefrency* giving the period of the separation between the harmonics in the frequency domain, while harmonics on the Cepstrum plot (known as *rahmonics*) would indicate the existence of several different harmonics which are a multiple of the frequency of interest.

$$C(\tau) = |X^{-1}(log(S_{xx}(k)))| \tag{2.21}$$

The Cepstrum is used fairly commonly in vibration monitoring because it is often the case that the power spectra taken from a machine can be extremely noisy, due to vibration coming from external influences, such as other machines operating on the same area. Thus, it can be very difficult to determine what parts of the signal constitute harmonics and what is noise in the power spectrum, and cepstrum analysis provides a quick way of isolating the harmonic frequencies present in the signal.

Higher Order Spectra

The power spectrum does not provide any information about the correlation between different frequencies of the spectrum; while it can show what proportion of a signal comes from a given frequency band, it provides no information about the interactions between multiple frequencies in the signal; many of the faults exhibit characteristic harmonics, which have a constant phase relationship between them.

Bispectral analysis is a development of the power spectrum, where the autocorrelation of the signal $E\{x(t)x(t+\tau)\}$ can be used to highlight the contributions of single frequencies to the power of the signal , through the use of the τ parameter. The bispectrum and trispectrum extend this further, by looking at the correlations between multiple frequencies within a signal, and showing any relationships that may exist between the different frequency components of the signal, particularly phase relationships. The bispectrum can be calculated relatively easily through the equation:

$$B(f_1, f_2) = E\{X(f_1)X(f_2)X^*(f_1 + f_2)\} \tag{2.22}$$

where $X(f)$ is the Fourier transform of $x(n)$ at a given frequency f. Correspondingly, the trispectrum can be calculated along similar lines:

$$T(f_1, f_2, f_3) = E\{X(f_1)X(f_2)X(f_3)X^*(f_1 + f_2 + f_3)\} \tag{2.23}$$

Space here precludes a comprehensive explanation for the motivations behind the use of the higher order spectra, however [23–25] cover the theoretical basis for the use of higher order spectra. Several experimental results have also been published [26–29] on the use of the bi and trispectrum in condition monitoring,

and also one of its derivatives, bicoherence [30], which is a normalised form of the bispectrum, and can provide more informative results than the raw bispectrum.

2.2.4 Time-Frequency Distributions

Fourier, or frequency, analysis examined thus far is based upon the assumption that the system being dealt with is running in a steady state, with a periodic signal content; this is one of the fundamental assumptions behind Fourier analysis. However, there are many situations where a system is not running in a steady state, or the objects of interest are only of a short duration. In this sort of situation, traditional Fourier analysis does not meet the requirements, and alternative strategies have been developed to allow examination and analysis of the signals.

As an example, consider the case of a machine rundown, where the machine may be taken from a steady state running condition, and stopped. As the machine moves from the steady state through to a stop, there maybe areas of interest in the rundown cycle that it would be desirable. However, due to the fact that the speed of the machine is decreasing to halt, meaningful frequency analysis is very difficult, as the frequency of a periodic fault will vary as the speed varies. There is no adequate way to examine *when* an event occurred during the rundown period, as a straightforward FFT will only show that something has occurred (assuming of course that it is possible to distinguish from the general machine related noise during the rundown). An FFT taken over the whole period of the rundown will not give any useful information of this nature.

Short Time Fourier Transform

As a means of dealing with this, the Short Time Fourier Transform (STFT) began to be used. However, by breaking the long time period into a series of

smaller time windows taken consecutively, it is possible to create a "waterfall" diagram, that shows the frequency content of the signal at different periods in time, giving an indication of the how the frequency content of the signal changes. However, the shorter time windows mean that the frequency resolution of the FFT will decrease markedly, and it becomes much more difficult to distinguish exactly at what frequency a problem is. There is a trade off between the size of the time windows used (which governs the time resolution of the analysis), and the frequency resolution (which is governed by the size of the window used in the FFT).

Wigner-Ville Distribution

The development of the Wigner Ville distribution can be traced back into the early 1930s; the basic ideas were developed in the field of quantum statistical mechanics where there was a problem that was mathematically analogous to time frequency distributions. Wigner first proposed what is now commonly recognised as the Wigner-Ville distribution in 1932 [31]. This work was developed by Ville into what would now be recognised as modern time frequency analysis [32]. In similar way to the Fourier transform, which decomposes a signal into its frequency components, the Wigner-Ville distribution describes a signal content in terms of both frequency and time. It is expressed as:

$$W(t,\omega) = \frac{1}{2\pi}\int x(t-\frac{\tau}{2})e^{-j\tau\omega}x(t+\frac{\tau}{2})d\tau \tag{2.24}$$

However, the Wigner-Ville distribution is not without problems in itself; one of the worst of these is the introduction of so called "artifacts" into the distribution, in areas of the time axis where it would be expected that there would be no frequency content [33]. To counter this, a modified version was proposed by Choi

and Williams [34], which has become more widely used. It reduces the effect of the artifacts by the use of a kernel to minimise the cross terms. The Choi-Williams distribution is defined:

$$P_{CW}(t,\omega) = \frac{1}{4\pi^{3/2}} \int\int \frac{1}{\sqrt{\tau^2/\sigma}} e^{-\left[(u-t)^2/(4\tau^2/\sigma)\right] - j\tau\omega} x(u-\frac{\tau}{2})x(u+\frac{\tau}{2})dud\tau \tag{2.25}$$

where σ is constant.

The Wigner-Ville distribution has been used in the diagnosis of faults of rotating machines [35, 36], robotic arms [37], and particularly in the analysis of gearbox faults [38–40]. It is however, computationally expensive to perform, and this perhaps hampers its wider use.

Wavelets

Wavelets [41, 42] are a comparatively recent development in the domain of time-scale analysis. One of the fundamental assumptions behind Fourier analysis is that any periodic signal can be decomposed into a number of different sine and cosine functions with an infinite period. This works very well for steady state signals, however as discussed earlier, this is often not the case in real life. The Fourier transform works on the basis that a signal can be localised in terms of frequency; the wavelet transform works on the basis that signals can be localised in both frequency and time. Rather than use continuous signals of infinite duration, the wavelet transforms uses a signal with finite duration, usually short. By moving this *wavelet* in time, it is possible to imitate the occurrence of "events" in the signal, and by stretching and compressing the wavelet, it is possible to model different frequencies, thus decomposing a given signal in terms of frequency and time.

Mathematically, a wavelet is a function of short duration that is localised both in time and frequency. One of the basic requirements for a waveform to be regarded as a wavelet Ξ, is that it should have a zero average when integrated:

$$\int_{-\infty}^{\infty} \Xi(t)dt = 0 \tag{2.26}$$

Translation (shifting the wavelet in time) and dilation (stretching/compressing)terms are introduced:

$$\Xi_{u,s}(t) = \frac{1}{\sqrt{s}}\Xi\left(\frac{t-u}{s}\right) \tag{2.27}$$

where u represents the time shift, and s is the scaling factor that is to be used on the wavelet. By correlating the function of interest with the wavelet, and using translation as necessary, it is possible to create the Wavelet Transform of the signal:

$$Wf(u,s) = \int_{-\infty}^{\infty} f(t)\frac{1}{\sqrt{s}}\Xi^*\left(\frac{t-u}{s}\right)dt \tag{2.28}$$

Wavelets have a similar tradeoff to the STFT, however it works in a slightly different fashion. As the wavelet becomes more and more tightly compressed (i.e. reduced in scale), the frequency resolution decreases in the higher frequency range, while the time resolution increases. This is basically the same as the STFT, however due to the way that the decomposition is made, the frequency resolution at the lower frequencies is better than the STFT, along with the time resolution at high frequencies; this tends to be of more use in condition monitoring, as many of the specific frequencies of interest tend to be relatively low in large machines, however the high frequency components that exist tend to be noise, where high frequency resolution is not so important, more the presence and

intensity of the noise. The STFT by contrast offers constant time and frequency resolution throughout the range of its measurement.

Wavelets have been used in many different areas of condition monitoring, with applications in gear fault detection and rotating machinery diagnosis [43–46], machining processes [47], electrical machines [48] and automated packing machines [49] . A number of papers have also been published regarding the general use of wavelets in condition monitoring, as there are certain factors regarding the most suitable choice of the correct wavelets for a given application [48,50].

2.3 Genetic Algorithms

2.3.1 The Genetic Algorithm

Genetic or evolutionary algorithms can trace their development back to 1970s, with the publication by Holland of his book [51], and a Ph.D. thesis by DeJong, one of his students [52]. Most of the current work in the field of evolutionary algorithms is derived from these works. A Genetic Algorithm (GA) models the process of evolution as it takes place in nature; utilising the the process of natural selection to find the best or optimal solution to a problem without requiring any prior knowledge on the task in hand; it is for this reason that GAs are regarded as belonging to a class of "weak methods", due to the fact that the GA makes few assumptions on the problem being dealt with. GAs are most commonly used for optimisation type problems in an engineering context, as they are able to find optimal solutions to high dimensional problems relatively quickly. Space here only allows a general overview of the operation of a GA, along with all the possibilities available; a good general introduction to genetic algorithms is given in [53–56].

The simple GA, as proposed by Goldberg [55], follows the basic form shown in fig 2.2. As can be seen, the algorithm starts with an initial randomly generated population. This is evaluated using the fitness function, and each member of the population is checked to determine whether it satisfies the stopping criteria. If no member does this, then the best members of the population are selected; the selection criterion will depend on the nature of the problem in question.

Having selected the best members of the current population, these are used as the parent members of the new population. The parents undergo a process known as crossover, where portions of each of the two parents are mixed, producing two new members of the population, in the same way that male and female chromosomes are mixed during reproduction. Finally, members undergo a process of mutation, where selected portions of the genome will be changed according to random chance; this mimics the variation that occurs in nature. This fresh population is used for the next generation; none of the parents are used in the next generation, only children.

Several other types of genetic algorithm have been proposed, which allow some of the parents to be kept in a population (incremental GA and the steady state GA), and also algorithms that utilise several populations working in parallel, with cross fertilisation between the different populations (Deme GA). Space here prevents their discussion, but [57–59] cover the various different possibilities available for use at the present time.

2.3.2 The Genome

The GA has a number of components that are analogous to the components of genetic organism; the primary component of a genetic algorithm is the genome, which mimics the properties of DNA in a cell. The genome is an encoded set

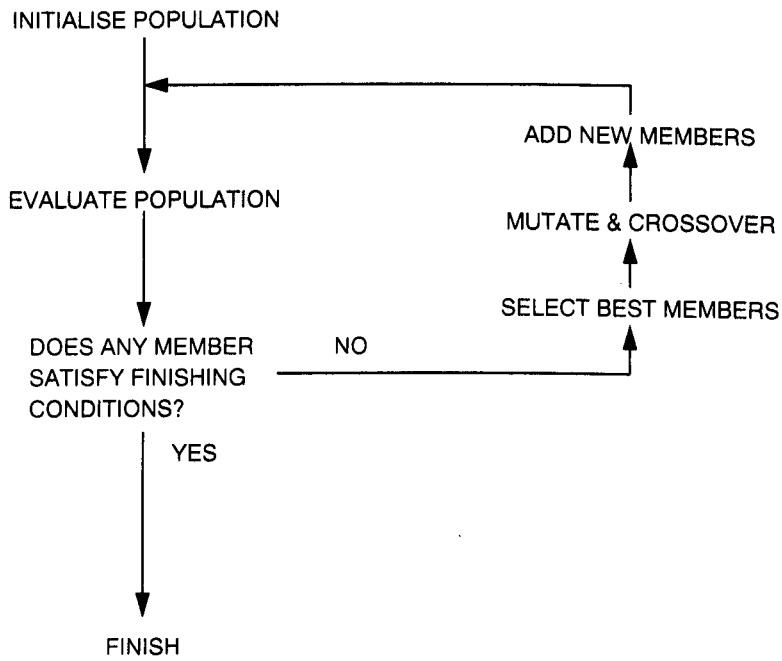


Figure 2.2: Basic form of a genetic algorithm

of instructions which the genetic algorithm will use to construct a new model or function. The encoding for the genome will vary depending upon the type of application that is required; in some cases a binary encoding (with ones and zeros representing the presence or absence of a component) will be sufficient, whilst in others an encoding composed of real or floating point numbers may be used instead.

Many different genome types are commonly in use now, including binary strings, where the genome is composed solely of 0s and 1s, real number genomes, where the genome elements can take either floating point or integer numbers are commonly used. 2D and 3D genomes can also be used, dependent on the problem, while lists, string genomes and tree genomes are also used. Genomes can also be variable length or fixed length, and in the case of tree genomes, the possibilities are much more varied. For some problems, it may also be necessary to combine different genome types to create a new form; perhaps a combination of

real and floating point numbers, or a binary and string genome. The nature of the problem will determine this, but the genome should be capable of allowing full representation of all the reasonable possibilities without allowing the creation of genomes that are impossible to evaluate; if this is impossible, then the evaluation function should be constructed in such a way as to penalise these.

2.3.3 Genetic Operators

There are two main genetic operators used in all GAs; crossover, or recombination, and mutation or perturbation. The use of these mechanisms is fundamental to the operation of the GA, and the choice of different methods for each can alter the performance of the GA dramatically, dependent upon the problem in question.

Crossover

The crossover operation occurs when the elements of two parent genomes are combined together to form two new members, in the same manner as cross-breeding occurs. In this way the child genomes created should have some characteristics of both parents. Dependent on the composition of the genome that is being used in the GA, the crossover techniques can be very different.

Each different genome has different types of mutation; for array type genomes, there is the single point crossover, where one break is made in each of the two parent genomes, and portions are swapped; this can be performed for fixed length, where the genomes are broken at the same point, or for variable length they may be broken in different places, and then recombined. There is also a multiple point crossover, and a uniform crossover, where multiple crossovers take place, on an element by element basis. For tree genomes, there is a crossover function where a whole portion of the subtree may be exchanged. The Probability of Crossover

P_C governs the likelihood that a crossover will take place on a given genome.

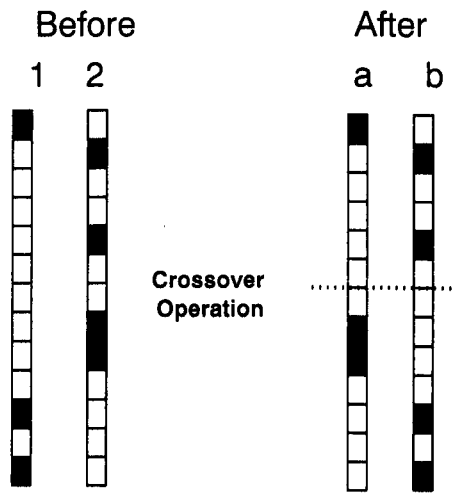


Figure 2.3: The crossover process

Mutation

Mutation is used to model the natural variation that occurs during breeding. A change due to mutation in nature tends to be small, and will not cause a major change to function of the organism in question. This applies for a natural organism, where there are many millions of elements in the chromosome, and the error rate in the natural copying mechanisms is extremely low. Correspondingly, the Probability of Mutation, P_M, is usually very small in GA problems, of the order of $P_M = 0.001$. This however is much larger than that in nature, but the error rate will be correspondingly higher, as the GA will commonly be operating on a much smaller scale of genome, and so the probability required to change a value will have to be much higher if a change is to take place within a reasonable period of time.

The manner in which mutation takes place varies, dependent upon the actual genome type in use. For binary genomes, bit flipping functions are used, while for

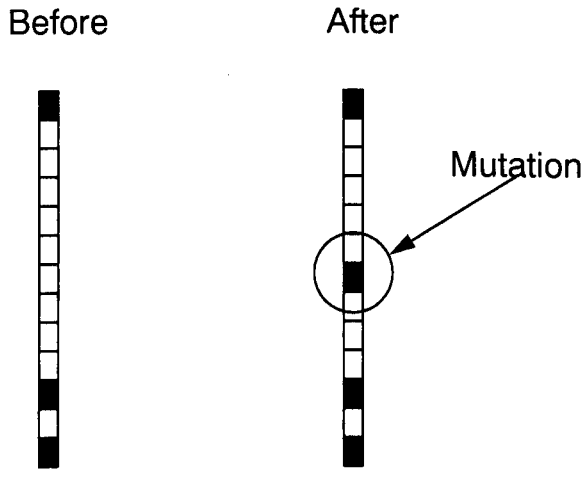


Figure 2.4: The mutation process

real valued genomes, several different possibilities exist. [57,58] give an overview of several different mutation functions, dependent upon the genome in use.

2.3.4 Selection Functions

A number of different selection functions can be used to determine which members of the population are to be used for crossover and mutation. At one end of the spectrum, there are purely stochastic techniques, such as uniform selection, where the selection is carried out on a random basis. Other techniques are also commonly used which increase the likelihood of selection on the basis of a high fitness value; roulette wheel selection and tournament selection are both examples [59] of this type of behaviour. There are several different selection techniques, however space here precludes there inclusion; [57,58] both give overviews of several different types of selection function.

2.3.5 Fitness and Objective Functions

The fitness and objective functions are the parts of the GA which determine the performance of a given genome during the evaluation stage. The objective function is used to evaluate the genome, and the score returned by the objective function may be modified by the fitness function, which might also combine the output of the objective function with other parameters indicating such desirable qualities in the solution as size or speed, that would be considerations in the overall performance of the solution rather than just the pure objective performance. The actual composition of the fitness function is as problem dependent as the genome encoding. If the fitness function is unable to adequately evaluate and rate the performance of a genome, then the GA will be unable to find an optimal solution to the problem. Hence, careful thought has to be given to how the fitness function will be calculated, and what properties the function will actively try and promote in the genome.

2.4 Feature Selection Algorithms

Feature selection is a process that can make a very real difference to the performance of a classifier network. The pattern recognition and classification process can be broken down into two sections; that of feature extraction, and the other being the actual classification process. Many different classification algorithms are available, and there are also many different ways of generating preprocessed features for the classifiers; however, there is no easy way of deciding which features will be the most useful for a given classifier, or the minimum number required to meet a task.

It is possible to generate hundreds of different features for use by a classifier, however it is much more difficult to determine which features actually impart

useful information to the classifier, and which are confusing the classifiers by providing data that is either spurious or uncorrelated with the different classifications. Feature selection provides a mechanism to determine which features are providing useful information and select them, while removing those features that are confusing the classifier, and impairing performance.

Feature selection algorithms can basically be broken down into two main types; filter approaches and wrapper approaches. Filter approaches attempt to select features on the basis of correlation between the different features, without testing the subsets on a classifier. The wrapper approach, as defined by Kohavi [60] uses the classifier to evaluate the suitability of the feature subset. This is the approach that will be discussed here. Three types of feature selection process are available for the wrapper approach [61]; complete, heuristic and random.

Complete feature selection carries out a complete search on the basis of the evaluation function used. A complete search will always find the optimal result; however it may find the optimal result in fewer evaluations than an exhaustive search would require. It should be emphasised that a complete search algorithm does not have to be exhaustive [61], although an exhaustive search will *always* be complete. The branch and bound technique is an example of a solution which is complete but not exhaustive, due to it's ability to backtrack.

Heuristic feature selection involves the use of a systematic approach or strategy towards finding an optimal combination. Examples of this include Sequential and Generalised Sequential Forward Selection, Sequential and Sequential Generalised Backwards Selection.

Random feature selection generates the features at random, with any concerted selection strategy. Examples of this type of selection include Genetic Algorithms and Synthetic Annealing

2.4.1 Branch and Bound Methods

Branch and Bound (BAB) methods are one of the oldest methods used for feature selection. BAB works on the principle of monoticity, where it is assumed that if two subsets (A and B) of features are evaluated, where $A \subset B$, then the "score" of the smaller subset will be less than that of the larger set. This takes no account of the fact that one of the inputs may provide no useful information, and diminish classifier performance by it's inclusion, and as such is flawed, however it is included here for information. BAB works by building a "tree" structure (see fig 2.5), which contains all the different permutations of feature combinations. Starting with a "root" which contains N features, $N-1$ branches are created on the next level, where each branch has had one variable eliminated from the feature set. This process continues down through several levels of the tree, where each branch of the tree eliminates one variable from the branch the level above, until the desired number of features are reached. Repetition is prevented by only allowing the elimination of values greater than the value eliminated in the level above. This means that the tree has more branches at one side than at the other. Evaluation of the tree is started from the end of the tree with the smallest number of branches (the rightmost branch in fig 2.5), and the search proceeds along the sub-branches. Assuming that the desired number of features is three, the search would start at {1,2,3}, setting the result to the best value. The search then proceeds to {1,2,4,5}, continuing down into the sub-branches only if it's evaluation is higher than that of {1,2,3}. If not, then evaluation carries on to the

next level 1 branch, and the search process again attempts to proceed into the level 2 sub-branches. Assuming that none of the level 1 branches have a higher evaluation than {1,2,3}, the selection process will terminate; otherwise the search will proceed into the corresponding level 2 sub-branches, testing all the branches from the node above. Any beating the highest value will be set to the new high value. [62,63] contains a fuller explanation of the technique in a feature selection context.

2.4.2 Sequential & Generalised Sequential Forward Selection

Sequential Forward Selection (SFS) is performed by evaluating the classification success of every feature from the feature set on an individual basis. Having evaluated all individual features, the best feature is kept, and then evaluated in combination with all the remaining features, repeating the procedure until the desired accuracy or number of features are achieved. Generalised Sequential Forward Selection (GSFS) works along similar lines, although rather than starting with one feature, a subset of g features is the starting the point, and a number of features are added at a time. The algorithm tests all different possible combinations of r features, meaning that $(N-g)/r$ additions have to be evaluated. This can be much more computationally demanding than SFS, as the classifiers are correspondingly larger.

One of the major drawbacks of SFS is that by adding features on an individual basis, then the selection method takes no account of any correlations between different features in the dataset, and is unable to take advantage of this in any way. GSFS is an attempt to work around this by adding groups of features at once, and trying to find the interactions between them. However, the strategy

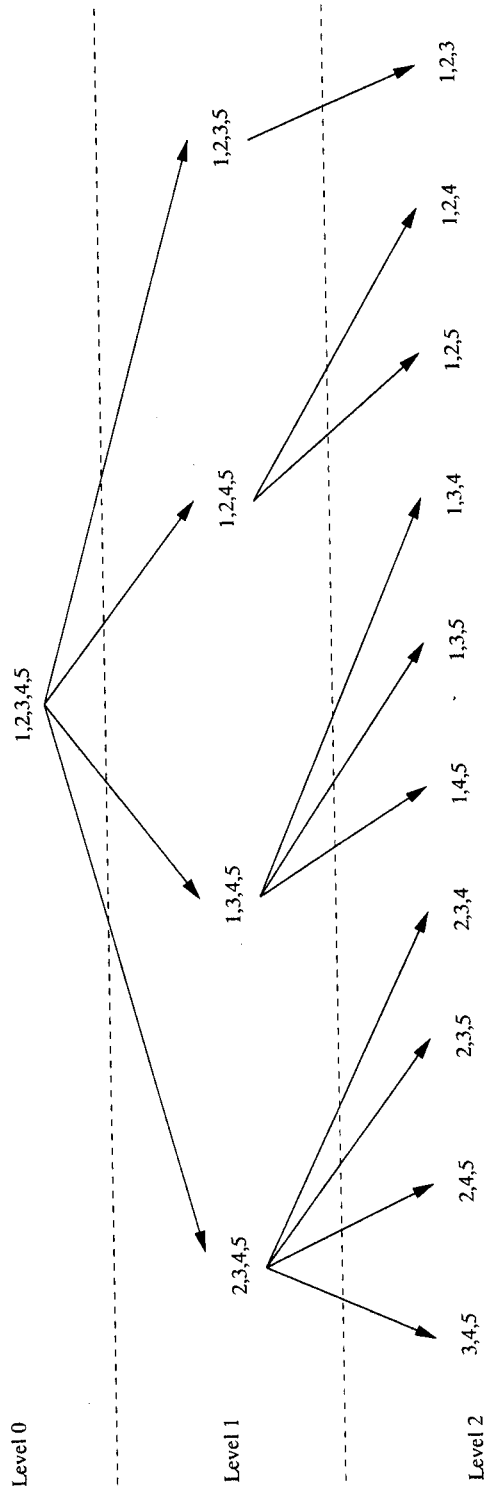


Figure 2.5: The Branch and Bound (BAB) method

can be very slow when the number of possible features increases, or if features are added in small increments.

2.4.3 Sequential & Generalised Sequential Backwards Selection

Sequential Backwards Selection (SBS) is a very similar idea to SFS, however, rather than starting with no features, the algorithm starts with the full feature set, and removes features one at a time, testing all features individually and selecting the one which causes he greatest drop in performance when it is included in the featureset. Generalised Sequential Backwards Selection (GSBS) is analogous to GSFS, although (like SBS) starting with the full set of features, and removing them in groups. This is also much more computationally intensive than SBS, although like GSFS, it can produce better results as it allows the relationships between different groups of features to manifest themselves, and allows the classifier to exploit them.

2.4.4 Principal Components Analysis

Principal Components Analysis (PCA) was first proposed in 1901 [64], and has become very widely used in the ANN community for dimensionality reduction. In basic terms, PCA can be thought of as a rotation, or transformation, of the axes on which data is originally presented to new axes which maximise the variance of the points in feature space. After transformation, fewer dimensions may exist than was originally the case, as any dimensions which were not independent will be eliminated (i.e have zero values). This can be used to improve the separability of different classes of data. The transformation uses linear combinations of the original axes to produce a set of new independent axes, which are orthogonal in

all dimensions, but rotated in feature space.

PCA can simply be carried out by calculating the eigenvalues of the covariance matrix of the data [63, 65]. Reordering the eigenvalues in terms of magnitude, and multiplying the original data matrix with the eigenvalue matrix transforms the data onto the new axes. In the case where one or more of the eigenvalues is small or close to zero, it is possible to remove the corresponding rows from the matrix, and reduce the dimensionality of the data without removing a significant part of the information content. This allows data to be transformed easily, as any new data can also be transformed onto the same axes by multiplying by the eigenvalue matrix. One cautionary note exists with PCA however; through the use of the transformation matrix, the original number of features still have to be calculated, so if the desire is actually to *reduce* the number of inputs to the classifier, or reduce the *amount* of preprocessing required, then PCA does not satisfy this requirement.

2.5 Summary

This section has provided an overview of the historical developments of some of the techniques in the field of condition monitoring. A short review of probability theory and statistics was made, leading into the definition of some of the different statistical measures used in this thesis. Some of the commonly used performance measures were examined, and spectral, higher order spectral and time-frequency analysis were also examined. An introduction to the general principles of GAs was made, and a review of some other feature selection techniques were also covered.

Chapter 3

Experimental Data, Sampling and Feature Extraction

All of the work contained in this thesis has been performed on raw vibration time-series data taken from two machines. This chapter describes the two different machines that suffered "damage", and gives an overview of the different characteristics of the fault types.

3.1 Data Sources

3.1.1 Bearing Faults

The work presented in this thesis is based around experimental results performed on vibration data taken from two test rigs, both of which can be fitted with a number of interchangeable faulty roller bearings. This is used to represent the type of problems that can commonly occur in rotating machinery. Rolling element, or ball bearings are one of the most common components in modern rotating machinery (see figure 3.1); being able to detect accurately the existence

and severity of a fault in a machine can be of prime importance in certain areas of industry, as in many cases the machine may be safety or emergency related. Ball, roller, and needle bearings are widely used in many applications that have low to medium speed, and light to relatively high duty ranges. They are not used in extremely heavy duty, or high speed applications, as other bearing types such as journal bearings can serve this better.

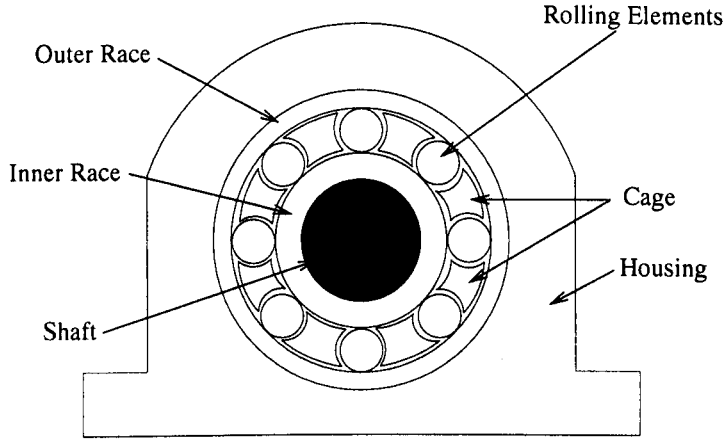


Figure 3.1: A typical roller bearing, showing different component parts

A rolling element bearing consists of a number of components: the inner race, which is normally fixed to the shaft; the outer race, which would normally be placed in a bearing housing of some description; the rolling elements themselves, of which the number will vary, dependent upon factors such as the degree of bearing loading and the metallic composition of the elements (i.e. softer metals will require more balls to spread the load evenly without suffering deformation). The last major component is the cage, which is commonly made from either plastic or metal, and is used to separate the elements equally about the bearing. This is very critical to the operation of the bearing, as if the elements become unequally distributed about the bearing then serious, and even catastrophic failure may occur, as during certain parts of the cycle of rotation, there will be no load-bearing elements available to support the weight of the shaft, and distortion and damage

may result.

Damage normally occurs to bearings through the course of operation. Inadequate lubrication, the ingress of dirt and grit into the raceways, vibration/impact in the raceways, excessive loading, misalignment, and even excess lubrication can all be factors in the failure of a bearing. Faults can occur in each of the four main components, giving rise to four different faults.

3.1.2 Machine A

Machine A was a small vibration test rig loaned to the University of Strathclyde by Weir Pumps Ltd. of Cathcart, Glasgow. The machine (see figure 3.2) is approximately 1m long, and consists of a one inch thick, mild steel bedplate, to which are fixed two bearing blocks, and a 12V DC electric motor (variable speed). Drive from the motor is carried to the shaft by means of a small flexible coupling, which can accommodate some limited radial and axial misalignment.

The shaft consisted of silver steel, 1/2" in diameter, to which were fixed two RHP 1017-1/2 bearings, using grub screws. These were seated in the bearing housings, and the receptacles are radiused in order to allow the bearings to swivel in the housings, taking up a certain degree of radial misalignment. The bearings are secured to the shaft by means of two grub screws, rather than a machined key, but this has sufficient grip for such a small diameter shaft under relatively low loading. One of the bearing blocks was spot faced, drilled and tapped to accept the mounting of two accelerometers, mounted perpendicular to each other. Unfortunately, it was impossible to have the accelerometers mounted on the vertical and horizontal axes, however they are mounted at approximately 10° to the vertical, and the actual vertical and horizontal movements can be calculated by simple trigonometry.

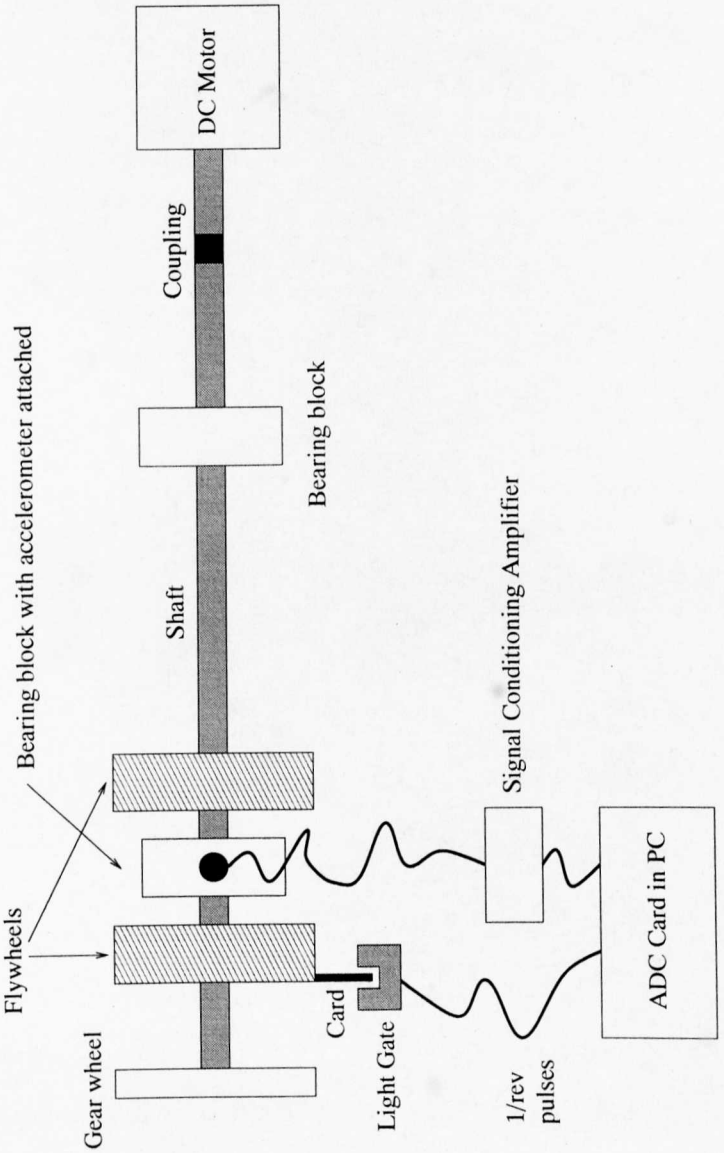


Figure 3.2: The machine test rig used in experiments

"Damage": Induced Faults

Due to the time-scale of the research project, there was insufficient time available to create bearings with faults that had been derived under actual running conditions. Instead, faults were induced in the bearings for machine A, under workshop conditions.

Inner Race Fault

The inner race fault was created by removing first the cage, moving the elements to one side of the bearing, and removing the inner race. A groove was then cut in the raceway of the inner race, using a small grinding stone, and the bearing was reassembled.

Outer Race Fault

The outer race fault was created by removing the cage, pushing all the balls to one side, and then inserting a small grinding stone, and cutting a small groove in the outer raceway.

Rolling Element Fault

The rolling element fault was induced by using an electrical etcher to mark the surface of one of the balls, simulating corrosion. This gave what was probably the most pronounced of the faults, as the etcher increased the diameter of the ball slightly, and this made for tighter clearances than would usually occur.

Cage Fault

The cage fault was simply created by removing the plastic cage from one of the bearings, and cutting away a section of the cage, so that two of the balls were

free to move, and not held at a regular spacing, as would normally be the case. the effect of this was fairly minor at the loading that the bearing was put under, and as a result, this makes the fault relatively difficult to distinguish from the normal condition.

3.1.3 Machine B

The data for machine B was supplied by Dr. Keith Worden of the University of Sheffield, and taken in Poland by Prof. C. Cempel and Mr. M. Tabaszewski, both of the Poznan Technical University, who recorded the data. Machine B employs slightly larger bearings than machine A, with a shaft diameter of 20mm, rather than the 12.5mm of machine A. The machine consists of a drive unit, connected to a shaft supported by two bearings, type 6204. The shaft was loaded by means of a large flywheel. One of the two housings holds the faulty bearing, and an accelerometer was mounted vertically on the bearing housing.

Fault Conditions

Four fault conditions exist for machine B, along with one normal condition. These are: a broken outer race (i.e. outer race fault), a broken cage, with one element loose, a damaged cage with 4 elements loose (both cage faults), and a bearing with a badly worn ball (i.e. Rolling Element Fault).

3.2 Sampling and Data Acquisition

3.2.1 Machine A

Signal conditioning was carried out using Bruel and Kjær charge amplifiers, which fed the acceleration output direct to the ADC. A low pass filter in the ADC card

had a cutoff frequency of 18.3kHz. Sampling on machine A was carried out using a Loughborough Sound instruments DSP32C card, fitted with dual ADCs, which recorded the output from the accelerometers; a daughterboard connected to the card through the DSPlink2 interface was used to sample the one pulse per revolution signal. Sampling was carried out on the two vibration channels and the once per pulse signal simultaneously, taking place at a frequency of 48kHz, for a period of 40 seconds.

The machine was run at a series of different speeds (sixteen in total), and ten samples were taken at each speed. This gave a total of 160 examples of each condition, and a total of 960 raw datafiles to work with.

3.2.2 Machine B

Sampling for machine B was carried out using a Bruel and Kjær analyser. Data was sampled at a rate of 16.384 kHz, running at a constant speed. For the five fault conditions, between ten and twelve examples of each condition were recorded, giving a total of 56 datafiles to work with.

3.3 Source Data

3.3.1 Machine A

Six different conditions are used within the experiments conducted for this machine. Two normal conditions exist: one bearing in a brand new condition (NO), while another is a bearing in slightly worn (NW) condition. There are four fault conditions:- inner race fault (IR), outer race (OR) fault, rolling element(RE) fault, and a cage fault (CA).

Each of the fault conditions has different characteristics, with the inner and

outer race faults having a periodic signal; the rolling element fault may or may not be periodic, dependent upon several factors including the degree of damage to the rolling element, the loading of the bearing, and also the track that the ball describes within the bearing. The cage fault tends to exhibit a random behaviour, again dependent upon the degree of damage and bearing loading.

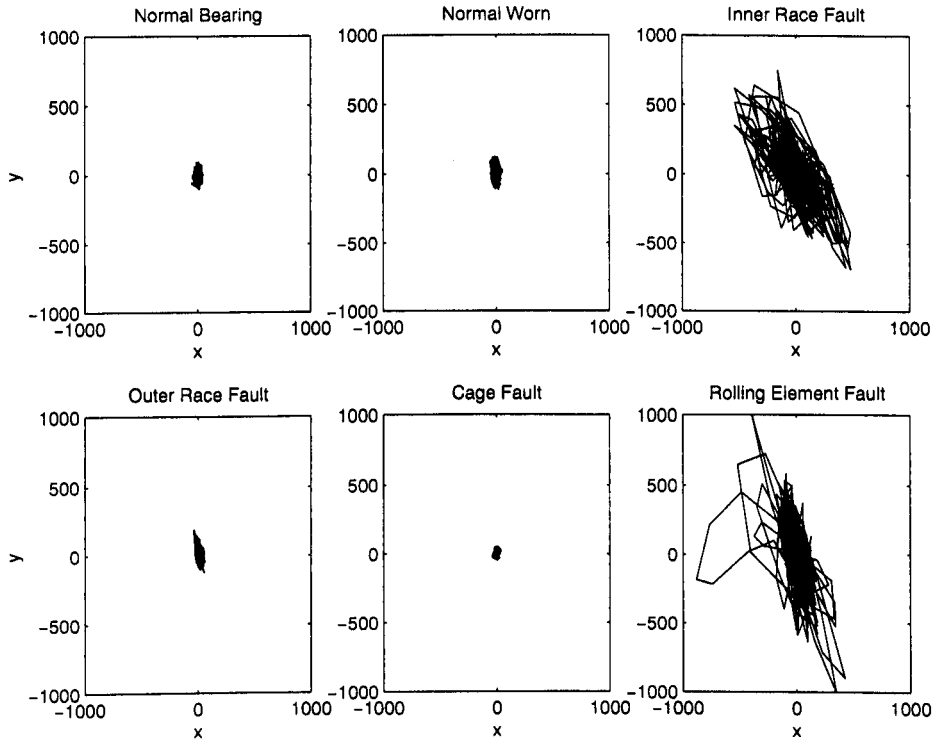


Figure 3.3: Machine A: Acceleration orbit plots for the six different conditions.

Figure 3.3 shows the acceleration orbit plots for the six different cases. As can be seen, the two normal bearings have a fairly similar profile, both occupying an elliptical region in the acceleration space. Understandably, the vertical acceleration is greater than the horizontal, although the normal bearing describes a slightly smaller ellipse than that of the worn bearing; again this would be consistent with slightly larger clearances on the worn bearing. The inner race fault has vertical and horizontal components that are much larger than those of the two normal bearings. The much greater acceleration here would be accounted for by

the the impact of the element against the defect on the inner race of the bearing, with the resultant bounce as the element strikes the defect, and rebounds into the outer race. It is interesting to note that the distribution in acceleration space is once again elliptical, although this time the ellipse is skewed. The outer race fault also seems to show some mild skewness, but the magnitude of the acceleration is much more like that of the two normal bearings. The main obvious difference between the normal bearing plots and that of the OR is that the OR plot has a slightly "spikier" appearance than that of the normal cases, with the spikes appearing along an axis, that would correspond to the radial location of the fault on the outer race. The cage fault has the smallest magnitude of all the different signals, producing a very localised and almost circular plot. This is in marked contrast to that of the two normal bearings. The rolling element fault shows the most distinctive characteristics of all the different conditions. The plot is skewed, but at a larger magnitude than any of the other conditions; additionally, it can be seen that the plot is not localised in the same way as the other conditions. The RE fault describes a rather more erratic course than the other conditions, and this is due to a large part to the rotation and twisting action of the ball as it moves around the raceways. The twisting action means that the damaged portion of the element will only come into contact with the load carrying parts of the bearing at certain times, and as a result, this means that the fault may appear erratic.

Examining the actual vibration plots on a time series basis (figure 3.4), again, a number of different characteristics are obvious.

Visually, four of the signals seem quite similar; the two normal conditions look similar, although the worn normal bearing seems to be a bit noisier than the new normal bearing. The outer race fault, and cage fault also appear to be very similar to the normal conditions, in terms of magnitude, and also noise. It would

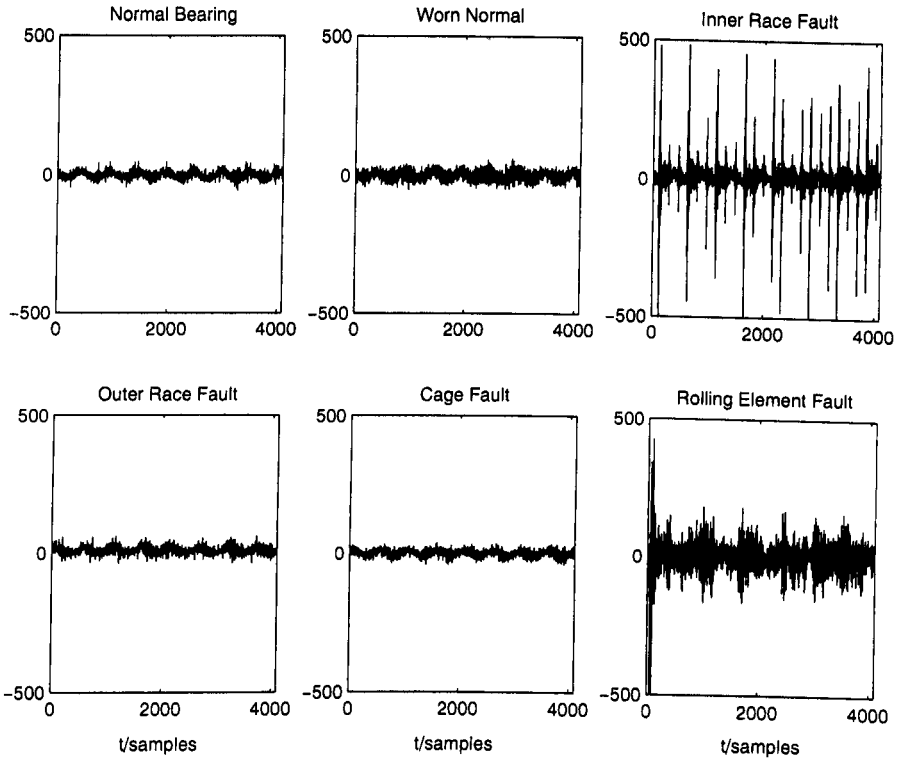


Figure 3.4: Machine A: Vibration plots for the six different conditions.

be difficult to determine accurately which condition was which, solely by means of the vibration signals. The inner race fault is very easy to distinguish, by means of the long spikes present in the signal, as is the rolling element fault, which is discernible due to the the large magnitude fluctuations. It is also interesting to note that the rolling element faults is not entirely periodic, although this may be explained by the way that the ball rotates in the raceways, spinning both in the direction of travel around the shaft, and also (dependent upon any thrust action along the shaft) in the direction of the shaft axis. This gives the elements a twisting, spinning motion that means that the defective portion of the element may not always make contact with the load-bearing portion of the raceways on a once per revolution (of the element) basis.

Figure 3.5 shows the histograms of the raw vibration signals. Of the three plots (figures 3.3 - 3.5), this plot allows the different conditions to be distinguished more

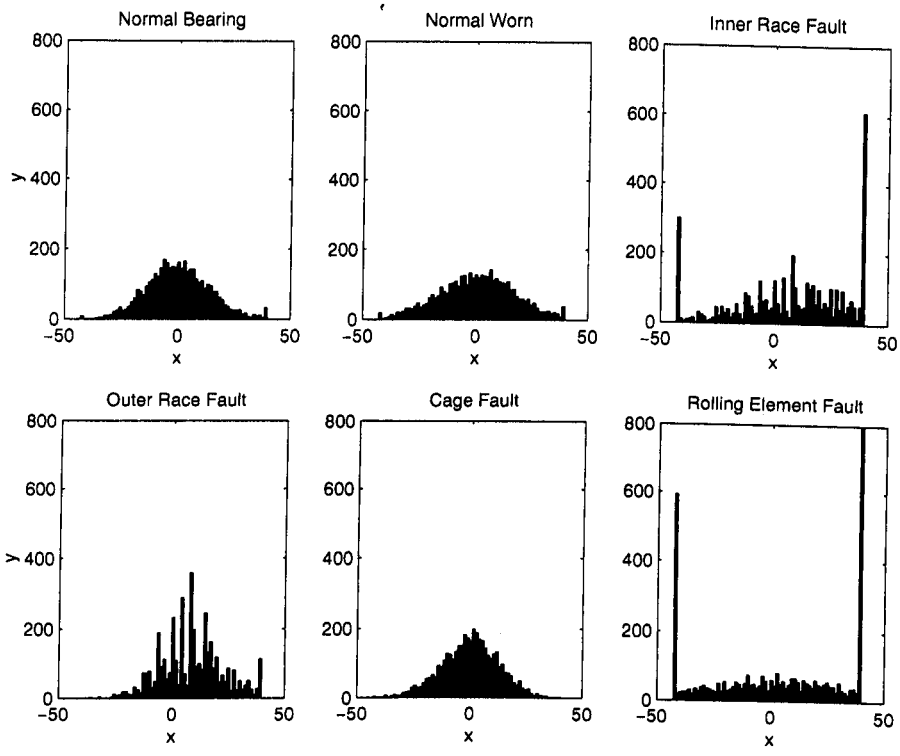


Figure 3.5: Machine A: Histograms for the six different conditions.

easily. As can be seen, by examining the two normal conditions, it is possible to distinguish between the values of the histograms around the centre bins, with the normal bearing having a slightly higher magnitude around the centre bins than the worn normal bearings. The worn bearing also shows slightly longer "tails" on the distribution as well, making it easier to distinguish. The Outer race fault is now easy to distinguish, as the distribution has a high magnitude about the centre of the distribution that is markedly larger than that of any of the other three conditions that there was confusion over in the vibration plots. The cage fault is slightly harder to distinguish, appearing very similar to the normal condition, although the distribution appears more even than for the normal condition. The inner race fault is relatively easy to distinguish due to the extremely large values in the tails of the distribution, caused by the spikes that the fault generates. In the same way, the rolling element fault is also fairly easy to distinguish, as

the magnitude of the outside edges of the tails are extremely large, while the distribution about the centre is fairly small and flat, characteristics that are relatively simple to detect.

3.3.2 Machine B

Examining figure 3.6, the form of the vibration signals for the five different conditions can be seen. The normal condition has the smallest magnitude of vibration, although two of the other conditions, namely the bearing with the worn element, and the bearing with 4 elements lose, exhibit raw vibration characteristics which might make them comparatively difficult to distinguish between. The cage fault with one loose element has a larger noise component than the other three, and so this stands out slightly from the normal, cage(4), and worn element bearings, but it may still prove difficult to spot. The outer race fault is the easiest condition to distinguish by eye, as the regular spikes caused by the elements hitting the defect are of a much larger magnitude than any of the other signals; additionally, none of the other signals have the same amount of noise present in them. Training the classifier to spot the outer race fault should be comparatively easy; distinguishing between some of the other conditions may prove much harder.

The vibration plots provide a visual indication of the different signals that are received; however, this does not say a lot about how the statistical properties of the signals will compare; by examining the histograms of the different signals, it is possible to see the different distributions of the signals. Figure 3.7 shows the histograms of the signals in figure 3.6, and from this it is possible to see much more clearly how the different signals compare.

As can be seen, it is a simple matter to distinguish between the normal (NO) bearing, outer race (OF) fault, and the other three conditions. It is however much

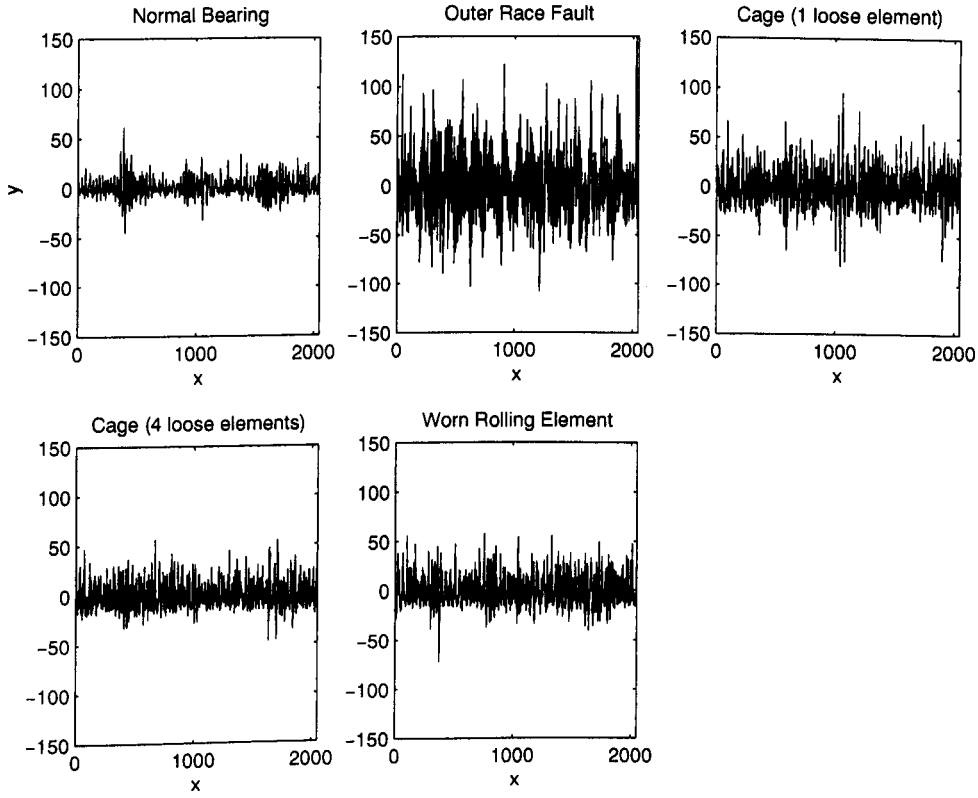


Figure 3.6: Machine B: Raw Vibration plots for the five different conditions.

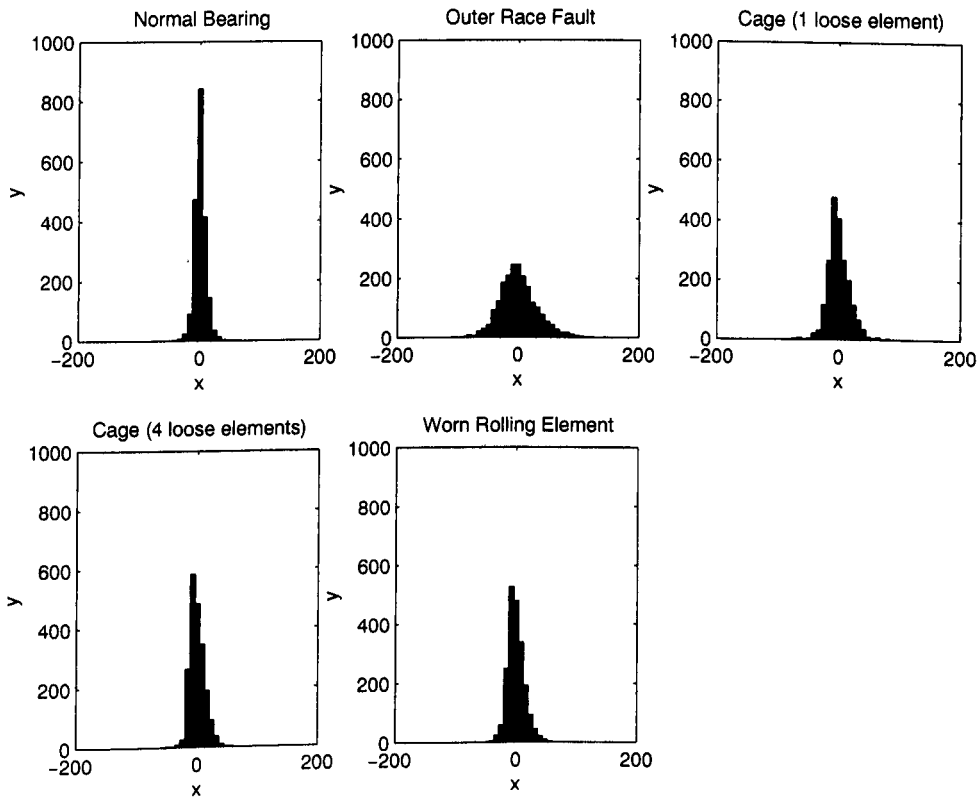


Figure 3.7: Machine B: Histograms of the raw vibration for the five different conditions.

more difficult to distinguish between the two cage faults (CA1 and CA4), and the worn (WO) rolling element, as the histograms appear very similar. This will make things difficult for the classifiers to characterise the different conditions. It should be possible to characterise the fault/no fault case, but a lot more difficult to successfully characterise between the different fault conditions.

3.4 Feature Extraction

The feature extraction stage of the pattern recognition process is one of the most important stages in the pattern recognition process. The purpose of feature extraction is twofold; firstly, feature extraction is an attempt to reduce the dimensionality of the data presented to the classifier, without diminishing the content presented in the data. Secondly, feature extraction is intended to turn raw data into information, that the classifier can use more usefully. The vibration data as sampled will consist of several hundreds, if not thousands of data points. To train a classifier to deal with this data will require a very large training time, and also make a network extremely complex; it will also make the generalisation of the network fairly poor, as so many input factors will make it difficult for the classifier to determine useful relationships between inputs, and consequently, to generalise effectively. This effect, known as the curse of dimensionality, can be dealt with using a number of techniques, however, feature extraction is one of the simplest and most effective.

3.4.1 Machine A

Having sampled the data as described in 3.2, a number of different forms of preprocessing were used as shown below. Part of the emphasis in attempting to use different feature extraction techniques was to investigate the effect of the

different techniques on the classifier performance. This is examined in more detail in later chapters.

Plain Statistics

A number of different statistical features were taken based on the moments and cumulants of the vibration data. Higher order spectra have been found to be useful in the identification of different problems in condition monitoring [29]. A good introduction is given in [17, 25]. For each of the basic preprocessed signals, a set of eighteen different moments and cumulants were calculated. The format of the values is shown in equation 3.1. The cumulants are defined in equations 2.13 - 2.16.

As two dimensional information exists for vertical and horizontal movement, it was decided to calculate values based upon the the equivalent polar positions, and values calculated on this basis are given the subscript z, defined in equation 3.2.

$$\begin{bmatrix} m_{\mathbf{x}}^{(1)} & m_{\mathbf{y}}^{(1)} & C_{\mathbf{x}}^{(2)} & C_{\mathbf{x}\,\mathbf{y}}^{(1)(1)} & C_{\mathbf{y}}^{(2)} & C_{\mathbf{x}}^{(3)} & C_{\mathbf{x}\,\mathbf{y}}^{(2)(1)} & C_{\mathbf{x}\,\mathbf{y}}^{(1)(2)} & C_{\mathbf{y}}^{(3)} \ldots \\ C_{\mathbf{x}}^{(4)} & C_{\mathbf{x}\,\mathbf{y}}^{(3)(1)} & C_{\mathbf{x}\,\mathbf{y}}^{(2)(2)} & C_{\mathbf{x}\,\mathbf{y}}^{(1)(3)} & C_{\mathbf{y}}^{(4)} & m_{\mathbf{z}}^{(1)} & C_{\mathbf{z}}^{(2)} & C_{\mathbf{z}}^{(3)} & C_{\mathbf{z}}^{(4)} \end{bmatrix}^T \tag{3.1}$$

$$z = \sqrt{(x^2 + y^2)} \tag{3.2}$$

This was applied to the sampled vibration data, and the results were saved in a 18×960 matrix.

Signal Differences and Sums

In an attempt to highlight both high and low frequency features, it was decided to apply simple calculations that would emphasise the high and low frequency content of the raw signals. The differences will increase in value wherever high frequency changes take place, while the sums of the signal will help to emphasise the low frequency content of the signal. This is useful in the context that several of the fault conditions exhibit "spikes" as characteristics of the balls striking the defects; taking differences will hopefully accentuate this effect among the signals containing spikes, and allow them to be easier to characterise simply. Equally, the summation should also allow those conditions with a greater proportion of lower frequency noise to be characterised more easily.

Taking the original vibration signals, the derivative of each vibration signal was calculated according to equation 3.3, and then the statistical parameters given in equation 3.1 were calculated from the modified signals, and saved in another 18×960 matrix. This process was then repeated using the integral of the vibration signal (calculated according to equation 3.4, where m_x represents the mean value of the sequence x), creating yet another 18×960 matrix.

$$d(n) = x(n) - x(n-1) \tag{3.3}$$

$$i(n) = \{x(n) - m_{\mathbf{x}}^{(1)}\} + i(n-1) \tag{3.4}$$

High and Low Pass Filtering

Much like the signal differencing and summation, there is much useful information contained in the high and low frequency bands of the spectrum. Rolling elements

have a natural resonant frequency that can be relatively high, and this can be used to detect faults, as the balls will resonate when striking a defect [66]. By using high pass filtering to remove the low frequency content of the vibration signal, it becomes much easier to locate and detect these resonant peaks, and use them for defect detection and classification. Low frequency content of the signal will also contain useful information about the signal, that may also be obscured by the high frequency content of the signal (particularly in the case of those conditions that create lots of spikes, and it can be difficult to determine what is happening in the lower frequency bands.

The signals were passed through an eighth order Butterworth IIR high pass filter with a cut off frequency of 129 Hz; the statistical values of equation 3.1 were then computed, and the results saved in a 18×960 matrix. The process was then repeated using a low pass filter with the same cut-off frequency, creating another 18×960 matrix.

Spectral data

Spectral data has been one of the most effective forms of feature extraction used in condition monitoring. As many of the machines monitored are rotational, many of the faults that exhibit themselves are frequency related. Where the construction of these machines is also known, it is a comparatively simple matter to calculate the frequencies at which certain defects would be likely to occur. However, to read frequency plots, identify harmonic peaks, and give confident diagnoses of problems is a skilled task, and requires experience. Spectral information is still very useful for providing information for classifiers, and as a result, it was used as one of the methods of preprocessing here.

For each of the two channels sampled, a 32-point FFT of the raw data was carried out, and 33 values were obtained for each channel. These were then stored

as a column vector of 66 values, which was used as the input data set for the given data sample. The full input data set formed a 66×960 data set.

Target data

For each given vector in the input datasets, a corresponding vector was created in a second matrix containing the target information used during the neural network training. This information reflected the actual condition of the machine. This then gave, for the six conditions, a 6×960 matrix containing the target data. This is all the input information assembled to train the networks. A second matrix was created with only two categories, containing fault/no fault classification.

3.4.2 Machine B

The second data set contains vibration data measured on one axis only. This reduces the number of statistical possibilities that may be used for pre-processing the data. In this case, it was decided to use only spectral processing on this data set. The fact that the data was sampled at the frequency of 16.384 kHz means that a lot of the higher frequency information that was contained in the vibration signal for machine A will not be present in that for machine B, and so the possibilities for doing different forms of preprocessing were further reduced.

Spectral Features

Spectral features were generated by taking a simple 32 point FFT of the data, averaged over the length of the data (2048 points). This was then placed in a 33×56 matrix.

Target data

Once again, two matrices were created, consisting of a 2×56 matrix, for the fault/no fault case, and a 5×56 case, for the five class problem.

3.4.3 Normalisation

Prior experience (for example, see [3]) with training neural networks had indicated the significance of normalisation to both the speed and success of training. Prior to commencing the training run for the neural network, all the data in the input data set was normalised on a row by row basis. Rows were normalised according to the formula:

$$x_i = \frac{(x_i - m_{\mathbf{x}})}{\sigma_{\mathbf{x}}} \tag{3.5}$$

where $m_{\mathbf{x}}$ is the mean value of the row vector $\mathbf{x}$, and $\sigma_{\mathbf{x}}$ is the standard deviation of the row vector $\mathbf{x}$.

3.5 Summary

This chapter has discussed the two different machines used as data sources for the experiments carried out later in the thesis, methods of data acquisition, and preprocessing the data for use by the classifiers. A short discussion of some of the characteristics of the different data was also given, showing some of the different characteristics of the data that are obvious visually.

Chapter 4

Artificial Neural Networks

One of the biggest problems in MCM is actually diagnosing the fault. Vibration experts are needed to examine spectral plots, and a fairly high degree of skill and experience is required. The increasing reliance on remote management systems also means that some automatic way of diagnosing and characterising a problem is attractive in these circumstances, as no-one has to be sent out to examine equipment. Pattern recognition and classification algorithms fit these needs very well; however, it is often the case that traditional statistically based pattern recognition techniques cannot adequately deal with some of the problems that the MCM arena creates. Probably the largest of these is the (general) lack of availability of training data. This can make it extremely difficult to use statistical indicators reliably, as these require large quantities of training data in order to use them successfully.

The nonlinear properties of ANNs makes them ideal for applications such as machine condition monitoring, where the training data is very often relatively sparse, yet the network will have to generalise to a certain extent. Several applications have demonstrated that a neural network has successfully recognised

and classified different faults in a variety of different condition monitoring applications [3–8]. A good general introduction to Neural Networks is provided by Haykin [65] and also Rojas [67].

4.1 Basic Principles

In this chapter, the performance of three different types of ANN are considered; the Multi Layer Perceptron (MLP), Radial Basis Function network (RBF), and Probabilistic Neural Network (PNN). Each of the three networks takes a different approach to the way in which they partition the data into separate classes. Fig 4.1 compares the ways in which the three ANNs characterise the feature spaces in which they work.

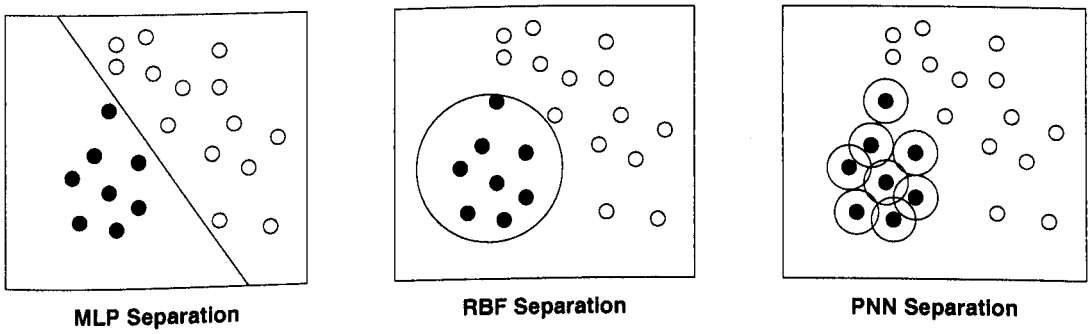


Figure 4.1: The different separation methods of MLP, RBF and Probabilistic neural Networks

As can be seen, each technique uses a different approach to partitioning the feature space in order to make a classification. The MLP creates partitions that use a series of linear separators to create a classification region. The RBF function attempts to enclose clusters of data points using a hypersphere (which corresponds to a circle in two dimensions, centring the hypersphere at the centroid of the cluster of data points. If the RBF is unable to satisfactorily enclose the data with one hypersphere, then more are added as necessary, until the error falls below a

predetermined level. The PNN, rather than attempt to find one hypersphere, creates a series of hyperspheres, one around each data point in the training set, with the user specifying the desired radius of the hypersphere. The spheres converge to create a partitioned volume in feature space.

4.1.1 The Multi Layer Perceptron

The Multi Layer Perceptron (MLP) is probably one of the most common realisations of neural networks in use today. The MLP is composed of a number of neurons, which all share the same general form:

$$y = \varphi(v(\mathbf{x})) = \varphi\left(\sum_{i=1}^{N} w_i x_i + b\right) \tag{4.1}$$

where the neuron has N inputs. There is a weight w_i for each input x_i, and one bias term b which can be used to offset the sum of the inputs. y is the output of the neuron. The MLP used in this case consists of one hidden layer and an output layer (see fig 4.2), the hidden layer having a logistic activation function (equation 4.2), whilst the output layer uses a linear activation function (equation 4.3). Other configurations were tested during initial stages of the work, using different numbers of hidden layers and also different activation functions, however, the configuration above was found to give the best results for the data.

$$\varphi(v) = \frac{1}{1 + \exp(-v)} \tag{4.2}$$

$$\varphi(v) = v \tag{4.3}$$

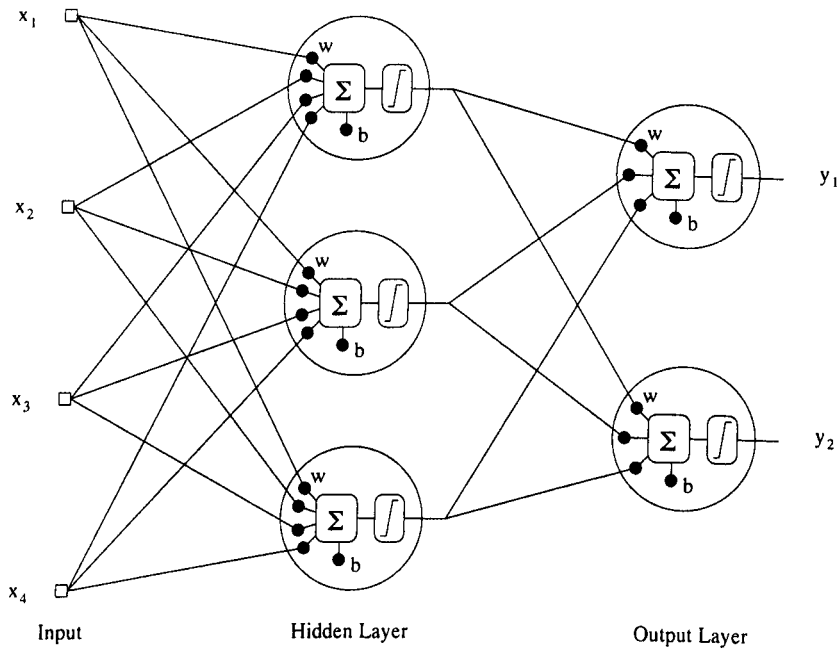


Figure 4.2: Basic structure of an MLP

where v is the sum of the weighted outputs (equation 4.1). The size of the hidden layer is very important to the operation of the neural network, and particularly it's generalisation properties. If the hidden layer is too small, then the network will find it extremely difficult to partition the data into different classes, and the performance will be extremely poor. If the hidden layer is too large, then the network begins to learn too much of the specific details of the training set (i.e. the network becomes *overtrained*), and the generalisation of the network will be extremely poor, with the MLP only responding correctly to the training examples it has already seen. The size of the output layer is determined by the number of outputs required, with one output for each classification.

Training

Training of the network is carried out using a back-propagation algorithm with adaptive learning and momentum, and the network is trained using one third of

the data set as training data, and the remaining two thirds split equally between the test and validation sets. Training is terminated by means of the validation set, when the performance of the validation set begins to deteriorate rather than improve.

The backpropagation algorithm, proposed by Rumelhart *et al* and Werbos independently [68,69], works by using the error values that occur during training to determine what adjustments should be made to the weights of the neuron. The normal measure of the error after a forward pass is to calculate the instantaneous error of the network, $\mathcal{E}(n)$ (equation 4.4).

$$\mathcal{E}(n) = \frac{1}{2}\sum_{q=1}^{C_o} e_q^2(n) \tag{4.4}$$

The instantaneous error is composed of the error at each of the nodes of the output, $e_q(n)$.

$$\begin{aligned} e_q(n) &= d_q(n) - y_q(n) \\ &= d_q(n) - \varphi_q\left(v(n)\right) \end{aligned} \tag{4.5}$$

where $d_q(n)$ is the desired target value, and $y_q(n)$ is the actual output of the neuron. For a given instantaneous error value, it is possible to calculate a corresponding weight adjustment:

$$\frac{\partial \mathcal{E}(n)}{\partial w_{qp}(n)} = e_q(n)\frac{\partial e_q(n)}{\partial w_{qp}(n)} \tag{4.6}$$

where:

$$\frac{\partial e_q(n)}{\partial w_{qp}(n)} = \frac{\partial e_q(n)}{\partial \varphi_q(n)} \frac{\partial \varphi_q(n)}{\partial v_q(n)} \frac{\partial v_q(n)}{\partial w_{qp}(n)} \tag{4.7}$$

$$\frac{\partial e_q(n)}{\partial \varphi_q(n)} = -1 \tag{4.8}$$

$$\frac{\partial \varphi_q(n)}{\partial v_q(n)} = \varphi'(v_q(n)) \tag{4.9}$$

$$\frac{\partial v_q(n)}{\partial w_{qp}(n)} = y_p \tag{4.10}$$

Combining these different expressions gives an expression for weight update in the weights of the output layer of neurons of a network.

$$\frac{\partial \mathcal{E}(n)}{\partial w_{qp}(n)} = -e_q(n)\varphi'(v_q(n))y_p \tag{4.11}$$

$$= -\partial_q(n)y_p \tag{4.12}$$

This gives a generalised expression for the weight update carried out during back-propagation:

$$\Delta w_{qp} = \eta\, \partial_q(n) y_p(n) \tag{4.13}$$

where η is the learning rate, $\partial_q(n)$ is the local gradient, which varies dependent upon the position of the neuron within the network, and $y_p(n)$ is the input p to the neuron. The values of $\partial_q(n)$ for neurons in the output (q) and hidden (p)

layers of a two layer network are:

$$\partial_q(n) = e_q(n)\varphi'(v_q(n)) \tag{4.14}$$

$$\partial_p(n) = \varphi'(v_p(n)) \sum_{q=1}^{C_o} \partial_q(n) w_{qp}(n) \tag{4.15}$$

A more comprehensive derivation of backpropagation can be found in [65].

Backpropagation is a gradient descent algorithm, and provides a relatively quick way of finding a minimum in space; however it is not always the case that the minimum found is the actual global minimum. It is common to find that the training algorithm becomes stuck in a local minimum; to reduce the likelihood of this happening, several different strategies were developed; a momentum term was added, which attempts to cause the hill descent algorithm to oscillate about the local minimum, and pushes the search out of the local minimum [70]. Another strategy which has also proved useful is to vary the learning rate of the training algorithm [71]. The algorithm attempts to keep learning rate as high as is possible. If the error after modification increases beyond a pre-defined limit, then the changes are discarded, the learning rate is decreased, and the process repeats until modification is found that effects gradient descent (rather than ascent). This allows the learning algorithm to take as large steps as are possible, and decreases the search time required to reach the global minimum. When momentum and adaptive learning are combined in the one algorithm, this makes a very robust training algorithm, and this has been used for all training of MLPs in this thesis. Backpropagation is not the only training algorithm used for MLP training, as there exists another series of approaches known as quasi-newton [72], which uses the second-order derivatives to calculate the gradient descent, and can be between 10 and 100 times faster at training MLPs [73], requiring significantly fewer steps to reach the global minimum. However, these methods can require large amounts

of memory, and as a result, it was decided to use backpropagation based training.

MLPs have found wide use in MCM applications; they are probably the most common type of ANN based classifier used in this type of application. Examples of use of the MLP in MCM include [3,5,7]

4.1.2 The Radial Basis Function Network

Radial Basis Function (RBF) networks have a similar structure to that of the MLP, however they differ from the MLP in one fundamental way. The activation function of the hidden layer is a Gaussian spheroid function (equation 4.16), that measures how close an input vector is to the centre on which the neuron has been placed in feature space. The output of the activation is Gaussian, with the value returned by the activation reflecting how near a data point is to the centre of the neuron. Points closest to the centre return the highest activation values, and those furthest away return little or no value from the neuron. In this way, the output layer can effectively choose which neuron is closest to the input vector, and use this information to determine what class the data point should belong to.

$$y(\mathbf{x}) = exp\left(-\frac{\|\mathbf{x} - \mathbf{c}\|^2}{2\sigma^2}\right) \tag{4.16}$$

The centre positions of neurons are chosen by calculating the centroid $\mathbf{c}$ of a cluster of data points in space. This centroid defines the centre vector, while the radius of the hypersphere is set by the σ value of the activation function. This can be varied to enclose more or less of the local data points as appropriate. Training of the RBF network uses the simple algorithm below:

1. Train network for one neuron

2. Simulate Network

3. if error < goal then exit; else continue

4. Find input vector with greatest error

5. Add neuron centred on vector with greatest error

6. Retrain weights on output linear layer

7. Go to 2

For a more detailed coverage of RBF networks, [63, 65, 74] all have comprehensive treatment of RBF networks. There have been various applications of RBF networks in MCM; [75,76] both detail some of the advantages in using RBF networks for MCM tasks.

4.1.3 Probabilistic Neural Networks (PNN)

The PNN and the RBF network are very similar, in that they both use a two layer network with a Gaussian spheroid activation function in the first layer of the network. The output layer is different however, as the linear output layer used by the RBF is replaced with a competitive layer in the PNN, which will only allow one neuron to fire, with all others in the layer returning zero. The size of the hidden layer also differs markedly between the different networks; the RBF network uses only as many neurons as are required to cover feature space, while the PNN assigns one neuron for each input in the training set, making for a (potentially) large hidden layer. Indeed, the large hidden layer (and the consequent computational penalty) was one of the reasons that PNN were not developed until the late 1980s, when the original development stages had taken

places during the late 1960s [77]. Figure 4.1 shows the difference between the RBF method of enclosure and that of the PNN.

The PNN can be a Bayesian classifier, approximating the PDF of a class through the use of Parzen windows [78]. Figure 4.3 shows how this achieved by superimposing Gaussian distributions on each other in feature space. The

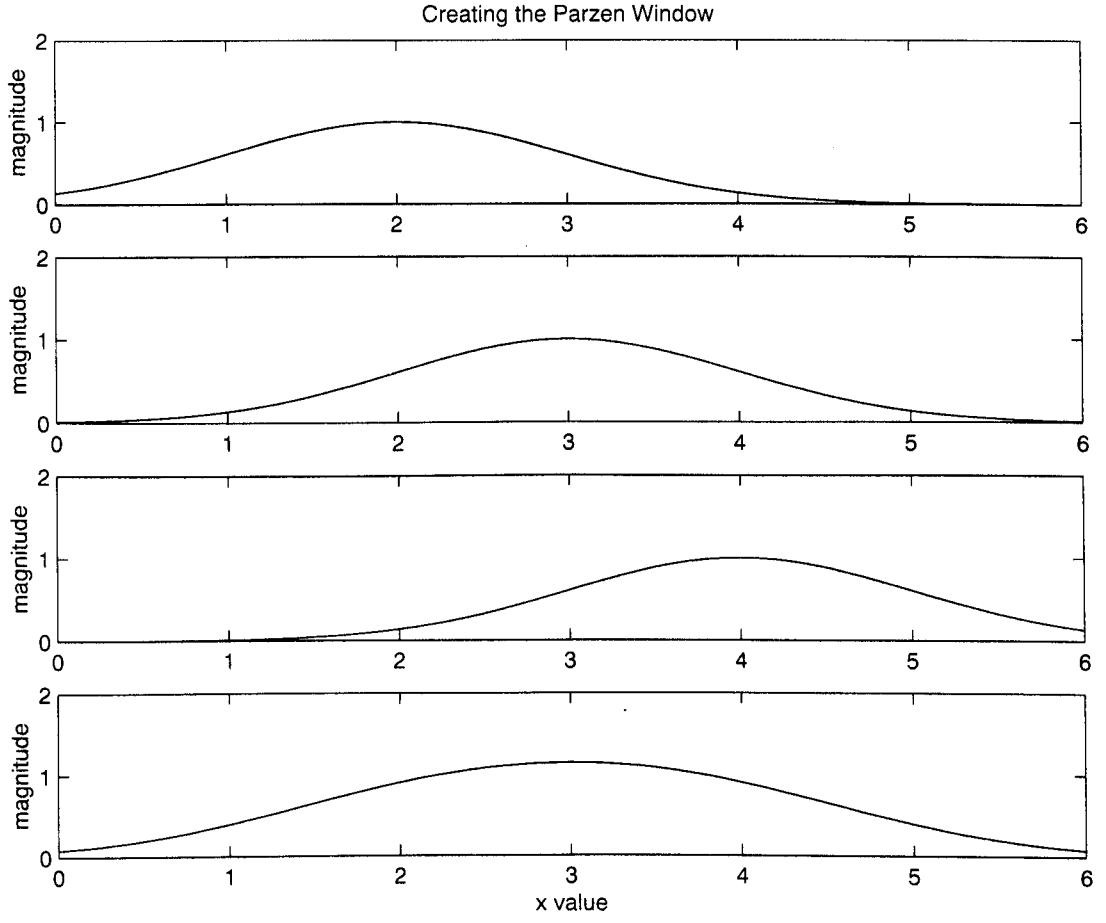


Figure 4.3: Creation of a Parzen Window

generalised expression for calculating the value of a Parzen approximated PDF at given point $\mathbf{x}$ in feature space is:

$$f_A(\mathbf{x}) = \frac{1}{(2\pi)^2 \sigma^p N_A} \sum_{i=1}^{N_A} exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_i\|^2}{2\sigma^2}\right) \tag{4.17}$$

where p is the dimensionality of the feature vector, N_A is the number of examples of class A that the network was trained on (i.e. that form the centres $\mathbf{c}_i$). The σ parameter is the "spread" of the Gaussian function, and can be used to control how smooth the Parzen distribution is. This can have a significant impact on how well the PNN generalises, as if the spread value is too small, then there will be gaps in feature space that should "belong" to a given class, but the functions will not cover them. Varying the spread parameter introduces overlap between the functions, improving the coverage of the PNN, and consequently the generalisation. Figure 4.4 shows the effect of varying the spread parameter in a two dimensional scenario. As can be seen, the medium value of σ is optimal, as

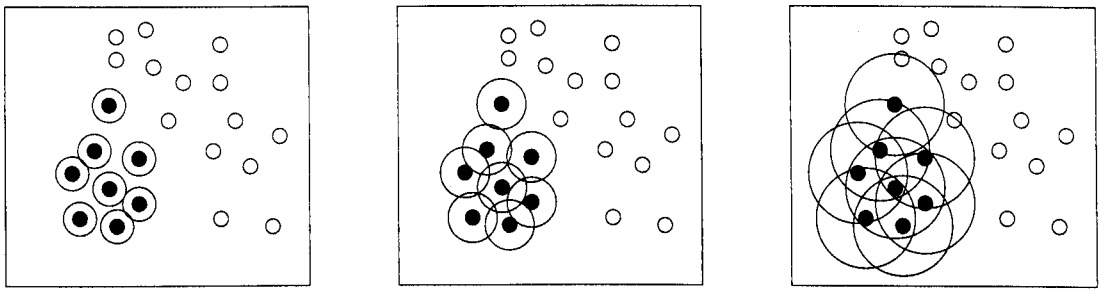


Figure 4.4: The effect of varying the spread (σ) parameter on the PNN's coverage of feature space.

it allows coverage of the area in feature space without accidentally partitioning members of the other class in the boundary, as happens in the large σ diagram.

For each class, it is possible to create a Parzen approximated PDF. There will also be a corresponding value h_A for the relative frequency of the class within the whole training set. Using this information for each class, the PNN calculates the probability that a given sample $\mathbf{x}$ belongs to a given class A as:

$$P(A \mid \mathbf{x}) = f_A(\mathbf{x})h_A \tag{4.18}$$

This expression is evaluated using the corresponding values of $f(\mathbf{x})$ and h for each different class. The class which returns the highest probability is judged to be the correct classification.

Some work using PNNs in MCM has been carried out; [79–81] are all examples of the use of a PNN in monitoring of machines.

4.2 Experiments

In an attempt to compare the different classifiers, a number of different experiments were carried out, using several of the different datasets that were discussed in chapter 3. A total of seven different datasets were used in the comparisons:

- Machine A
  - Plain Statistics (18 features)
  - Statistics of high and low pass filtered signals (36 features)
  - Statistics of signal sums and differences (36 features)
  - Spectral Only (66 features)
  - Combined Statistics (90 features)
  - Combined Statistical and Spectral (156 features)
- Machine B
  - Spectral Features (33 features)

Each of these datasets was tested using both a dual class (fault/no fault) target, and a multi-class target, where the classifier was asked to differentiate between the different fault categories relevant to the machine in question.

Training of the classifiers was carried out using the different training algorithms detailed in the earlier sections, and training was allowed to run until the standard termination terms of the different algorithms were reached. For the RBF and PNN, iterative algorithms were used in an attempt to find the optimum value of width/spread for each, and maximise performance.

4.3 Results: Statistical, Spectral and Combined Datasets

In the tables given on the following pages, in all cases the values given are the best results achieved using the different classifiers, after varying relevant parameters as necessary.

4.3.1 Dual case: Machine A

Table 4.1 shows the results for the dual class problem using the plain statistics (18 features) dataset. Both the MLP and RBF performed well on this dataset, achieving results in excess of 98% on the test data. The MLP performed best, managing 99.9% on the training set, and 99.5% on the test set. The Fault Not Recognised (FNR) rate was very low at 0.5%, with no false alarms. The RBF was only slightly worse, managing 99.2% on the training set and 98.5% on the test set. The FNR rate is lower than that of the MLP, at 0.1%, while the False Alarm (FA) rate was 1.4%. In some respects this network would actually be the more attractive of the two, as in MCM terms, not recognising the existence of a fault is more significant than a false alarm, as it would imply that damage was being done to the machine while the operator thought the machine was working acceptably. The PNN performs poorly in comparison to the other two classifiers,

managing only 83.3% on the training data, and 83.2% on the test set. While the generalisation properties of the network are good, the performance is poor, and proportions of the the FNR rate (16.8%) are very close to the percentage breakdown of each class within the the dataset (16.66%). This would therefore suggest that one of the categories is being completely misclassified by the PNN. Determining which one of the fault categories is causing this problem is unfortunately impossible, as no breakdown of the fault categories is made by the testing algorithm.

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	18	99.9	99.5	0.5	0.0
RBF	18	99.2	98.5	0.1	1.4
PNN	18	83.3	83.2	16.8	0.0

Table 4.1: Dual class: comparison of classification performance with the plain statistics dataset (18 features)

Table 4.2 compares the three classifiers for the high and low pass filtered dataset. Once again, the performance of the MLP and RBF is high, with both classifiers scoring in excess of 98% on the unseen test set data. The MLP has the best overall performance, with a training success rate of 99.6%, and a test success rate of 99.0%. The FNR rate is low, at 1%, while there have been no false alarms at all. The RBF performance is only marginally less than the MLP, reaching 99.1% on the test set, and 98.8% on the training set. The RBF has a good FNR rate of 1.1%, and a small FA rate of 0.1%. The PNN has a much higher success rate for the high and low pass filtered data than the plain statistical data, as the performance on the PNN has risen to 97.4% on the training data, and 96.8% on the test data, which is a more creditable level than the results with the plain statistical set. There is a slightly higher FNR rate than the other two classifiers of 3.2%, however there are no false alarms.

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	36	99.6	99.0	1.0	0.0
RBF	36	99.1	98.8	1.1	0.1
PNN	36	97.4	96.8	3.2	0.0

Table 4.2: Dual class: comparison of classification performance with the high and low pass filtered dataset (36 features)

The results for the signal sums and differences dataset are shown in table 4.3. Both the RBF and MLP networks perform well, with the RBF having the best performance at 99.1% and 98.4% on the training and test sets respectively. The MLP performance is comparable, at 98.8% for the training set, and 98.1% for the test set, with an FNR rate of 1.9%, and no false alarms. The PNN is the worst performer again, with a training success rate of 83.3% and a test success of 83.2%. The FNR rate is 16.8%, with no false alarms. The confusion matrix for the test set with the PNN was examined, to try and determine what the problem was. Table 4.4 shows the classification breakdown on a per-class basis. As can be seen, 25% of the fault (FA) category are being classified as normal (NO). There are four fault sub-conditions that make up the fault category, so it would appear that one of the categories is being completely misclassified by the PNN. The reasons for this drastic problem are not immediately clear, but it may be that either the cage fault or outer race fault is being classified as normal, as reference to figures 3.3 and 3.5 shows that the two classes appear very similar to the normal conditions.

Table 4.5 shows the results of experiments performed with the spectral dataset, consisting of 66 features. Performance of all three classifiers on this dataset is very high, with the MLP and RBF achieving in excess of 99% on the test set, while the PNN manages to reach 96.5% on the test set. The best overall performer is the RBF network, which manages 99.9% success on the training data, and

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	36	98.8	98.1	1.9	0.0
RBF	36	99.1	98.4	1.6	0.0
PNN	36	83.3	83.2	16.8	0.0

Table 4.3: Dual class: comparison of classification performance with the signal differences and sums dataset (36 features)

Classification Success (%)			
		Actual Condition	
		NO	FA
Perceived Condition	NO	**100**	25
	FA	0	**75**

Table 4.4: Classification results for dual class PNN, using signal sums and differences.

99.8% on the test data. The FNR rate is 0.1%, which corresponds to 1 mistake in 960 cases, while the FA rate is also 0.1%, which is a very high classification performance. The MLP is only slightly worse, managing 100% on the training set, and 99.0% on the test set. The FNR rate is slightly higher than that of the RBF, reaching 1%, however there are no false alarms. The PNN manages 98.8% on the training set, and 96.5% on the test set, which is slightly lower than the performance achieved with the high and low pass filtered dataset (table 4.2). The FNR rate is 3.4%, and the FA rate is low at 0.1%.

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	66	100	99.0	1.0	0.0
RBF	66	99.9	99.8	0.1	0.1
PNN	66	98.8	96.5	3.4	0.1

Table 4.5: Dual class: comparison of classification performance with the spectral dataset (66 features)

Table 4.6 shows the results of the combined statistical dataset, which groups all the features from the plain statistics, high and low pass filtered statistics, and signal sums and differences datasets, consisting of 90 features. The RBF is the best overall performer, managing 99.3% on the training set, and 98.5% on the test set, with an FNR of 1.5%, and no false alarms. The MLP performs well, managing 98.9% on the training set, and 97.6% on the test set, however the FNR rate (2.3%) is slightly higher than that of the RBF, and the FA rate is low at 0.1%. The PNN also performs well, reaching 97.7% and 95.7% on the training and test data respectively. However, due to the lower accuracy of classification, the FNR rate is higher at 4.3%, although there are no false alarms.

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	90	98.9	97.6	2.3	0.1
RBF	90	99.3	98.5	1.5	0.0
PNN	90	97.7	95.7	4.3	0.0

Table 4.6: Dual class: comparison of classification performance with the combined statistical dataset (90 features)

The results for the full combined dataset, consisting of the three different statistically based datasets and the spectral dataset (a total of 156 features) is shown in table 4.7. The RBF is the highest overall performer, managing to achieve 100% on both the test and training sets. Both the MLP and PNN have relatively high performances as well, with the MLP managing 99.4% on the training set, and 97.5% on the test set. The FNR rate for the MLP is 2.5%, and no false alarms were generated. The PNN reaches 98.3% on the training set, and 95.1% on the test set, however the lower classification success values mean that the FNR rate is slightly higher, at 4.8% and the FA rate is 0.1%.

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	156	99.4	97.5	2.5	0.0
RBF	156	100	100	0.0	0.0
PNN	156	98.3	95.1	4.8	0.1

Table 4.7: Dual class: comparison of classification performance with the combined statistical and spectral dataset (156 features)

4.3.2 Dual Case: Machine B

Table 4.8 shows the results for the dual class problem with the spectral data from machine B. As can be seen, all three classifiers manage to achieve 100% on both the training and test data, which would suggest that the fault data and the normal condition data are well separated in feature space. This would suggest that even with the limited number of samples available, it should still be possible to determine whether a vibration signature was a fault or normal condition, although the possibility of actually discriminating between fault conditions is much more difficult to guarantee.

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	33	100	100	0.0	0.0
RBF	33	100	100	0.0	0.0
PNN	33	100	100	0.0	0.0

Table 4.8: Dual class: comparison of classification performance with the machine B dataset (33 features)

4.3.3 Multiclass: Machine A

Using the six-class target set, MLP, RBF and PNN were trained, and the best results were recorded for each of the six different datasets available for Machine A. Tables 4.9 to 4.17 show the results for each of the six different datasets with the

different classifiers, in an attempt to give a comparison of how each of the classifiers performs with the different datatypes, and how the generalisation properties of each network vary with the data.

Table 4.9 shows the results achieved with the smallest dataset, consisting of 18 features. As can be seen, there is a marked difference in the performance of the different classifiers on multiclass data. The MLP is the most successful of the three, with a success rate of 99.2% on the training data, and 98.2% on the unseen test data. This is reflected in the relatively low FNR level of 0.8%, and a false alarm rate. The poorest performer by contrast is the RBF network, which while achieving 100% on the test set, only manages to achieve 70.6% on the unseen test set. The FNR rate is a high 20.7%, although the FA rate is quite acceptable at 0%. The PNN does not perform as well as the other two on the training set, achieving only 94.8% on the training set after training, however the generalisation properties of the PNN are significantly better than that of the RBF, as it manages a success rate of 91.1% on the unseen test set, with a FNR of 5.1%, and a FA rate of 0.3%. This is reasonable successful as a classifier, however it is interesting to speculate why the RBF performance is so poor. To examine further, the confusion matrix was examined (see table 4.10) for the unseen test set, to try to determine where the performance was so poor.

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	18	99.2	98.2	0.8	0.0
RBF	18	100	70.6	20.7	0.0
PNN	18	94.8	91.1	5.1	0.3

Table 4.9: Multiclass: comparison of classification performance with the plain statistics dataset (18 features)

As can be seen, in three of the categories (NO, NW and OR), the network is performing very well, achieving a full 100% classification within the category,

Classification Success (%)							
		Actual Condition					
		NO	NW	IR	OR	RE	CA
Perceived Condition	NO	**100**	0	36.9	0	23.8	1.8
	NW	0	**100**	43.7	0	29.3	0.6
	IR	0	0	**0.6**	0	0	0
	OR	0	0	1.3	**100**	0	0
	RE	0	0	0	0	**25**	0
	CA	0	0	17.5	0	21.9	**97.6**

Table 4.10: Classification results for RBF, using plain statistics data only.

with a fourth (CA) achieving 97.6% accuracy. The worst performing category is the inner race (IR) fault, in which the classification success is only 0.6%, which is exceedingly poor. The rolling element (RE) fault also has a poor classification success of 25%. Examining the confusion matrix, it can be seen that most of the errors in classifying the IR and RE fault conditions are misclassified as the two normal conditions and the cage fault condition.

Table 4.11 shows the results of training the ANNs with the high and low pass filtered data. As can be seen, the generalisation performance of all three classifiers is much stronger than the performance on the plain statistics data set, however, there is still a fair degree of variation in the success rate. The MLP reaches a success rate of 98.6% on the training data, and a level of 96.5% on the unseen test data. The FNR and FA on the test set are relatively good, with 2.1% of faults not recognised, while there are no false alarms. The RBF has the second best performance, reaching 100% on the training data, and also a much higher rate (relative to that of plain statistics) of 95% on the unseen data. The false alarm rate is 0% again, while the FNR is 2.1% which is significantly lower than that of the RBF with the plain statistics dataset; This would suggest that there is information in the high and low pass filtered signals which allows the RBF centres to be placed in different locations in feature space and hence

perform much more successfully in terms of classification performance; the higher dimensionality of the feature space will improve things in this respect. The PNN also enjoys a higher performance, managing 97.5% on the training set, and 92.9% on the test set. The PNN has the higher FNR value, at 3.6%, and an FA rate of 1%, which is higher than the other two classifiers. This is probably caused by the fact that there are insufficient vectors in the training set to build a good Parzen approximated PDF, and the PDF of some of the other classes is higher in areas where the training data was relatively sparse for a given class.

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	36	98.6	96.5	2.1	0.0
RBF	36	100	95.0	0.2	0.0
PNN	36	97.5	92.9	3.6	1.0

Table 4.11: Multiclass: comparison of classification performance with the high and low pass filtered dataset (36 features)

Table 4.12 shows the results for the dataset using signal sums and differences. The performance of this dataset is generally slightly worse than that of the high and low pass filtered data but better than that of the plain statistics. Once again, the MLP has the best overall performance, achieving 98.4% on the training set, and 96.7% on the test set. The FNR rate is relatively low, at 1.9%, while there have been no false alarms. The RBF by contrast manages 100% on the training set, and 94.2% on the test set, although there is a higher FNR rate of 3.3%, which might suggest that there is a similar problem to that seen with the plain statistics dataset. The PNN has a high training success rate of 99%, but a low success rate on the test data of only 80.3%. The FNR rate is significant, at 10.4%, although there have been no false alarms at all. This behaviour seemed unusual, and it was decided to examine the classification matrix (Table 4.13).

On examining the matrix, it was once again apparent that there was confusion

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	36	98.4	96.7	1.9	0.0
RBF	36	100	94.2	3.3	0.0
PNN	36	99.0	80.3	10.4	0.0

Table 4.12: Multiclass: comparison of classification performance with the signal differences and sums dataset (36 features)

Classification Success (%)							
		Actual Condition					
		NO	NW	IR	OR	RE	CA
Perceived Condition	NO	**58.8**	16.9	20	0	10.6	14.4
	NW	16.2	**81.9**	0	0	0	6.8
	IR	0	0	**78.8**	0	0	0
	OR	0	0	0.6	**100**	0	0
	RE	0	0	0.6	0	**89.4**	0
	CA	25	1.25	0	0	0	**78.8**

Table 4.13: Classification results for PNN, using signal sums and differences.

between the two normal conditions and the fault conditions, and also between the cage fault and the normal conditions. A significant number of the fault conditions were misclassified as normal conditions, although interestingly, predominantly as the new normal bearing (NO), and not the worn normal bearing. This is probably accounted for by the fact that the normal conditions are concentrated in a very tight area, as shown by the plots of the bearing vibrations (Fig. 3.4). With more training vectors located at or near the origin of acceleration, then the Parzen approximation of the normal conditions about the origin would be higher than that of the fault conditions; this would tend to make the PNN classify the vectors as either one of the normal conditions or the cage fault, as there would be relatively few training vectors for the fault conditions around the origin, while there are a significant number of training vectors (and hence Gaussian centres) for the other classes.

Table 4.14 shows the results for the spectral dataset; this is probably the dataset for which all three classifiers have the best performance. The RBF is the highest performing classifier, managing to achieve 100% on the training set, and 99.5% on the test set. The MLP is slightly worse, managing 100% again on the training set, and 98.8% on the test set. The best performance of the PNN occurs on this dataset, achieving 98.9% on the training set, and 96.6% on the test set; there is a moderate FNR value of 2.3%, however there are no false alarms at all. It is interesting to note that all three classifiers perform well with this dataset; the probable cause of this is that discrimination between classes is relatively easy in the spectral data. The FFT was a low resolution process, containing only 32 bins; this would suggest that there is a high likelihood that spectral lines will lie in either one or two bins, and remain relatively static for a given condition. These static properties make it relatively easy to position Gaussian centres in feature space, without the same confusion problems that occurred with some of the statistically based datasets. The spectral data is easily separable, and advantages of this are shown in the results achieved.

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	66	100	98.8	0.7	0.0
RBF	66	100	99.5	0.1	0.0
PNN	66	98.9	96.6	2.3	0.0

Table 4.14: Multiclass: comparison of classification performance with the spectral dataset (66 features)

Table 4.15 shows the results with the combined statistical dataset of 90 features. As can be seen, the performance on the statistical dataset is slightly poorer than that of the spectral dataset (Table 4.14). Once again, the RBF network is the best performer, with classification success rates of 100% and 97.1% for the training and test sets respectively. The MLP performs well, with success rates of

98.9% and 96.8% respectively. The PNN is the poorest performer, at 98.0% on the training set, and only 90.2% on the test set. This is a relatively poor level of performance. There are a lot of faults (5.2%) being misclassified as normal conditions, which is of concern. Table 4.16 shows the breakdown of the confusion matrix. As can be seen, most of the misclassification is taking place with the normal (NO) condition, which seems to be misclassifying several different fault categories as normal. This is probably due to the hypothesis raised earlier, regarding the high number of Gaussian centres belonging to one class in an area of feature space. There has also been some misclassification between the normal and cage fault (CA) condition, with 10.6% of the normal data being classified as cage fault. Again, this is probably due to the position of cage fault vectors in close proximity to the normal data.

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	90	98.9	96.8	2.6	0.0
RBF	90	100	97.1	0.8	0.0
PNN	90	98.0	90.2	5.2	1.8

Table 4.15: Multiclass: comparison of classification performance with the combined statistical dataset (90 features)

Table 4.17 shows the results for the combined statistical and spectral dataset; this is the largest of the datasets, containing a total of 156 features. It might be hoped that this dataset, would have the best performance, as there is a combination of both the statistical and spectral results, and it might be the case that the combination of the two data types would improve the classification success further. Unfortunately, this does not seem to be the case, as while the performance is good, it is not any better than the spectral dataset. The best performance is the RBF network, which manages to achieve 100% on the training set, and

Classification Success (%)							
		Actual Condition					
		NO	NW	IR	OR	RE	CA
Perceived Condition	NO	**79.4**	5	11.3	0	8.8	6.2
	NW	10	**95**	0	0	0	5
	IR	0	0	**86.8**	0	0	0
	OR	0	0	0.6	**100**	0	0
	RE	0	0	1.3	0	**91.2**	0
	CA	10.6	0	0	0	0	**88.8**

Table 4.16: Classification results for PNN, using combined statistical features.

99.1% on the test data, while the MLP manages a slightly lower 99.8% on training, and 97.0% on the test set. The PNN has what is overall the second best performance in all the datasets, reaching 99.0% on the training set and 93% on the test set. Once again the PNN has a moderately high FNR rate of 5.3%, but no false alarms.

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	156	99.8	97.0	2.3	0.0
RBF	156	100	99.1	0.1	0.0
PNN	156	99.0	93.0	5.3	0.0

Table 4.17: Multiclass: comparison of classification performance with the combined statistical and spectral dataset (156 features)

4.3.4 Multiclass: Machine B

Table 4.18 shows the performance of the classifiers for the spectral dataset for machine B. As can be seen, the levels of performance differ rather markedly from the datasets for machine A, with none of the performance values exceeding 95% on the test set. The MLP is the worst performer of the three classifiers, although achieving 100% on the training set, on the unseen test set, the classifier only manages 76.8%, with a relatively low FNR rate of 1.8%, and a FA rate of 0.0%.

This suggests that any confusion that takes places is within the fault categories themselves. The RBF has the second highest performance, reaching 100% on the test set, and 89.3% on the test set. The FNR and FA are both 0%, which means that all the confusion must be taking place within the fault categories. The PNN by contrast, is the highest performing of the three classifiers, managing 100% and 94.6% on the training and test sets respectively. It is interesting to note that in this particular case, the PNN approach with Parzen windows appears to offer the best solution of the three approaches where it has tended to be the poorest performing classifier in the multiclass data from machine A.

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	33	100	76.8	1.8	0.0
RBF	33	98.2	89.3	0.0	0.0
PNN	33	100	94.6	0.0	0.0

Table 4.18: Multiclass: comparison of classification performance with the machine B dataset (33 features)

Examining the results for the MLP (Table 4.19), a number of things are evident. As can be seen, most of the confusion is occurring between the fault categories - particularly the two cage faults (CA1 and CA4). This is not entirely unexpected, as the two faults are similar in nature, and it may prove quite difficult to discriminate between them on the basis of data provided. It may be the case that some features are "obscuring" the information provided by other features, or indeed it may be the case that there is not sufficient information contained in the current features to allow better discrimination.

Classification Success (%)						
		Actual Condition				
		NO	OF	CA1	CA4	WO
Perceived Condition	NO	**100**	0	0	8.3	0
	OF	0	**100**	0	0	0
	CA1	0	0	**91.6**	8.3	16.6
	CA4	0	0	8.3	**58.4**	41.7
	WO	0	0	0	0	**41.7**

Table 4.19: Classification results for MLP, using spectral features of machine B.

4.4 Discussion

The results given have shown how the three classifiers deal with different pre-processed data. Each one appears to have different strengths and weaknesses, dependent upon the data set which is used to train the network. For the dual class problems, where the classifiers are asked to partition the data into fault/no fault cases, the overall performance of the three classifiers is good, but the PNN seems to have problems detecting one of the fault classes, and persistently confuses the classification with one of the normal conditions (Tables 4.1 and 4.3). This is of concern, however there is a question as to *why* this is happening, and what exactly the problem is.

It seems strange that the other two classifiers, when confronted with the datasets in question, are able to train the training set to classify the data in excess of 99%, but the PNN is only capable of 83.3% on the same training set. One possible explanation is through the use of the Parzen approximated PDF. In the two-class case, we have a number of different conditions that are sub groups of each of the fault and no fault classes. The no fault class has two normal conditions, which are very tightly clustered in feature space. The cage fault (which belongs to the fault class) is also tightly clustered about the same point in feature space (as shown in figure 3.3). Through the basic principle of the

Parzen approximated PDF, where individual distributions are superimposed on each other, then if there are twice as many examples belonging to one class in one area as there are of another class, then the PDF of the more frequent class will be the larger of the two, and as a result, will be much more likely to fire than the correct PDF when a fault class vector is presented to the network. This is a fundamental problem with the PNN. Obviously, there is insufficient information contained in the preprocessed data to allow the PNN to separate the data adequately in feature space. This effect can also be seen in the multiclass results, although the effect is lessened, as there are the same number of examples from each class, and as a result, the imbalance between different classes is much less, making the performance of the PNN better in the multiclass scenario. However, it may be possible to improve the performance of the PNN by introducing some form of weighting function that will take account of any imbalance in the number of training samples available for each class, and improve performance in the respect. It is interesting to note that in the multiclass experiments using the data from machine B, the PNN scored the highest of the three classifiers in terms of generalisation performance.

The RBF and MLP both seem to be fairly robust classifiers, although the RBF classifier seems to be slightly more erratic than the MLP, as evidenced by the results in the multiclass cases, where the RBF scores highly in training set (usually of the order of 99-100%), and then the performance drops off to a much greater extent than the MLP, which might have a slightly lower training success, but seems to generalise slightly better. Perhaps the worst example of this behaviour in the multiclass case is the plain statistical data (Table 4.9), where the performance of the RBF is 100% on the training data, and only 70.4% on the test data.

One possible cause may be that the width parameter of the RBF is causing a

problem, in that to be able to find a width value that manages to enclose all of each individual fault class (which is most likely to be separated in feature space), this forces the radius of the sphere enclosing the normal conditions to become so large that it starts to enclose some of the data points belonging to the other conditions, and this causes the misclassification. There is evidence to support this in Table 4.10, as misclassification is consistently happening in the normal conditions, and they are misclassifying fault conditions as normal conditions. This problem is difficult to address, as it is a simple matter to allow different neurons to have different width values, but it is much more difficult to train the network, as there are many more parameters that influence the performance of the network, and it is difficult to determine just how to set the correct width for a given neuron.

The MLP is perhaps the most suitable choice for a robust classifier in problems of this type. While the performance may not always be as high as some the RBF classifiers, the generalisation performance over all the different datasets seems to be more consistent than the other two classifiers. The MLP also offers advantages from the point of view that it is also the most compact in terms of network layout, and hence requires less computation after training has taken place.

4.5 Summary

This chapter has covered the basic principles of three different types of ANN: the MLP, RBF and PNN. The basic principles behind each of the different networks was presented, and the results of experiments performed using a two-class fault/no fault target dataset, and a multiclass dataset, which asks the classifiers to characterise different fault conditions. Experiments were run using data from two different machines, and it was seen that there are problems in using the PNN

with some types of training data. This requires further work to clarify whether this is the case indeed. The performance of the RBF was evaluated, and it was determined that certain problems exist with the RBF, due to using a standard width parameter for all neurons. The MLP was found overall to be the most consistent classifier of the three considered, and gave the best generalisation performance with all the datasets. For this reason, it is believed to be the best choice for most MCM classification problems. The data from machine A was found to be easy to classify in both the dual case and multiclass scenarios, however the data from machine B was very easy to classify in the dual class case, however the generalisation performance was found to be much poorer in the multiclass case. The reasons why the PNN should outperform both the MLP and RBF with the multiclass data from machine B are unclear, but it may be the case that for data with very limited datasets available for training, the PNN can outperform other classifiers.

Chapter 5

Support Vector Machines

5.1 Support Vector Machines

Support Vector Machines (SVMs) can trace their roots back to statistical learning theory, as introduced by Vapnik [82] in the late 1960s. It was not until the early 1990s that the techniques used for SVMs began to emerge and become practical with the increased computing power available. While multiclass SVMs have now been developed [83, 84], the simple SVM solves a binary problem - in that data exists on one side or other of a hyperplane. The hyperplane is defined by a number of *support vectors*, which are a subset of the training data available for both cases, and is used to define the boundary between the two classes. The use of the support vectors allow complex boundaries to be created, and through the minimisation of a quadratic programming problem, the margin of separation between each class of data is maximised. Space here prevents the detailed explanation of the principles of an SVM, however a number of good tutorial papers exist [1, 85, 86], and these cover the basic principles well.

In simple terms, the SVM can be thought of as creating a line, or hyperplane between two sets of data. If we imagine a two dimensional case, the action of the

SVM can be shown easily. In Figure 5.1, a series of data points for two different classes of data are shown, black circles (class A) and white squares (class B). The SVM attempts to place a linear boundary between the two different classes, and orientate it in such a way that the *margin* (represented by the dotted lines) is maximised. In other words, the SVM tries to orientate the boundary in such a way as to ensure that the distance between the boundary and the *nearest data point* in each class is maximal. The boundary is then placed in the middle of this *margin* between the two points. The nearest data points are used to define the margin, and are known as *support vectors* (represented by the gray circles and square). Once the support vectors have been selected, the rest of the dataset is not required, as the support vectors contain all the information needed to define the classifier.

Mathematically, the SVM can be defined comparatively easily. The explanation that follows is an overview of the functioning of an SVM; for a more detailed explanation, it is suggested that one of the several excellent tutorials available is consulted.

For any point that lies on the boundary line, we can write:

$$(\mathbf{w}.\mathbf{x}) + b = 0 \tag{5.1}$$

where $\mathbf{w}$ is a vector that defines the boundary, $\mathbf{x}$ is an input, or data vector, and b is a scalar threshold value. At the margins, H1 and H2, where the support vectors are situated, the equations for class A and B respectively are:

$$(\mathbf{w}.\mathbf{x}) + b = 1 \tag{5.2}$$

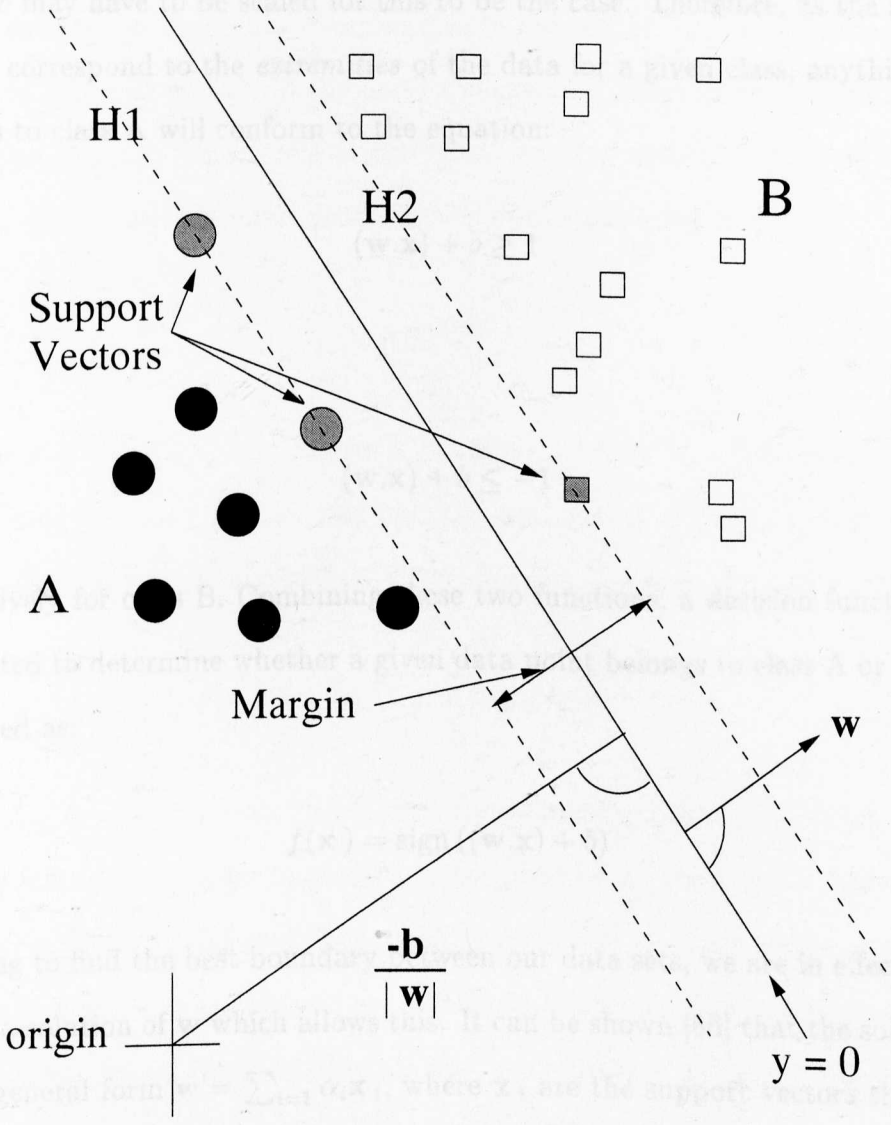


Figure 5.1: Separation of two classes by SVM

and

$$(\mathbf{w}.\mathbf{x}) + b = -1 \tag{5.3}$$

$\mathbf{w}$ and b may have to be scaled for this to be the case. Therefore, as the support vectors correspond to the *extremities* of the data for a given class, anything that belongs to class A will conform to the equation:

$$(\mathbf{w}.\mathbf{x}) + b \geq 1 \tag{5.4}$$

and

$$(\mathbf{w}.\mathbf{x}) + b \leq -1 \tag{5.5}$$

respectively for class B. Combining these two functions, a decision function can be created to determine whether a given data point belongs to class A or B. This is defined as:

$$f(\mathbf{x}) = \text{sign}\,((\mathbf{w}.\mathbf{x}) + b) \tag{5.6}$$

In trying to find the best boundary between our data sets, we are in effect trying to find a solution of $\mathbf{w}$ which allows this. It can be shown [86] that the solution is of the general form $\mathbf{w} = \sum_{i=1} \alpha_i \mathbf{x}_i$, where $\mathbf{x}_i$ are the support vectors that have been kept from training. Substituting this into equation 5.6, we get:

$$f(\mathbf{x}) = \text{sign}\,(\sum_i \alpha_i (\mathbf{x}.\mathbf{x}_i) + b) \tag{5.7}$$

A case has been constructed for a linear boundary in two dimensions; however, there will be cases where the linear boundary in input space will be unable to separate two classes properly. This at first seems a large problem, however, by transforming the data into a higher dimensional space, it is possible to create a hyperplane that allows linear separation in the higher dimension (which corresponds to a curved surface in the lower dimensional input space). In SVMs, this is achieved through the use of a transformation, $\Phi(\mathbf{x})$, which transforms the data from an N dimensional input space into Q dimensional feature space:

$$\mathbf{s} = \Phi(\mathbf{x}) \tag{5.8}$$

where $\mathbf{x} \in \mathbb{R}^N$ and $\mathbf{s} \in \mathbb{R}^Q$. Figure 5.2 shows the effect that this transformation has on some fictitious dataset, and how the separability of the data changes after the transformation. Substituting the transformation into equation 5.7, this gives:

$$f(\mathbf{x}) = \text{sign}\,(\sum_i \alpha_i(\Phi(\mathbf{x}).\Phi(\mathbf{x}_i)) + b) \tag{5.9}$$

Making transformations into higher dimensional space is relatively computationally intensive, in order to perform a dot product on the results. A kernel can be used to perform this transformation and the dot product in one step if the transformations that are allowed are restricted to those which can be replaced by an equivalent kernel function. In this way, it is possible to reduce the computational load, but retain the effect of the higher dimensional transformation. The kernel function, $K(\mathbf{x}, \mathbf{y})$ is defined as:

$$K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}).\Phi(\mathbf{y}) \tag{5.10}$$

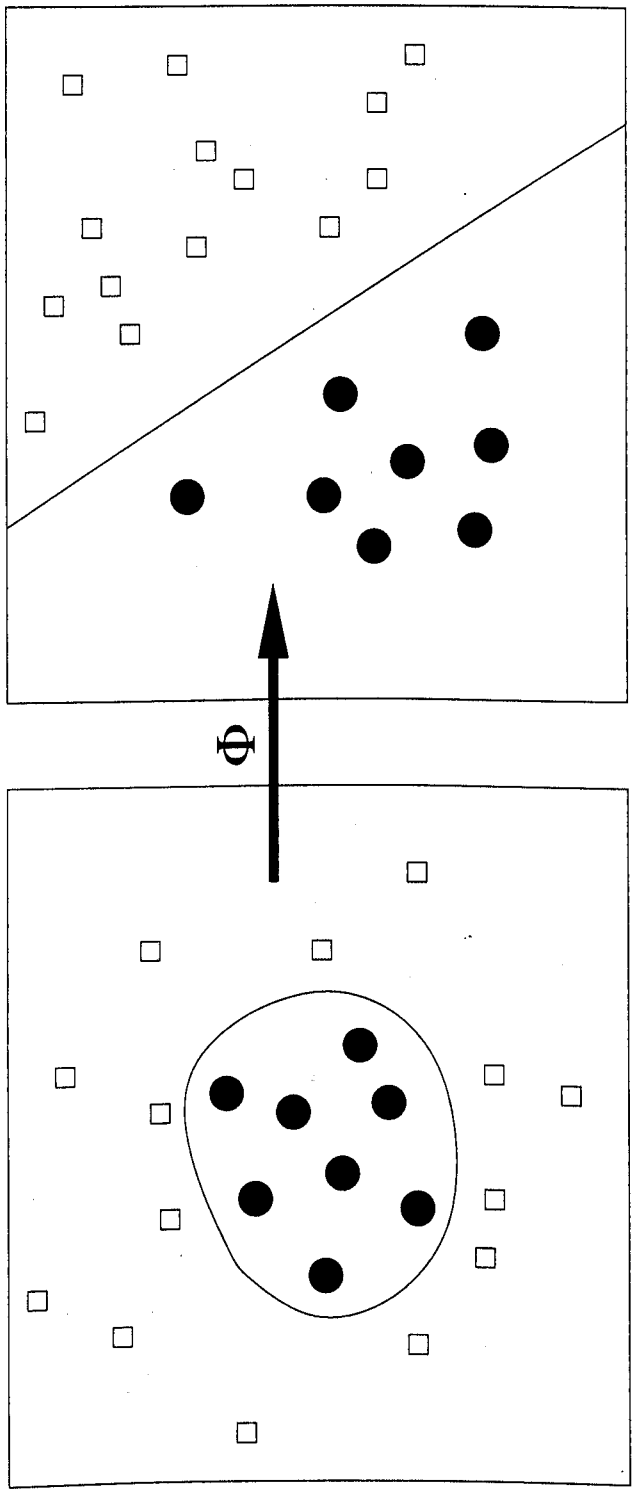


Figure 5.2: Nonlinear transformation from input to a higher dimensional feature space

substituting into equation 5.9, we get the final basic form of the SVM:

$$f(\mathbf{x}) = \text{sign}\,(\sum_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b) \tag{5.11}$$

α_i is used as a weighting factor to determine which of the vectors in the input are actually support vectors (i.e. $\alpha_i > 0$), while others may have a lesser importance. Input vectors with a corresponding $\alpha_i = 0$ are not support vectors and may be, for all intents and purposes, discarded. The kernel used for the experiments in this paper is the RBF kernel, defined by the equation:

$$K(\mathbf{x}, \mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}} \tag{5.12}$$

The width of the RBF function, σ, can be determined by an iterative process that selects the optimum width on the basis of the full datasets.

5.1.1 Multiclass SVMs

The SVM that we have just explored only covers the binary case - i.e a two class problem. For multiclass problems, there are two options available; using a multiclass SVM, where the single output gives several possible output levels [83], or to use several SVMs in parallel; this is the method that has been selected for this particular application, where each SVM is used to recognise one category against all others, and the maximum output over all the categories is taken to be the correct classification.

5.2 Training

Training of SVMs is one of the areas attracting the most attention within current SVM research. SVM theory [1] states that the training of an SVM is carried out by maximising the quadratic form

$$W(\alpha) = \sum_{i-1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j K(x_i, x_j) y_i y_j \tag{5.13}$$

subject to the constraints

$$\sum_{i=1}^{l} \alpha_i y_i = 0, \qquad \alpha_i \geq 0 \tag{5.14}$$

Various different techniques are available for maximising the quadratic; however all algorithms proposed tend to be very intensive computationally, with consequent penalties in training time. Vapnik's original approach was to treat the problem as a straightforward Quadratic Programming (QP) problem. However, the QP problem is such that it involves solving $N \times N$ dimensional matrices, where N is the number of examples in the training set. As the training set increases in size, so does the awkwardness of the problem. For the datasets from machine A, which contains 960 examples, this means that the matrix to be solved contains 921,360 elements. This has the potential to become a significant problem with larger datasets. Several approaches have been advanced to try and deal with the QP problem in a more tractable manner; Vapnik proposed a technique now known as "chunking" [87], where a series of sub-problems using a subset of the matrix are solved. After solving each problem, all the rows and columns of the matrix corresponding to those values of α_i which are zero are removed from the matrix, and replaced with new rows and columns which have still to be solved. The new matrix is solved, and the algorithm repeats until the remaining

QP problem contains all the values that correspond to the support vectors (i.e. non-zero solutions of α). This is then solved to give the final solution. Osuna *et al* [88] proposed a different method of solving this problem, in which a series of subproblems are solved, with the matrix always a fixed size. After every iteration, a set number of rows and columns are removed, and replaced with other α values which have yet to be solved. By repeating this process until all values have been determined, the solution to the problem converges. All of these approaches require the use of a QP solving package, which is a numerical solver working by iteration. Platt [89] proposed the Sequential Minimal Optimisation algorithm (SMO), which rather than solving multiple instances of the QP problem using an iterative numerical process, solves the α values analytically, in pairwise steps. This allows the optimal α values to be found in a single step, at the cost of many solutions being calculated in order to solve the problem. However, the speed with which the analytical solution can be calculated means that calculating a large number of analytical solutions is invariably faster than calculating a smaller number of numerical steps. Indeed, figures have been recorded which show that SMO can be up to 1200 times faster [89] than the chunking type algorithms, with a linear SVM. Further improvements to SMO have been proposed [90], which improve training speed further.

Unfortunately, the algorithms mentioned above work on the assumption that the training data used is statistically representative of the likely set of values that will be encountered. The very nature of condition monitoring means that it is extremely unlikely that it will be possible to get hundreds if not thousands of examples of different types of failure. As a result, it is very difficult to generate a training set that is statistically representative of the real life fault conditions that are likely to be encountered, and as such this is a fundamental problem in using these training methods for SVMs. Generalisation of SVMs can be controlled

through the application of a measure known as the *Vapnik-Chervonenkis* (VC) dimension [91], which can be used to describe the maximum number of points that can be correctly classified for all possible different binary classifications. However, the VC dimension is difficult to evaluate in most practical cases [92]. It would be preferential to carry out training using a validation set so as to allow the training to stop when the generalisation began to deteriorate, as it is comparatively simple to use the performance of the validation set to monitor the generalisation performance of the SVM. Due to the nature of the training data, and also the limited number of training examples available for training with machine B, it was decided to use a technique which allowed the use of a validation set to monitor the generalisation performance of the SVM during training.

In order to train the SVM using a validation set, the Kernel Adatron (KA) [93, 94] algorithm was used, which uses gradient ascent to maximise the Lagrangian given in equation 5.13. This process was found to work well, with good generalisation. This gives a performance comparable to traditional SVM solutions, without having to utilise expensive QP packages in order to carry out the optimisation. The algorithm also scales better than the QP packages as the size of the training set increases [93].

5.2.1 Kernel Parameter Selection

One of the big problems in SVMs is the selection of the value of a kernel parameter that will allow good performance. The kernel parameter can make a huge difference to the performance of the SVM on the task in question, however one of the largest problems is that there is (as yet) no analytical way to determine the optimum value of the kernel parameter for a given problem; the selection of

values is still very much regarded as an art, in the same way that selecting the size of the hidden layer in an ANN is still an empirical task.

Using the Gaussian RBF kernel detailed in equation 5.12, there is only one parameter to be selected. The σ parameter of the Gaussian kernel determines the "width" of the function. This can be thought of as a hypersphere, with the σ value determining the radius of the hypersphere. It is a comparatively simple process to select the best parameter value using an iterative process, such as [95] to try and find the optimal generalisation performance. For the multiclass case, the problem is more complex, as there are several SVMs running in parallel, and each has a width parameter. The interaction between the different width parameters is complex, as dependent upon the separation of each class in space, training the different SVMs for the best performance in each category and then assuming that they will correctly separate the different classes correctly does not always work.

Two main problems occur in trying to allow the SVMs to choose different width parameters for each class; one is a problem of overlap between classes, and another is the problem of the variance of different features in different classes. The datasets used vary in dimension between 18 and 156. Obviously, for each dataset there will be features that for one class have little or no variance, while other features will have a large variance within the same class. Due to the nature of the spherical Gaussian function, the kernel has a constant diameter in all dimensions, which means that those features with a large variance will dominate the search for an optimal value of kernel parameter, tending to give the sphere a radius large enough to enclose those features with the large variance. This causes problems with those features having a smaller variance because the sphere that is generated may be so large as to enclose data points from another class in one of the dimensions with less variance.

While the whole training dataset is normalised to have zero mean and a standard deviation of one, within each feature different classes will have different standard deviations that are less than one. It would be feasible to normalise the data on the basis of one feature, making the variance of the other classes much greater than one, however the question of which feature is the correct one to use in order to normalise the dataset arises. Additionally, not having a dataset normalised to one over the whole range of the dataset increases the training time drastically. However, as there is a degree of correspondence between the standard deviation of the Gaussian distribution and the σ value of the Gaussian kernel, it is fair to assume that there will be some correspondence between the standard deviation of the data for each class and the value of the kernel parameter required for good classification performance. If we could assume that the distribution of each class was Gaussian in nature, we could use a value of three times the standard deviation directly, knowing that statistically, it will catch 99% of all the training data. In practice, we must determine a scale factor to apply to the standard deviation of the data, that will allow good performance on classification.

Using the standard deviation creates a problem with regard to what value to use, since for a multidimensional input vector, each input feature will have a different standard deviation value for a given class. Table 5.1 shows the standard deviations for each of the first five input features of the combined statistical data set taken from machine A, on a class by class basis. As can be seen, examining the σ values for a given class, there is a fair degree of variation between the different features. How can a value be selected that will allow good discrimination amongst all features? One approach is to calculate the standard deviation of all the members of a given class for each feature (σ_i) in the input vector $\mathbf{x}$. The individual standard deviations form a vector, $\vec{\sigma}$; our averaged value can then be

Class	σ_1	σ_2	σ_3	σ_4	σ_5
NO	0.144	0.248	0.196	0.015	0.020
NW	0.118	0.235	0.009	0.010	0.011
IR	0.126	1.266	0.534	0.584	1.099
OR	0.095	0.158	0.017	0.016	0.014
RE	0.558	0.320	1.466	0.536	0.459
CA	0.113	0.262	0.008	0.007	0.033

Table 5.1: Standard deviations of the first five input features for all six classes of the plain statistical data for machine A

found by taking the norm of this vector:

$$\sigma_{av} = ||\vec{\sigma}|| \tag{5.15}$$

This averaged value can then be used directly in the kernel equation 5.12

The second approach is to modify the kernel function in order to allow different widths for each feature to be incorporated, allowing the creation of a *hyperellipsoid*, rather than a hypersphere, which will incorporate the relative widths of each of the different features in the classification bounds. The modified kernel is of the form shown in equation 5.16. For each feature in the training set, there is now a corresponding width parameter, σ_k. The value of σ_k is scaled by a scaling factor v, which varies the size of the hyperellipsoid, while maintaining the proportions of the data in all dimensions.

$$K(\mathbf{x}, \mathbf{y}) = \prod_{k=1}^{P} exp\left(-\frac{(x_k - y_k)^2}{2(v\sigma_k)^2}\right) \tag{5.16}$$

Various different approaches have been advanced towards this problem, as the variance of a class within a feature is very much a problem with real datasets. Chapelle and Vapnik [96] proposed a method which rescales the data using the

eigenvalues of the covariance matrix of the training data, which is slightly different from the multiclass case which is faced here. Schölkopf *et al* [97] have also proposed an approach that is slightly more complex, but uses the same general principle. Amari and Wu [98] have also proposed another scaling technique which increases the spatial resolution near to the boundary by applying a magnifying factor at, or close to the support vectors. This is a two stage process, involving training the SVM twice, and so there are drawbacks to this technique. All of these different techniques are based around the fundamental principle that the training data supplied for one class is representative of that class, allowing the kernel scaling and parameters to be selected without the use of a validation set; unfortunately, due to the fault conditions present in the MCM data and the nature of the faults (which can be very erratic in nature), no guarantees can be given as to whether the training data is representative of the likely state of the machine during these fault conditions. For this reason, kernel selection has been carried out using a validation set, and monitoring the performance over two sets.

5.3 Experiments

In order to compare the performance of the different SVMs a series of experiments were carried out on four different datasets; from machine A, the combined statistics, spectral, and the combined statistical and spectral datasets were used, while the single spectral dataset from machine B was also used. Three different types of SVMs were trained for each dataset; the "constant width" SVM, where the standard kernel function was used, and the kernel width parameter was kept fixed over all classes of the classifier, the "averaged width" SVM where the kernel parameter was averaged on the basis of the standard deviations within a class (equation 5.15), and the "specific width", where the kernel parameters are set

using the standard deviations (equation 5.16). For the constant width SVM, the training algorithm was allowed to iterate over different values of σ in order to try and find the best performance for each dataset. For the averaged width kernel, different scaling factors between 1 and 9 were applied to the kernel parameter, in order to examine what impact the scaling factor had on the performance of the SVM. For the "specific width" kernel, the SVM was trained with different scaling values ranging between 1 and 9, with the best value selected. Performance was measured using the classification success on the unseen test set.

Experiments were carried out using both the two class fault/no fault target data, and the multi class target data, where the SVM also attempts to characterise the nature of the fault. As a comparison, the performance of the MLP on the same datasets is also considered.

5.4 Results

5.4.1 Dual Case: Machine A

Table 5.2 shows the results for the experiments carried out using the spectral data from machine A. As can be seen, the results are a mixed bag. The MLP classifier has a robust performance, achieving 100% on the training set and 99% on the test set. The FNR value is low at 1%, and there are no false alarms. The constant width SVM has the worst performance, managing 98.2% on the training set, but only 89.4% on the test set. The FNR value is relatively high at 9.0%, while there are also a number of false alarms (1.6%). The averaged width SVM performs better overall, having a better generalisation performance, with 93.8% on the training set and 91.5% on the test set. The FNR rate is low at 0.3%, however there is a high FA rate of 8.2%. The specific width SVM has the best

performance of the three SVMs tested, managing 96.6% on the training set, and 93.9% on the test set. The FNR rate is again low at 0.2%, while the FA rate is moderate, at 5.9%. It is interesting to note the switch between the proportions of the FNR rate and FA rate between the three different types of SVM. Where the SVM uses the constant width, the FNR is high while the FA is low, but when the averaged and specific width SVMs are trained, the FA becomes high, while the FNR is very low. This can probably be explained in terms of the different ways that the different SVM approaches work.

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	66	100	99.0	1.0	0.0
SVM (const. width)	66	98.2	89.4	9.0	1.6
SVM (av. width)	66	93.8	91.5	0.3	8.2
SVM (sp. width)	66	96.6	93.9	0.2	5.9

Table 5.2: Dual Class: comparison of classification performance with the spectral dataset (66 features)

The constant width SVM uses the same diameter of sphere for each class of data in the classifier, while the other two approaches try and set the sphere size proportional to the variance of the different classes. Fundamentally, the problem is caused by the fact that the two different classes are nonseparable, and there is a degree of overlap between the two clusters. In the constant width SVM, the fault data has the larger variance of the two classes, and in order to allow the SVM to enclose the fault data with a sphere, the normal class sphere has to increase to a size which is disproportionately large in terms of the variance of the data in feature space. As this increase in size happens there is an area of overlap

between the two spheres where some of the fault data falls closer to the centre of the normal sphere than the fault sphere, and as a result will be classified wrongly as normal conditions.

The averaged and specific width SVMs work in the opposite fashion. As the fault data has a much larger variance, the fault sphere has a large diameter, although the variance is probably large in only a few dimensions. This forces the enclosures to encompass a portion of the normal class, which in turn causes the normal conditions to be enclosed as faults.

The results for the experiments using the combined statistical datasets are shown in table 5.3. Once again the MLP is the highest performing of the different classifiers, managing 98.9% on the training set, and 97.6% on the test set. The FA rate is 0%, and the FNR rate is 2.3%. The averaged width SVM is the worst overall performer in this case, only managing 81.4% on the training set, and 80.4% on the test set. The FNR is significant at 19.0%, while the FA rate is lower at 0.6%. The constant width SVM performs slightly better, managing 99.4% on the test set, and 84.9% on the test set. The FNR rate is 12.6%, while the FA rate is 2.5%. The specific width SVM has the best performance of the three SVMs, managing to reach 85.6% on the test set, and 85.5% on the training set. The generalisation performance is good, however the FA is high at 14.1%, while the FNR rate is low at 0.3%.

It is interesting to note that the same switch in the proportions of the FNR and FA rates occurs again between the three SVMs. This is likely to be caused by the same problem as detailed earlier. The overall SVM performance on this dataset is very poor, and it is interesting to compare the difference in the performances of the ANN and the SVMs. Why can the ANN manage to classify the data with such a high degree of success, while the SVMs cannot? Even the specific width SVM, which might be expected to have the best performance of the different

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	90	98.9	97.6	2.3	0.0
SVM (const width)	90	99.4	84.9	12.6	2.5
SVM (av. width)	90	81.8	80.4	0.6	19.0
SVM (sp. width)	90	85.5	85.6	0.3	14.1

Table 5.3: Dual Class: comparison of classification performance with the combined statistical dataset (90 features)

SVMs, only manages a comparatively poor partition of the data. One possible explanation is that there is a greater degree of overlap between the two classes than there was with the spectral data, and so the separation of the different classes is more awkward with the SVMs.

Table 5.4 shows the results for the experiments run using the combined statistical and spectral dataset. As can be seen, once again the MLP has the overall best performance, managing 99.4% on the test set and 97.5% on the test set. The FNR rate is low at 2.5%, and there are no false alarms. Of the three SVMs, the averaged width SVM performs the worst, managing only 81.7% and 79.1% on the training and test sets respectively, with a low FNR value of 0.3%, and a higher FA rate of 20.6%. The constant width SVM performs better, managing 99.8% on the training set, but only 85.3% on the test set, coupled with a FNR rate of 12.9%, and a false alarm rate of 1.8%. The best performing SVM was the specific width, which while only managing 90.3% on the training set, has the highest classification success rate at 86.9% on the unseen test set. Once again the FNR value is low at 0.3%, while the false alarm rate is 12.9%.

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	156	99.4	97.5	2.5	0.0
SVM (const. width)	156	99.8	85.3	12.9	1.8
SVM (av. width)	156	81.7	79.1	0.3	20.6
SVM (sp. width)	156	90.3	86.9	0.3	12.8

Table 5.4: Dual Class: comparison of classification performance with the combined statistical and spectral dataset (156 features)

5.4.2 Dual Case: Machine B

Table 5.5 shows the results for the experiments run with the spectral data from machine B. As can be seen, all the classifiers manage to achieve 100% accuracy on both the test and training sets, which would suggest that the fault data and the normal data is well separated in feature space, making it easy to classify correctly.

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	33	100	100	0.0	0.0
SVM (const. width)	33	100	100	0.0	0.0
SVM (av. width)	33	100	100	0.0	0.0
SVM (sp. width)	33	100	100	0.0	0.0

Table 5.5: Dual Class: comparison of classification performance with the spectral dataset from machine B (33 features)

5.4.3 Multiclass Case: Machine A

Table 5.6 shows the results for experiments carried out using the spectral dataset with the six class target data. As can be seen, the performance level of the different classifiers is much closer with the six class data than the dual class data. The worst performance comes from the constant width SVM, which achieves 99.4% on the training set, and 90.3% on the test set, with a FNR rate of 9.0%, and an FA rate of 0.5%. This level of performance is comparable with that of the constant width SVM with the dual class target data. The MLP manages to reach 100% on the training set, and 98.8% on the unseen test set. There is a low FNR rate of 0.7%, and there are no false alarms. The two modified SVMs manage to beat the MLP however; the averaged width SVM manages 100% on the training set, and 99.0% on the test set, with a low FNR rate of 0.4%, and an even lower FA rate of 0.1%. The specific width SVM manages to better this performance, reaching 100% on the training set, and 99.2% on the test set, with no false alarms, and an FNR rate of 0.6%. The performance of the SVMs on the multiclass data is much higher than that using the dual class data.

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	66	100	98.8	0.7	0.0
SVM (const. width)	66	99.4	90.3	9.0	0.5
SVM (av. width)	66	100	99.0	0.4	0.1
SVM (sp. width)	66	100	99.2	0.6	0

Table 5.6: Multiclass: comparison of classification performance with the spectral dataset (66 features)

Table 5.7 shows the results for the combined statistical dataset. The specific

width SVM has the highest performance of the classifiers, managing 100% on the training set and 99.2% on the test set. The FNR is low at 0.8%, and there are no false alarms. The worst result comes from the constant width SVM, which, while managing 100% on the test set, only manages 80.7% on the test data. The FA rate is low at 1.0%, however the FNR rate is high, at 17.5%. The averaged width SVM manages 100% on the training data and 96.5% on the test data, with a FNR of 2.2%, and a false alarm rate of 0.5%. The MLP has the second highest performance, managing 98.9% on the training set, and 96.8% on the test set. There are no false alarms, although the FNR rate is 2.6%.

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	90	98.9	96.8	2.6	0.0
SVM (const width)	90	100	80.7	17.5	1.0
SVM (av. width)	90	100	96.5	2.2	0.5
SVM (sp. width)	90	100	99.2	0.8	0.0

Table 5.7: Multiclass: comparison of classification performance with the combined statistical dataset (90 features)

The confusion matrix for the constant width SVM is shown in table 5.8. The inner race fault and the rolling element faults have suffered badly as a result of these classification problems, and the inner race fault only manages a level of 30% accuracy on it's own data, while the rolling element fault classification accuracy falls to 70.6%. This is of concern, however the most likely cause is due to the equal sizes of the different hyperspheres. The two normal conditions are highly localised in feature space,, and it is likely that in order to enclose the bulk of the fault categories with the hypersphere, the radius for the normal cases

Classification Success (%)							
		Actual Condition					
		NO	NW	IR	OR	RE	CA
Perceived Condition	NO	**90.6**	0.6	66.9	0	29.4	5.0
	NW	3.1	**99.4**	0	0	0	3.8
	IR	0	0	**32.5**	0	0	0
	OR	0	0	0.6	**100**	0	0
	RE	0	0	0	0	**70.6**	0
	CA	6.3	0	0	0	0	**91.2**

Table 5.8: Classification results for the constant width SVM, using the combined statistical data.

is much larger than what is actually required to enclose the data; as a result, the normal spheres start to enclose data from the other classes. There is also a lesser degree of misclassification between the cage fault and the normal condition, however this only amounts to 6.3% of the normal (NO) category, and no other misclassifications are made in favour of the cage (CA) fault, giving rise to the lower false alarm count.

Table 5.9 shows the results for the combined statistical and spectral dataset. The specific width SVM has the best performance, managing 100% on the training set, and 99.4% on the test set. The FNR rate is extremely low at 0.5%, and there are no false alarms. The second best performance comes from the averaged width SVM, which achieves 100% on the training set, and 97.9% on the test set. The false alarm rate is small at 0.1%, while the FNR rate is slightly higher at 0.7%. The MLP manages 99.8% on the training set, and 97.0% on the test set, with no false alarms, and a FNR value of 2.3%. The worst performance of the classifiers comes from the constant width SVM, which while managing 100% on the training set, manages only 84.2%, with a relatively high FNR rate of 15.1%, and a small FA rate of 0.5%.

Examining the confusion matrix for the constant width SVM (not given here)

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	156	99.8	97.0	2.3	0.0
SVM (const. width)	156	100	84.2	15.1	0.5
SVM (av. width)	156	100	97.9	0.7	0.1
SVM (sp. width)	156	100	99.4	0.5	0.0

Table 5.9: Multiclass: comparison of classification performance with the combined statistical and spectral dataset (156 features)

shows the misclassification problem to be the same as that for the spectral data, with a large proportion of the mistakes being caused by the excessive diameter of the hyperspheres belonging to the two normal classes. Table 5.10 shows the confusion matrix for the specific width SVM, showing the breakdown of the misclassifications made. As can be seen, the first four categories are classified perfectly, and a very small proportion (2.5% in total) of the rolling element fault are misclassified, while only 1.2% of the cage faults are misclassified . Examining the misclassifications, it can be seen that the majority of misclassification is caused by the normal condition, suggesting that either the data is not entirely separable, or that the scaling factor applied (scf=9) is slightly too large. Unfortunately, the results for scale factor of 8 are slightly worse, suggesting that there is some degree of inseparability in the data.

5.4.4 Multiclass Case: Machine B

Table 5.11 shows the results for the multiclass experiments run using the data from machine B. The best results comes from the specific width SVM, which manages 100% on the training set, but only 87.5% on the test set. Despite

Classification Success (%)							
		Actual Condition					
		NO	NW	IR	OR	RE	CA
Perceived Condition	NO	**100**	0	0	0	1.9	1.2
	NW	0	**100**	0	0	0	0
	IR	0	0	**100**	0	0.6	0
	OR	0	0	0	**100**	0	0
	RE	0	0	0	0	**97.5**	0
	CA	0	0	0	0	0	**98.8**

Table 5.10: Classification results for the specific width SVM, using the combined statistical data.

the low accuracy, the number of misclassifications between the fault and normal categories is low, with no false alarms and a low FNR of 1.8%, indicating that most of the misclassification is occurring between the different fault conditions. The constant width SVM has the second highest performance, again managing 100% on the training set, and 85.7%. Interestingly, the FNR rate and the FA rate are both zero, indicating again that all the confusion is occurring within the fault classes. The averaged width SVM has a training success rate of 100%, with a test success of 83.9%. Once again the FNR is 1.8%, while the FA rate is 0%. The MLP has the worst performance of all the classifiers, managing 100% on the training set, but only 76.8% on the test set. despite this poor score, the misclassification between the fault and normal classes is no worse than the averaged or specific width SVMs, with an FNR of 1.8% and no false alarms.

The results of all the classifiers on the unseen test set are poor, with no classifier managing to exceed 88% on the unseen test set, while all manage 100% on the training set. This would suggest either that the training data is insufficient to allow the classifiers to build up a good model for generalisation, or that the features used contain insufficient information in them to allow good discrimination between the different fault conditions.

	No. Features	Training Perf (%)	Test Perf (%)	Fault Not Recognised (%)	False Alarm Rate (%)
MLP	33	100	76.8	1.8	0.0
SVM (const. width)	33	100	85.7	0.0	0.0
SVM (av. width)	33	100	83.9	1.8	0.0
SVM (sp. width)	33	100	87.5	1.8	0.0

Table 5.11: Multiclass: comparison of classification performance with the spectral dataset from machine B (33 features)

Classification Success (%)						
		Actual Condition				
		NO	OF	CA1	CA4	WO
Perceived Condition	NO	**100**	0	8.3	0	0
	OF	0	**100**	0	0	0
	CA1	0	0	**75**	8.3	0
	CA4	0	0	8.4	**66.7**	16.7
	WO	0	0	8.3	25	**83.3**

Table 5.12: Classification results for the averaged width SVM, using spectral features of machine B.

Table 5.12 shows the classification breakdown for the averaged width SVM (the MLP having been covered in section 4.3.4 and Table 4.19). Most of the misclassifications occur among three of the four fault categories, with the outer race fault being easy to characterise. It is interesting to note that with the CA4 fault, the majority of misclassifications are not with the CA1 class (8.3%), but rather the worn (WO) class (25%). The CA1 data seems to be the most widely distributed about the different classifications, with mistakes being attributed to the normal (NO - 8.3%), CA4 (8.4%) and worn (WO - 8.3%) classes. The WO class also has some mistakes, with 16.7% being classified as belonging to the CA4 class.

5.4.5 Support Vectors

Examining the number of support vectors used in the different SVMs can give a lot of information about how the vectors are actually used, and also give an indication of the complexity of the data in the transformed feature space. Where the boundary is complex, a lot of support vectors will have to be used, while less vectors imply that the boundary is relatively easy to define, and consequently that the performance will be relatively good. Taking as an example the results from the combined statistical and spectral dataset, tables 5.13 and 5.14 show the different numbers of support vectors required by each class for the three different SVM types considered. As can be seen, the number of support vectors varies considerably between both different classes and the different types of SVM. With the dual class data, the constant width SVM uses the same number of support vectors to define the boundary; this is to be expected, as both classifiers are trying to find the same boundary in kernel space using hyperspheres the same size, and so to define the boundary both use the same number of support vectors. It is interesting to note that the averaged width SVM uses substantially less vectors to define the boundary for the fault case than the no fault case, using only 116 vectors for the fault case, but 532 for the no fault case. The scaling factor that the algorithm selected is also large, at scf=9. This would suggest that in order to enclose the no fault data, more support vectors are required because the average width enclosure is not enclosing as large a volume as the constant width hypersphere. The specific width SVM uses more support vectors (709) than the averaged width SVM (648), but less than the constant width SVM (934). The increase in the number of support vectors required by the specific width is probably due to the ellipsoid nature of the enclosure. The narrowness of the function in certain dimensions means that more support vectors are required

to cover the same space as one averaged function, depending upon the difference between the averaged standard deviation and the standard deviation in that dimension. It is also interesting to note that the scaling factor selected by the algorithm was the largest, at scf=9.

In the multiclass case, the effect of the three different SVM types on the number of support vectors is different. The constant width SVM uses a similar number of support vectors to create the partition for each class, varying between 442 and 500. This behaviour is slightly different from the dual class case, as in the dual class case, the data being partitioned by each SVM is identical, and so the boundary is also identical; however in the multiclass case, the dataset for each class changes, as each individual one is partitioned from all the others present. This explains why a different number of support vectors are required for each to partition the data correctly for all three types of SVM. As can be seen from the different values for the IR and RE faults, both the averaged and specific width SVMs require a substantially lower number of support vectors than the constant width SVM does for the same class, implying that the hyperplane is actually much simpler when the size of the enclosures is allowed to vary in proportion to the data itself. In the case of the IR and RE faults, comparing the orbit plots (figure 3.3), it can be seen that the two faults are very distinct from the other fault conditions; they probably occupy very different positions in feature space from the other conditions, and as a result, it will be fairly straightforward to define a boundary between the different cases and the rest of the data. It is interesting to note that again the averaged SVM uses the lowest number of support vectors (2503) to carry out the classification, although in this case the specific width SVM uses the greatest number of support vectors (3185). In the multiclass case the specific width SVM requires a larger number of support vectors than the other SVMs for the same reasons as the dual class SVM . Once again, the scaling

factors chosen by the algorithms for the averaged width and specific width SVMs are relatively large, at 8 and 9 respectively.

Generally speaking, the total number of support vectors increases as the complexity of the SVM increases; the specific width SVM requires more support vectors than the constant width or averaged width SVMs to cover the same area of feature space, due to the narrower enclosure used. The averaged width SVM requires the least support vectors as it is able to take advantage of the different sizes of the classes in feature space, and turn this to its advantage by using less support vectors, at the cost of an slightly lower classification success due to misclassification caused by the constant diameter hypersphere.

	No. Support Vectors			σ	Scaling
	No Fault	Fault	Total	Value	Factor
SVM (const. width)	467	467	**934**	1.1	-
SVM (av. width)	532	116	**648**	-	9
SVM (sp. width)	595	114	**709**	-	9

Table 5.13: Dual Class: Number of support vectors used by each SVM in classifying the combined statistical and spectral data, with σ or scaling factor values, as relevant.

	No. Support Vectors							σ	Scaling
	NO	NW	IR	OR	RE	CA	Total	Value	Factor
SVM (const. width)	500	458	442	442	442	485	**2769**	1.0	-
SVM (av. width)	602	565	102	425	134	675	**2503**	-	8
SVM (sp. width)	784	634	177	582	247	761	**3185**	-	9

Table 5.14: Multiclass: Number of support vectors used by each SVM in classifying the combined statistical and spectral data, with σ or scaling factor values, as relevant.

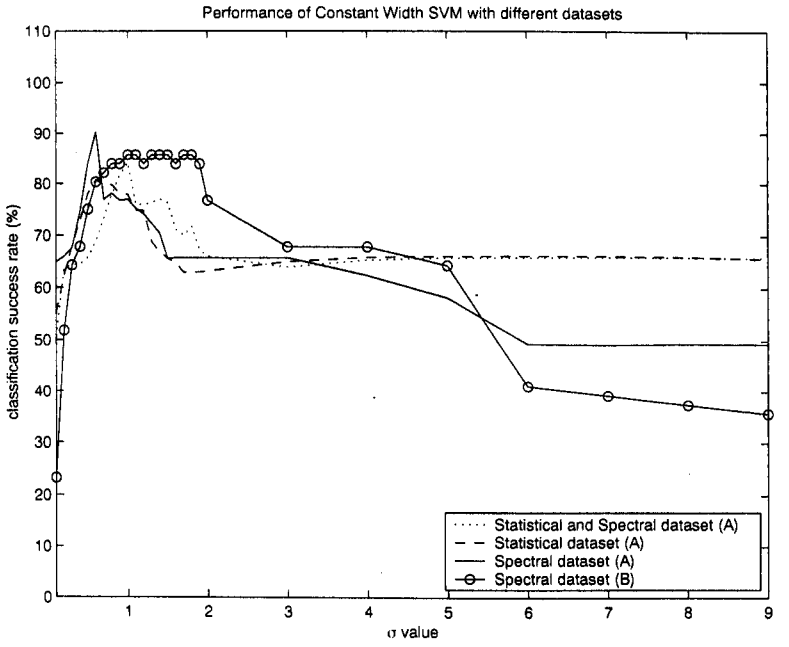


Figure 5.3: Classification performance for different values of σ with the constant width SVM in multiclass case.

5.4.6 Width Parameters and Scaling Factors

Figures 5.3, 5.4 and 5.5 show the impact of different width and scaling factors upon the performance of the three different types of SVM, for the three datasets associated with machine A, in the six-class case, and the five-class case from machine B.

Figure 5.3 shows the performance of the constant width SVM for varying value of width value σ. As can be seen, there is a great deal of variation in the results, dependent upon the value of σ. In all three cases, the best performance of the constant width SVM occurs when σ is between approximately 0.5 and 1.5; once the upper value is exceeded, then performance drops off rather quickly. The spectral data from machine B behaves differently; best performance occurs when σ is between 1 and 2.5. In the case of the two datasets containing statistical data, the performance drops to approximately 66%, which examination of the results shows to be caused by the two normal conditions being classified as cage faults.

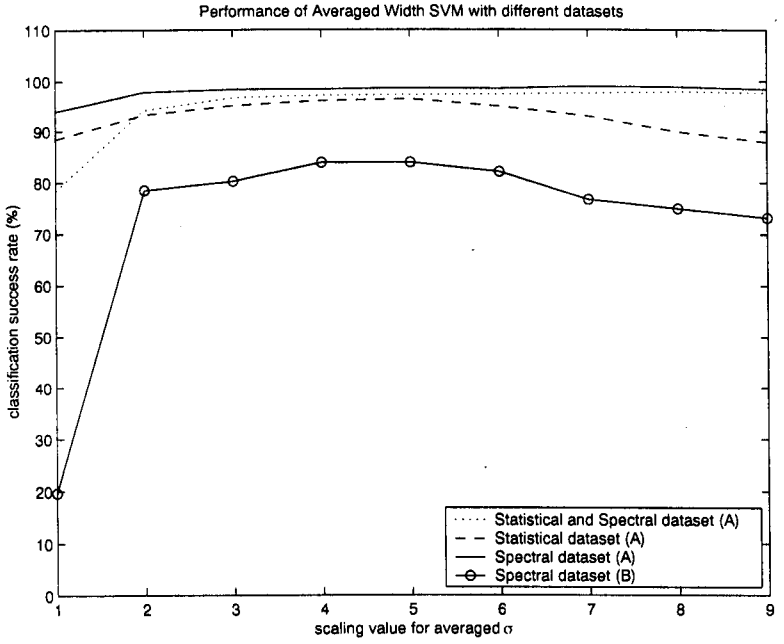


Figure 5.4: Classification performance for different scaling values with the averaged width SVM in multiclass case.

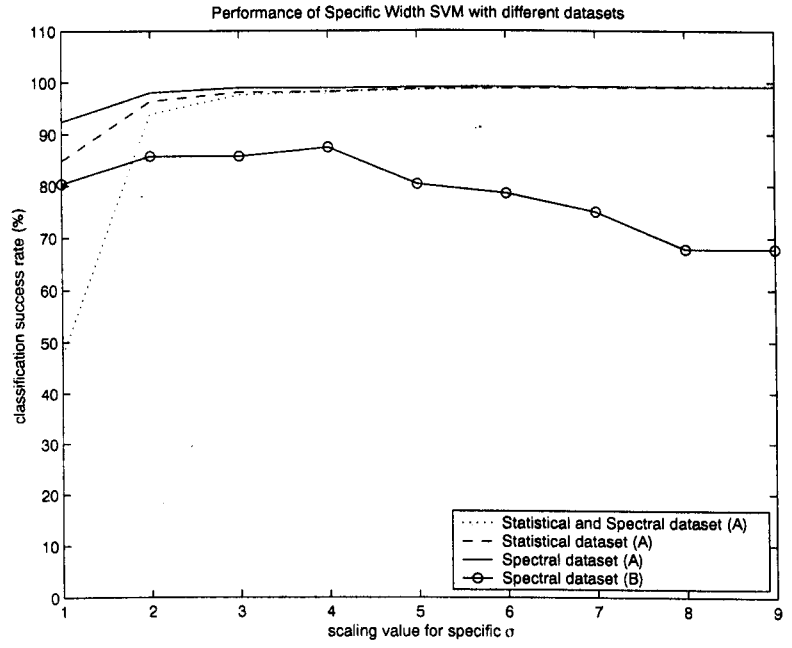


Figure 5.5: Classification performance for different scaling values with the specific width SVM in multiclass case.

All of the fault conditions are classified correctly. In the case of the spectral data from machine A, the result converges to approximately 50%, which is caused by the cage fault again incorrectly classifying the two normal conditions and the outer race fault all as cage faults. The performance of the machine B spectral data drops to approximately 35%, caused by fault conditions being classified as normal conditions.

Figure 5.4 shows the variation of classification performance with the scaling factor value. As can be seen, the performance of the averaged SVM is much less dependent upon the value of the scaling factor than the constant width SVM is with regard to the kernel parameter value. For values of scaling factor in the interval 3-5, the performance is high for all three datasets. For the machine B dataset, the performance remains fairly constant for scale factor values of between 2 and 9, peaking around values of 4 or 5.

Table 5.5 shows the variation of classification performance with scale factor for the specific width SVM. As can be seen, the specific width SVM is very insensitive to variation of the scaling factor on all the datasets from machine A, with all three datasets improving classification (and converging at approximately 99%) as the scaling factor is increased. Machine B however peaks at scf=4, and then falls away to a level of around 67% at scf=9. The poor performance by the machine B dataset at large scale factor values is caused by the normal condition classifying a large number of the fault conditions as normal. This problem would be likely to occur if the data was very tightly clustered together with the fault conditions, and the normal condition possessed a larger variance. If the different closures were in similar areas of hyperspace, then there would be interplay between the different hyperellipsoids, causing the misclassification.

It is interesting to note that the constant width SVM records the highest performance of the three classifiers for the spectral dataset from machine B. The

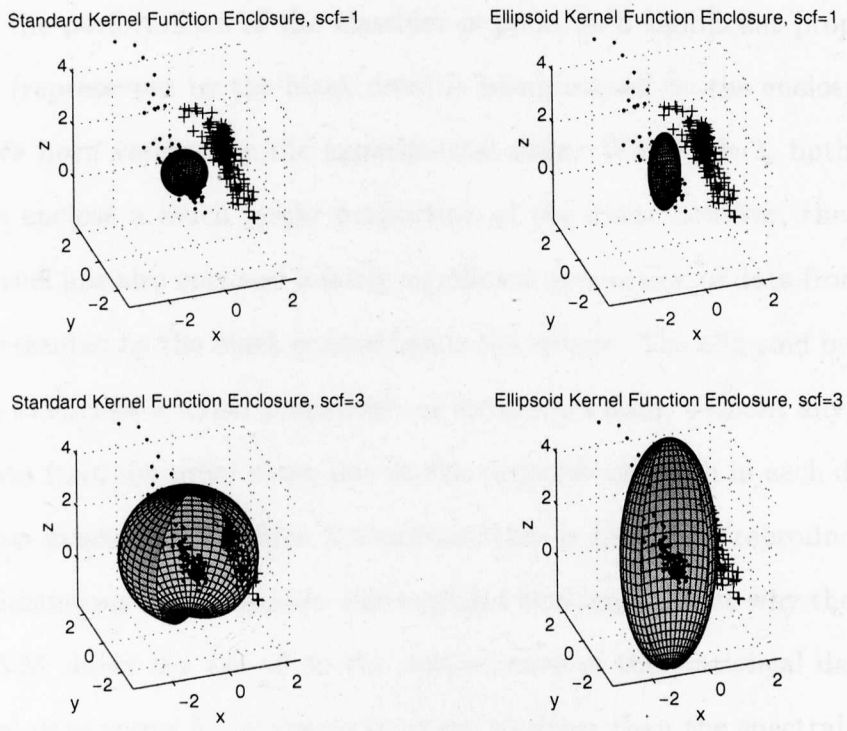


Figure 5.6: The effect of different scaling values upon enclosure for both the averaged and specific width kernels

reason for this is the way in which the different kernels interact with each other when the data is non separable. Where two closures intersect, there will be a conflict between the two enclosures. In the case of the constant width SVM, the intersection between the different spheres will be an equal split; however, in the case of the modified kernels, the boundary between the two classes will actually be a curved surface, as the different sizes of the enclosures mean that the smaller enclosure has a greater influence than the larger, and as a result, the boundary changes. This relationship is complex, and requires further work to determine what is happening.

To illustrate the effect of the different enclosure types, figure 5.6 shows the different ways in which the two modified SVM kernels enclose data. The effect of different scaling factors is shown as well. As can be seen, at scf=1, both different functions are unable to enclose any significant part of the data, and as

a result, the performance of the classifier is poor, as a significant proportion of the data (represented by the black dots) is being missed by the enclosure; these ideas have been verified by the experimental data. When scf=3, both different functions enclose a much larger proportion of the data; however, the averaged width kernel has also enclosed a fairly significant proportion of data from another class represented by the black crosses inside the sphere. The ellipsoid by contrast, manages to enclose a larger proportion of the correct data, without any enclosure of the data from the other class, due to the variation of width in each dimension. While this example is in three dimensions, this is the effect reproduced in the higher dimensions of the dataset, and explains to a large extent why the averaged width SVM shows the fall off in the performance of the statistical data, as the statistical data seems to be clustered closer together than the spectral data.

5.5 Discussion

In general, the multiclass results are better than the dual class results, with the SVM managing to equal or outperform the MLP on the multiclass datasets. However, that result in itself raises questions as to why the SVMs perform so well on the multiclass data, but relatively poorly on the dual class data, while the MLP maintained a fairly consistent level of performance with all the data from machine A in both dual and multiclass experiments. Much of the problems in the dual class case probably stems from the way that the different classes are distributed in feature space. If it is the case that the normal cases are located in the middle of the fault clusters, then that would explain the lower performance of the dual class data, as the hypersphere enclosing the fault class will span the no fault class. Figure 5.7 shows how this might happen in practice. In the dual class case, as the fault (class 2) function is superimposed on the no fault case (class

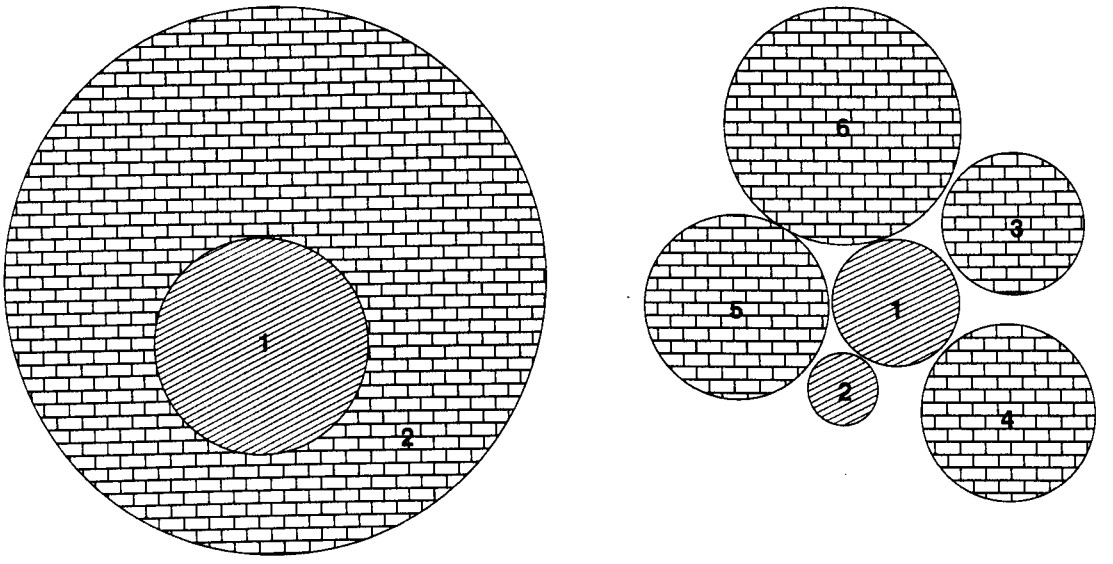


Figure 5.7: The different enclosure mechanisms at play in the dual and multiclass cases

1), then the likelihood of misclassification is much higher. In the multiclass case, where each condition is localised in space, enclosing each of the different fault categories (3-6) is distinct from the other data, while the two normal categories (1 & 2) are also distinct in the same space as was occupied by class 1 in the dual class case.

In this respect, it would seem that the SVM is much more adept at classifying the multiclass type problem; however, much of the problems that arise in the dual case scenario are caused by the overlapping of clusters in feature space. Some of the features being used in the feature extraction stage are forcing the data clusters to overlap. If some technique could be used to determine which feature are providing useful information, and reject those features which cause overlapping clusters to be created, then the classification performance could be improved further, which would help the performance on the dual class problem significantly.

The averaged width and specific width SVM seem to be fairly robust classifiers, certainly more robust than the constant width SVM. While not as good at the dual class problem for the reasons discussed earlier and as a result perhaps not as well suited to the fault/no fault problem, SVMs would appear to have something to offer in terms of improved performance over the MLP in the multiclass case. Even with the multiclass data from machine B, which falls short of 90% performance on unseen data, the SVM still manages to return a score that is better than the performance (Table 4.18) of the best MLP, and approximately equal to the RBF neural network in terms of performance, although less than the PNN. This lower performance on the multiclass dataset from machine B may well be caused by either insufficient information in the features extracted from the raw data, or one or more features providing information that misleads the classifier, causing overlap within the different clusters. A process of feature selection will determine to what extent the overlap is an issue.

The choice of the best SVM for a classification task is a compromise decision; for multiclass work, there is a tradeoff between accuracy and computational complexity. The averaged width SVM uses less support vectors to achieve a better performance than the constant width SVM, however the specific width SVM is capable of beating the performance of both SVMs, at an increased computational cost during training and operation. The decision to use one or other would depend on the requirements of a user - i.e. best overall performance no matter the computational cost, or best performance for minimal resources.

5.6 Summary

This chapter has introduced the use of the SVM as a classifier for bearing faults. The basic concepts behind the support vector machine were introduced, along

with an overview of the different training mechanisms currently available. A discussion of the significance of the kernel, along with some of the problems that make MCM problems ill posed was raised, and as a result, two modified kernels were proposed that improve the generalisation of the networks significantly, at the cost of additional computational complexity. The modified kernels were shown to perform well in a range of multiclass problems, on terms which are comparable with MLP based classification, and in several cases, exceeding the performance of the MLP. On dual class problems, the MLP continues to have a better performance than that of the SVM, although a hypothesis has been advanced which explains the nature of the problem, and the distribution of data within feature space.

Chapter 6

Feature Selection

Feature selection techniques for classification are intended to meet a number of needs simultaneously [99]:

- To reduce the cost of extracting features.
- To improve classification accuracy.
- To improve the reliability of the estimate of performance.

With the large number of different features that are being considered in some of the datasets used here (i.e. up to 156 features), calculating the features can be a time consuming task; obviously, being able to calculate only those features needed for classification is much preferable. The problem is determining which features provide information, and which provide uncorrelated data that only serves to confuse the classifier.

Genetic Algorithms (GAs) provide an ideal mechanism for feature selection in high dimensional spaces. Using Darwinian evolutionary principles, the GA is able to search quickly through a large number of possible solutions, and arrive at an answer in a much quicker time than many other comparable feature selection

algorithms. A recent survey [99] showed that for many dimensional data (> 50), the GA is one of the most efficient feature selection techniques available.

The use of GAs for feature selection has been known for some time [100,101]. As a technique, the GA has been shown to be extremely robust, and providing good accuracy without requiring any prior knowledge.

6.1 Motivation in Condition Monitoring: Why is Feature Selection Useful?

There are many potential advantages to be gained by performing feature selection on an MCM type problem. Modern MCM and supervisory systems can have any number of inputs, ranging from tens up to hundreds of channels. A lot of this data creates an information overload condition for the operators, and very often it is difficult to gain an overview of the whole system, and determine exactly what is happening. Using feature selection in conjunction with classifiers allows the classifiers to act as a data abstraction layer, interpreting the raw data to provide *information* that can be digested easily, and used to make meaningful assessments of the condition of the system. Using an automatic feature selection strategy can find relationships between different parameters that might not be intuitive to the casual observer. The complex interrelationships of the different channels can allow machine condition to be ascertained more reliably than is possible by human operators, and allows further high level integration with supervisory systems such as expert systems, which can provide wide-scale monitoring of the machine system as a whole, rather than the sub-assemblies that the classifiers monitor.

The advent of small silicon based accelerometers and velocimeters combined with modern classifiers lend themselves to the creation of small integrated chips

that may be incorporated directly into the fabric of a machine, giving it the ability to return information to either an internal control system or an external monitoring system. However, one of the main problems facing the use of classifiers in an on-board system is that the amount of computation required should be small. To be feasible, the classifier must be relatively compact, in order that a small, cost effective processing unit may be used. Feature selection provides a means to reduce the computational workload required for classification by reducing the number of features needed for classification.

6.2 General Comments

The GA used in this experiment is a simple GA, as proposed by Goldberg [55]. The experiments are programmed in C++, using the GAlib library [59] available over the internet. The genome class used to implement the form above is the GARealAlleleSetArray, which allows the use of bounded string values, with user defined step sizes.

The GA uses a population size of 10 individuals, starting with randomly generated genomes. This is a tradeoff between a comprehensive, broad search through a large population, and achieving the desired target within a reasonable computation time. An elitist population model is used, meaning that the best individual in the previous population is kept in the new population, and this prevents the performance of the GA worsening as the number of generations increase, meaning that once performance reaches a certain point, it should not deteriorate below that level.

6.3 Feature Selection & Encoding

The GA controls feature selection through the values of the genome. The classifier function used to train the network is written to accept a genome of one of the standard forms given below, and return the "score" of the classifier with that combination of features.

6.3.1 ANN Genome

For the ANNs, a simple genome string was used. For a training run requiring N different inputs to be selected as a subset of Q possible inputs, the genome string would consist of $(N + 1)$ real numbers. Each of the first N numbers (x) in the genome is constrained as $0 \leq x \leq (Q - 1)$, whilst the last number, (x), is constrained to lie within the bounds $2 \leq x \leq S$, where S is the maximum number of neurons allowable in the first layer (a predetermined value). This means that any mutation that occurs will be bounded within the limits set at the definition of the genome. This arrangement is shown in (6.1).

$$\left[\begin{array}{l} \left.\begin{array}{l} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{array}\right\} \; 0 \leq x < \text{Q-1} \\ \left.\begin{array}{l} x_{N+1} \end{array}\right\} \; 2 \leq x \leq \text{S} \end{array}\right. \tag{6.1}$$

6.3.2 Constant Width SVM

For the constant width SVM, a slightly different genome was used. For a requirement of a system with N inputs, the genome string is still $(N + 1)$ units long. However, for the SVM, the last value of the genome represents the σ-width value of the Gaussian function, rather than the number of hidden neurons. This was bounded between 0.1 and 2.0, with a step-size of 0.1 for these examples.

6.3.3 Averaged Width SVM

Two different genome strings were used; for the dual class SVMs a genome string that only specified the features to be tested was used, with the scaling factor fixed at scf=3. For the multiclass case, a slightly modified version of (6.1) was used. The last value of the genome string determines the scaling factor applied to the σ value, and this was allowed values between 1 and 9, in integer steps.

6.3.4 Specific Width SVM

Once again, two different genome strings were used; for the dual class case, the scale factor was fixed at scf=3, and the features only (i.e. no determination of scaling factor) were selected by the GA, while in the multiclass case, the same modified string as the averaged width SVM was used. The last value of the genome string determines the scaling factor applied to the σ value, and this was allowed values between 1 and 9, in integer steps.

6.3.5 Mutation Operation

The mutation operator used is real Gaussian mutation, with a probability of mutation equal to 0.2. For an allele string a, and using a probability of mutation

$p_m \in [0, 1]$, the mutation operation can be expressed as:

$$\text{If } \chi_i \leq p_m \Rightarrow a_i^{'} = a_i + \psi_i \mid a_{min} \leq a_i^{'} \leq a_{max} \tag{6.2}$$

where $a_i^{'}$ represents the modified allele member, $\chi_i \in [0, 1]$ is a randomly sampled uniformly distributed variable, ψ_i is a randomly sampled Gaussian distributed variable with 0 mean and a standard deviation of 1. Values of a_{min} and a_{max} represent the bounding limits of the allele values; if a value exceeds one of the limits, then it will be set to the maximum or minimum as relevant. The value of 0.2 was chosen after a number of trials which showed that the value of p_m was high enough to allow a good traverse of feature space without compromising of a convergence to a good solution.

6.3.6 Crossover

The crossover operator used is the uniform crossover, with the probability of crossover set to 0.75. If there are two strings, $a_{S,i}$ and $a_{R,i}$, and given a probability of crossover $p_c \in [0, 1]$, then the crossover operation for each individual element, a_i may be expressed as:

$$\text{If } \gamma \leq p_c \Rightarrow a_i^{'} = \begin{cases} a_{S,i} \ , & \chi_i > 0.5 \\ a_{R,i} \ , & \chi_i \leq 0.5 \end{cases} \tag{6.3}$$

where $\gamma, \chi \in [0, 1]$ are both randomly sampled uniformly distributed variables. No crossover will take place if $\gamma > p_c$.

The p_c value was set high after initial exploratory work showed that a value of 0.75 gave good rate of change to the population. Due to the nature of the

data, a high rate of crossover was desired in order to ensure that the comparison of different features traverses feature space relatively quickly.

6.4 Fitness Function

The fitness function used in the GA simply returns the number of correct classifications made over the whole dataset. No direct penalisation is made for incorrect classification here, just that the classification score will be correspondingly lower. The validation set is used to provide some form of evaluation of the generalisation properties of the network. While the performance could be rated on only the performance of the validation or training sets, it was felt that this would give an inadequate measure of the performance of the GA on unseen data, as this will be the scenario that will be encountered most often in real life. Due to the nature of the machine system, it is not always easy to choose training data that is completely representative of likely characteristics of the fault data, and so the performance of the GA/classifier is measured over the test set in order to allow the GA to be more representative in it's selection of features.

More complex forms of fitness function, either involving incorrect classifications or sum squared error or other factors could be used, to determine the performance of each network trained, however the performance achieved using this comparatively simple fitness function shows that they are not needed to achieve good results in this application.

6.5 Training

Training was carried out using first the ANN and then the same experiments were repeated with the SVM using constant width Gaussian functions, where the σ

parameter is found by iteration to one decimal place, using the σ_{av} parameter, with a range of different scaling factors from 1 to 9, and also the specific width kernel, again with scaling parameters from 1 to 9. Three different datasets from machine A were used for the dual class case, one consisting all the statistical data (90 features), one with only spectral data (66 features), and a combined dataset (156 features), consisting of both statistical and spectral features. For all four classifiers, training was stopped when the performance on the validation set dropped in relation to that of the training set. This sequence of events was repeated for the single dataset for machine B.

For the multiclass case, time constraints meant that it was only possible to test the combined statistical and spectral dataset from machine A, and the single spectral dataset from machine B. These were tested with all three SVM kernels, and also the MLP. In both cases, the GA was allowed to select a range of input sizes, from 4 to 12 inputs, as a subset of the total dataset.

6.5.1 ANN Training

The ANNs were trained using the backpropagation algorithm with adaptive learning and momentum. Training was stopped through use of a validation set, with the stop occurring when the performance on the validation set started deteriorating rather than improving. The ANNs were trained for various different sizes of hidden layer, from 2 to 15 neurons, and the best one was selected.

6.5.2 SVM Training

Training of the SVM is carried out using the Kernel Adatron (KA) algorithm [93]. KA has been shown to be a relatively fast way of training SVMs, with a speed comparable to that of Platt's SMO technique [89]. The advantage that the KA

algorithm offers is that it allows the use of a validation set to stop training, and thus maintain good generalisation. The SMO technique by comparison, finds a global minimum solution without the use of a validation set, and this can mean that the generalisation performance is slightly worse than that of the KA algorithm. Research in this area is still progressing, and other training methods and modifications are being developed, which may be more computationally efficient and simpler to implement [90, 102].

SVMs were created using the kernel function described in equation 5.12, σ values were for the constant width SVM, limited to one decimal place. For the averaged width training runs, the σ values were determined using equation 5.15, and various scaling values were used, ranging between 1 and 9 in integer steps. A similar procedure was used for the specific width kernel (equation 5.16), with scaling values between 1 and 9 permitted.

6.5.3 GA Training

Having trained the ANNs and both different forms of SVM with all the available datasets, training runs were then carried out using the GA to select different combinations of input features for each classifier, in an attempt to improve performance by selecting the most important feature set for a given classifier. Each selection method was run until the population converged; for the ANN, 40 generations was found to be sufficient, while the SVM, which seems to be more sensitive to changes, a total of 60 generations were used, with a population size of 10 members. The experiments were then repeated for each of the different datasets.

6.6 Results: Genetic Algorithm with ANN

Initial work using GA based feature selection was carried out using only the MLP classifier. Various experiments were carried out in an attempt to determine what was an acceptable number of generations to run the algorithms for. To this end, a series of experiments were carried out, running for 20, 40 and 60 generations in total, using all six datasets from machine A. The MLP was selected for use as previous work had shown it to be the most robust (and compact) classifier from those considered in chapter 4.

6.6.1 Genetic Algorithm with ANN after 20 Generations

Table 6.1 shows a comparison between the results generated after a run of 20 generations using the combined genetic algorithm ANN program (GA/ANN), and the stand-alone ANN program. The table shows specific results for the MLP considered with each dataset covered in chapter 4, and compares these against the best result achieved using the GA/ANN system. The third subset of columns show the mean performance and the range of performances achieved, expressed as percentage accuracies.

On examining the results on a like for like basis, it can be seen that all cases have a success rate in excess of 92.0%, and the best GA/ANN solution has a markedly higher classification rate than the equivalent "straight" ANN. The number of inputs selected in each case is significantly smaller than the complete feature sets used in each set, the selected features are subsets of the original datasets, and thus any information available to the small GA/ANN is also available to the larger one.

Examining the results for the mean performance, it can be seen that every single result achieved a higher performance using the GA/ANN than the stand-alone

ANN. This is also borne out by the performance range figures, which show that in all cases, the inputs selected by the GA/ANN provide a superior performance to an ANN using all available inputs. As can be seen, the spectral dataset, which had the highest performance improves from 97.0% (using 66 inputs) to 99.7% (using only 8 inputs). This means that out of the total of 960 examples, only three have been misclassified, using only one eighth of the inputs. Interestingly, the high/low pass filtered data has undergone the largest improvement in its performance, jumping from 83.1% to 97.7%. This compares well with the other results. The fact that the high/low pass data is able to reach a value in excess of 97% implies that the information contained within the feature set is sufficient to allow accurate classification, without the need for extraneous information. Additionally, the number of inputs required to achieve this level of accuracy is one fifth of that used by the straight ANN.

Examining the classifications of the spectral feature set (tables 6.2 and 6.3) after 20 generations , a number of things can be seen. The classification of the network has improved markedly in that there are only a few mistakes. All faults are now classified as faults, and all normal conditions are classified correctly. While the actual classification between some of the fault conditions is not yet entirely right, the fact that there are no errors being made in the classification of faults as normal and vice versa is perhaps the most helpful criterion in the selection of condition monitoring systems.

Looking at the results for the high and low pass filtering (tables 6.4 and 6.5), the improvement in the classification from the straight ANN (table 6.4) is very evident. Four of the six conditions achieve 100% accuracy, while one of the other two categories (rolling element fault - RE) has remained as accurate, and the other (cage fault - CA) has deteriorated slightly. The only fault condition which is being confused with the "normal" conditions is the cage fault, and this

is the same problem as before. It may be that the spectral information by itself is insufficiently detailed to allow discrimination between these conditions at the lower speeds of rotation.

6.6.2 Genetic Algorithm with ANN after 40 Generations

Table 6.6 shows the performance of the different feature sets after running under the GA for 40 generations. The results are again an improvement over the previous set (table 6.1). The mean performance of every set has increased from the values achieved after 20 generations. Five of the six datasets have their best performance in excess of 97.5%.

The feature set using all the available training data has managed to achieve an accuracy of 100%, indicating accurate classification. This is achieved using only six inputs out of the possible 156. Using 9 neurons in the hidden layer, a relatively small network has been created that fulfils the criteria set earlier on. A network of this size would be ideal for a real-time implementation on a small chip or microcontroller.

6.6.3 Genetic Algorithm with ANN after 60 Generations

Table 6.7 shows the results after running experiments through to 60 generations and these still show a degree of improvement over the shorter runs. In almost all cases, the algorithms had converged by 60 generations. Comparing the results for 60 generations, it can be seen that the performance of the best networks for each dataset remained similar, except that it improved for the statistics only case.

In four cases (statistics only, differencing/summing, spectral and all data), the mean performance of the feature sets improved, while in the other two (all statistics and high pass/low pass), the mean performance differed by no more

	Straight ANN			GA with Best ANN			GA with ANN	
Data Set	No. Inputs	No. Hidden	Perf. %	No. Inputs	No. Hidden	Perf. %	Mean Perf. %	Perf. Range
Statistics only	18	14	84.4	6	9	95.8	92.0	90.2 - 95.8
High Pass/ Low Pass	36	7	83.1	9	8	97.7	96.0	89.2 - 97.7
Differencing/ Summing	36	12	89.3	9	7	97.0	95.6	94.5 - 97.0
All Statistics	90	7	83.1	5	5	97.6	95.0	89.4 - 97.6
Spectral Data	66	8	97.0	8	11	99.7	98.7	96.7 - 99.7
All Data	156	6	91.1	11	12	99.7	96.8	94.6 - 99.7

Table 6.1: Comparison between stand-alone ANN and GA with ANN after 20 generations, for all six data sets

Classification Success (%)							
		Actual Condition					
		NO	NW	IR	OR	RE	CA
Perceived Condition	NO	**98.8**	0	0	0	0	12.5
	NW	0	**100**	0	0	0.6	0
	IR	0	0	**98.8**	0	0	0
	OR	1.2	0	1.2	**100**	0	2.5
	RE	0	0	0	0	**99.4**	0
	CA	0	0	0	0	0	**85.0**

Table 6.2: Classification results for straight ANN, using spectral data only.

Classification Success (%)							
		Actual Condition					
		NO	NW	IR	OR	RE	CA
Perceived Condition	NO	**100**	0	0	0	0	0
	NW	0	**100**	0	0	0	0
	IR	0	0	**98.9**	0	1.1	0
	OR	0	0	0	**100**	0	0
	RE	0	0	1.1	0	**98.9**	0
	CA	0	0	0	0	0	**100**

Table 6.3: Classification results for GA/ANN after 20 generations, using spectral data.

Classification Success (%)							
		Actual Condition					
		NO	NW	IR	OR	RE	CA
Perceived Condition	NO	**22.5**	0	0	0	0	0.6
	NW	61.9	**100**	1.3	0	1.3	10
	IR	0	0	**88.8**	0	0	0
	OR	0	0	2.5	**100**	0.6	0
	RE	0	0	0.6	0	**98.1**	0
	CA	15.6	0	6.8	0	0	**89.4**

Table 6.4: Classification results for straight ANN, using high pass/low pass data.

Classification Success (%)							
		Actual Condition					
		NO	NW	IR	OR	RE	CA
Perceived Condition	NO	**100**	0	0	0	0	4.4
	NW	0	**100**	0	0	0	6.9
	IR	0	0	**100**	0	1.9	0
	OR	0	0	0	**100**	0	0.6
	RE	0	0	0	0	**98.1**	0
	CA	0	0	0	0	0	**88.1**

Table 6.5: Classification results for GA/ANN after 20 generations, using high/low pass data.

than 0.2%.

Once again, the feature set containing all the data is capable of reaching 100% accuracy, although it achieves this using a larger network (7 inputs and 15 neurons in the hidden layer) than that found for 40 generations (6 inputs and 9 hidden neurons). This is probably partially due to an inadequacy of the fitness function, in that there is no penalty term for larger sizes of hidden layer. Incorporating this into the fitness function would allow better results in this respect.

6.7 Results: GA and SVM

Having conducted a series of successful experiments with the ANN, it was decided to use the GA in conjunction with the SVM, in an attempt to determine whether the feature selection could improve the robustness of the classifier, and also hopefully improve the performance in the dual class case with data from machine A, which was shown to be very poor in section 5.4.1. For the dual class experiments, the GA was allowed to control the inputs selected for the SVM. Additionally, for the constant width SVM, the GA was allowed to select the width parameter. For the averaged and specific width experiments, the scaling factor was fixed at

Data Set	Straight ANN			GA with Best ANN			GA with ANN	
	No. Inputs	No. Hidden	Perf. %	No. Inputs	No. Hidden	Perf. %	Mean Perf. %	Perf. Range
Statistics only	18	14	84.4	12	7	95.4	93.9	91.7 - 95.4
High Pass/ Low Pass	36	7	83.1	6	10	97.8	97.6	97.4 - 97.8
Differencing/ Summing	36	12	89.3	6	10	97.5	96.8	96.1 - 97.5
All Statistics	90	7	83.1	7	14	97.7	97.1	96.0 - 97.7
Spectral Data	66	8	97.0	6	11	99.8	99.2	97.3 - 99.8
All Data	156	6	91.1	6	9	100	98.3	95.0 - 100

Table 6.6: Comparison between stand-alone ANN and GA with ANN after 40 generations, for all six data sets

Data Set	Straight ANN			GA with Best ANN			GA with ANN	
	No. Inputs	No. Hidden	Perf. %	No. Inputs	No. Hidden	Perf. %	Mean Perf. %	Perf. Range
Statistics only	18	14	84.4	9	5	97.0	95.5	94.2 - 97.0
High Pass/ Low Pass	36	7	83.1	8	10	97.7	97.4	97.0 - 97.7
Differencing/ Summing	36	12	89.3	6	8	97.6	96.6	96.1 - 97.6
All Statistics	90	7	83.1	7	9	97.4	96.6	95.7 - 97.4
Spectral Data	66	8	97.0	7	10	99.7	99.4	98.9 - 99.7
All Data	156	6	91.1	7	15	100	98.7	96.4 - 100

Table 6.7: Comparison between stand-alone ANN and GA with ANN after 60 generations, for all six data sets

scf=3.

For the multiclass case, the GA was allowed to select the inputs and the kernel parameter - which meant the GA had control of the width parameter for the constant width SVM, and also set the scaling factor values for the averaged and specific width SVMs. Unfortunately, time constraints meant that it was not possible to conduct an exhaustive evaluation of the effect of allowing the GA control of kernel parameters with both datasets, and so it was decided to attempt an experiment on each dataset. This makes it difficult to draw direct comparisons between the performance of the different techniques, however it acts as a preliminary investigation.

6.7.1 Dual Class Results: machine A

Examining the results from table 6.8, it can be seen that the performance with the spectral dataset is high, with the ANN classifier achieving results of 99% for the unseen test set without the use of feature selection. Again, the constant width SVM has the poorest generalisation performance, achieving 98.2% on the training set but only 89.4% on the test set. The averaged width SVM has good generalisation performance, with scores of 93.8% for the training set and 91.6% for the test set. The specific width SVM has the best performance of the SVMs without feature selection, managing to reach 96.6% on the training set, and 91.6% on the test set. After feature selection, it can be seen that the performances on the unseen test sets have all increased. The best performance is that of the GA/ANN, which using only 9 inputs, manages 100% on both the training and test sets. The GA with constant width SVM offers a training success of 99.9% and the test set performance of 98.6% using only 4 features from the possible 66. The performance of the averaged width SVM rises to 99.3% on training data,

Classifier	No. Inputs	Training Success (%)	Test Success (%)
ANN	66	100	99.0
GA/ANN	9	100	100
SVM (const. width)	66	98.2	89.4
GA/SVM (const. width)	4	99.9	98.6
SVM (av. width)	66	93.8	91.6
GA/SVM (av. width)	7	99.3	99.1
SVM (sp. width)	66	96.6	93.9
GA/SVM(sp. width)	5	98.9	99.1

Table 6.8: Dual Class: Performance with spectral dataset from machine A (66 features).

Classifier	No. Inputs	Training Success (%)	Test Success (%)
ANN	90	98.9	97.6
GA/ANN	12	99.9	100
SVM (const. width)	90	99.4	84.9
GA/SVM (const. width)	6	99.5	98.1
SVM (av. width)	90	81.8	80.4
GA/SVM (av. width)	4	97.9	97.4
SVM (sp. width)	90	85.5	85.6
GA/SVM(sp. width)	5	98.5	97.7

Table 6.9: Dual Class: Performance with combined statistical dataset from machine A (90 features).

and to 99.1% success on the unseen test set, using only 7 inputs. The specific width SVM performance rises to 98.9% on the training set, and 99.1% on the test set after feature selection, using only 5 inputs. All these performance levels are very high, and certainly after feature selection, the SVM would appear to offer a comparable level of performance to the ANN.

Examining the results from table 6.9, a number of things can be seen. Firstly, examining the results of the standalone ANN and SVMs, the contrast between the performance of the three different SVM classifiers is very obvious. The constant width SVM is the most inconsistent performer, with a good training performance

Classifier	No. Inputs	Training Success (%)	Test Success (%)
ANN	156	99.4	97.5
GA/ANN	4	100	100
SVM (const. width)	156	99.8	85.3
GA/SVM (const. width)	4	100	98.6
SVM (av. width)	156	81.7	79.1
GA/SVM (av. width)	6	98.2	99.2
SVM (sp. width)	156	90.3	86.9
GA/SVM(sp. width)	5	99.1	99.5

Table 6.10: Dual Class: Performance with combined dataset (156 features).

of 99.4%, however this is offset by the poor performance on the unseen test set of 84.9%, indicating a high degree of overtraining on the training set. The ANN generalises well, achieving 98.9% on the training set and 97.6% on the test set. The averaged width SVM achieves only 81.8% on the training set, and 80.4% on the test set, the worst performance of all the classifiers. The specific width SVM reaches 85.5% on the training set and 85.6% on the test set. The reason for the poor performance of the two modified SVMs is not immediately apparent. After applying the GA for feature selection, all the performance levels rise, in some cases quite dramatically. The GA/ANN manages to achieve success levels of 99.9% and 100% on the training and test sets respectively, using only 12 inputs out of the possible 90 in the feature set. All three SVM classifiers show strong improvements when run with GA feature selection, as well as display very good generalisation performance. The best performer of the SVMs after feature selection is the constant width SVM, which reaches 99.5% and 98.1% on the training and test sets respectively, using only 6 inputs. The specific width SVM is close behind, managing 98.5% and 97.7% respectively with 5 inputs, while the performance of the averaged width SVM is also high at 97.9% for the training set and 97.4% for the test set, using 4 inputs.

Table 6.10 shows the performance of the different classifiers in the combined statistical and spectral dataset, consisting of a total of 156 features. As can be seen, the performance of the standalone classifiers is slightly poorer than their performance on the purely spectral dataset, which might seem counter-intuitive at first, but the higher number of features combined with the information from the statistical set may be confusing the classifiers slightly. The constant width SVM performs well on the training set (99.8%), but suffers again on generalisation performance, managing only 85.3%. The specific width SVM has a performance level that has lower training success than the constant width SVM, but higher success on the test set, scoring 90.3% and 86.9% respectively. The averaged width SVM has the poorest performance, scoring 81.7% on the training set, and only 79.1% on the test set. This might suggest that some of the the statistical data has a very high variance within a feature, or alternatively that there is a high degree of overlap between the values of certain features between the two classes, fault and no-fault. This would confuse the classifier.

After feature selection, much of the confusion dies away. The performance of the GA/ANN rises to 100% on both datasets, but now using only 4 features rather than the nine required for the purely spectral dataset, which indicates that the statistical dataset does contain some useful information, and this supplants several of the features required for the spectral dataset. The constant width SVM with GA jumps to 100% on training data and 98.6% on test data , using only 4 features out of the 156 available. This is the highest performance it achieves over the three datasets from machine A. The averaged width SVM with GA performance on the training set is 98.2%, and achieves 99.2% on the test set. This uses only 6 features out of the possible 156. The specific width SVM has the highest performance of the SVMs, managing 99.1% on the training set, and 99.5% on the test set, using only 5 inputs. This compares very well with the

Classifier	No. Inputs	Training Success (%)	Test Success (%)
ANN	33	100	100
GA/ANN	4	100	100
SVM (const. width)	33	100	100
GA/SVM (const. width)	4	100	100
SVM (av. width)	33	100	100
GA/SVM (av. width)	4	100	100
SVM (sp. width)	156	100	100
GA/SVM(sp. width)	4	100	100

Table 6.11: Dual Class: Performance with machine B dataset (33 features).

ANN in terms of performance.

6.7.2 Dual Class Results: Machine B

Table 6.11 shows the results of the different classifiers on the dataset from machine B. As can be seen, all the classifiers successfully manage to achieve 100% accuracy on both the training and test sets. Feature selection only serves to reduce the number of features required for correct classification. As can be seen, using only 4 features from the set of 33, all the different classifiers are able to classify between fault and no fault conditions correctly.

6.7.3 Multiclass Results: Machine A

Table 6.12 shows the results of feature selection with the SVMs. For comparison, the performance of the ANN after feature selection is shown (which has already been covered earlier in this chapter). As can be seen, the performance with the dataset from machine A is high with all classifiers after feature selection. Perhaps the biggest improvement in performance comes from the constant width SVM, which jumps from a level of 84.2% on the unseen test set, using 156 inputs, to 98.2% after feature selection, using only 9 inputs. While the constant width

Classifier	No. Inputs	Training Success (%)	Test Success (%)
ANN	156	99.8	97.0
GA/ANN	6	100	100
SVM (const. width)	156	100	84.2
GA/SVM (const. width)	9	100	98.2
SVM (av. width)	156	100	97.9
GA/SVM (av. width)	4	100	99.9
SVM (sp. width)	156	100	99.4
GA/SVM(sp. width)	4	100	100

Table 6.12: Multiclass: Performance with combined dataset (156 features).

SVM has the largest jump in performance, it is not the best performer of the four classifiers after feature selection. The averaged width SVM manages 100% and 99.9% on the training and test sets using only 4 inputs, while both the specific width SVM and the ANN manage 100% on both the training and test sets, the SVM using 4 inputs, while the ANN uses 6 inputs. These results show that using feature selection, the SVM can become as robust and accurate as an ANN for an application of this nature.

6.7.4 Multiclass Results: Machine B

Table 6.13 shows the results of experiments conducted using the spectral dataset from machine B. Feature selection has improved the performance of each of the classifiers over their performance with the full datasets. However, no classifier manages to achieve 100% accuracy on the unseen test set. The best performance on the test set is 96.4%, achieved by the ANN, which uses 10 inputs of a possible 33; this classifier only achieves 92.9% on the training set however. All others manage 100% on the training set, whether they have feature selection or not. The best performance of an SVM is achieved by the specific width SVM after feature selection, which reaches 94.6% on the test set, using 12 inputs. This

Classifier	No. Inputs	Training Success (%)	Test Success (%)
ANN	33	100	76.8
GA/ANN	10	92.9	96.4
SVM (const. width)	33	100	85.7
GA/SVM (const. width)	12	100	92.9
SVM (av. width)	33	100	83.9
GA/SVM (av. width)	8	100	92.9
SVM (sp. width)	33	100	87.5
GA/SVM(sp. width)	12	100	94.6

Table 6.13: Multiclass: Performance with machine B dataset (33 features).

is comparable to the best result managed by a standalone ANN using the full dataset, where the PNN also managed to reach 100% on the training set and 94.6% on the test set. The performances of the constant and averaged width SVMs were identical, managing a performance of (100% on the training set and) 92.9% on the test set, an increase of 7.2% in the case of the constant width, and 9.0% in the case of the averaged width SVM.

6.7.5 Support Vectors: Machine A

Table 6.14 shows the breakdown of the number of support vectors for the three different SVMs, when classifying the combined statistical and spectral dataset in the dual class case. In all three cases, after using feature selection, the number of support vectors required to define the classification boundary is much less than that required by the standalone classifiers without the benefit of feature selection. The greatest reduction in the number of support vectors occurs with the specific width SVM, which sees a reduction of 40% on the number of support vectors required to build the classifier from 709 to 420 vectors. This decrease in the number of vectors is accompanied by an increase in performance on the unseen test set of 12.6%, and a reduction in the number of inputs from 156 to 5. The

reduced dimensionality of the dataset means that the GA is able to select features that create a much simpler boundary in feature space, meaning that the SVM requires less vectors to define the boundary, and the robustness of the classifier improves as a result.

It is interesting to note the number of support vectors used to define the boundaries in the two class problem. Using the constant width SVM after feature selection, the SVM requires 431 vectors to create the correct partition for the fault data; this is a slight improvement over the 467 required when the full dataset is used. The number of vectors used for the fault and no fault classes is identical for the constant width SVM. However, for the averaged width SVM, the SVM without feature selection uses only 116 vectors to define the fault boundary, compared with 109 after feature selection has take place. This no fault category uses 532 vectors for the no fault class before feature selection, but only 395 after selection. Before feature selection, the specific width SVM uses a similar number of support vectors to the averaged width SVM. However, after feature selection, the number of vectors required to define both boundaries falls dramatically, with only 42 vectors being required for the fault class, compared to the 114 used before feature selection. The no fault class also sees a significant reduction in the number of vectors required, falling from 595 to 378 vectors.

Table 6.15 shows the breakdown of the support vectors in the multiclass classifiers, using the combined statistical and spectral datasets. In general terms, the behaviour of the multiclass SVMs after feature selection mirrors that of the dual class classifiers after feature selection. In all cases, the number of support vectors required to create the classifier has decreased. The greatest decrease comes from the specific width SVM, which uses less than half of the support vectors originally required after feature selection has been carried out, using only 1542 vectors rather than the 3185 required when the full dataset was used. The performance

	No. Support Vectors			σ	Scaling
	No Fault	Fault	Total	Value	Factor
SVM (const. width)	467	467	**934**	1.1	-
GA/SVM (const. width)	431	431	**862**	0.2	-
SVM (av. width)	532	116	**648**	-	9
GA/SVM (av. width)	395	109	**504**	-	3
SVM (sp. width)	595	114	**709**	-	9
GA/SVM (sp. width)	378	42	**420**	-	3

Table 6.14: Dual Class Machine A: Number of support vectors used by each SVM after feature selection in classifying the combined statistical and spectral data, with σ or scaling factor values, as relevant.

has also improved marginally (see table 6.12). It is interesting to note that the GA has not chosen the maximum scaling factor permitted, which was where the best performance was reached when feature selection was not used. It is also interesting to note that the width parameter of the constant width SVM is small in both cases, meaning that the classifier is acting more like a nearest neighbour classifier rather than enclosing the data completely.

Examining the number of support vectors used by the different classes, it is interesting to note that the inner race (IR) and rolling element (RE) faults both require significantly less support vectors to define their boundaries than any of the other classes when the two modified kernels are used. The IR fault uses 71 vectors for the averaged SVM after feature selection, while the specific width SVM uses 63, against the 408 used by the constant width SVM after feature selection. The RE fault uses 408 vectors for the constant width SVM, 32 for the averaged width SVM, and only 23 for the specific width SVM, however still manages to

achieve 100% classification accuracy. This implies that the boundary for these two classes is easy to define, and also distinct from the other classes spatially within feature space. The specific width kernel seems to require less vectors to define the boundary after feature selection, and this is caused by the hyperellipsoidal enclosure, which will be enclosing more of the training data points with one vector than a support vector using a spheroid enclosure from the averaged dataset. By removing the garbage features, this works much more effectively than when using the full dataset, as the removal of the high variance features mean that the points can be enclosed much more simply, requiring less support vectors.

Generally speaking, the lower dimensionality of feature space after feature selection means that the boundaries are much simpler to construct than in the high dimensional feature space. Where a large number of support vectors are still being used even after feature selection implies that the boundary is complex, and in order to allow the boundary to be defined accurately, many vectors must be used. Figure 6.1 shows a representation of how a relatively complex boundary might appear. An easier boundary, such as that used on the IR and RE faults would be a much smoother curve, meaning that fewer support vectors are required to define it.

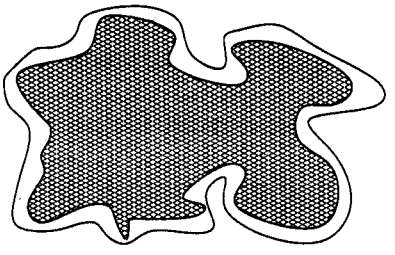


Figure 6.1: A complex boundary requiring many support vectors

	No. Support Vectors							σ	Scaling
	NO	NW	IR	OR	RE	CA	Total	Value	Factor
SVM (const. width)	500	458	442	442	442	485	**2769**	1.0	-
GA/SVM (const. width)	396	375	408	408	408	399	**2394**	0.2	-
SVM (av. width)	602	565	102	425	134	675	**2503**	-	8
GA/SVM (av. width)	389	347	71	432	32	451	**1722**	-	2
SVM (sp. width)	784	634	177	582	247	761	**3185**	-	9
GA/SVM (sp. width)	422	295	63	391	23	348	**1542**	-	5

Table 6.15: Multiclass Machine A: Number of support vectors used by each SVM after feature selection in classifying the combined statistical and spectral data, with σ or scaling factor values, as relevant.

6.8 Discussion

Feature selection would appear to offer a great many improvements over manual methods of selecting features for classifiers in this type of situation. In every example, using feature selection with a classifier has produced a result that is at least as good as or better than what the standalone classifier is able to achieve. Additionally, the robustness of the classifier also tends to improve, through the removal of "garbage" features which are confusing the classifier by retaining misleading information and reducing the effectiveness of the classifier. Reducing the number of features required for classification also lessens the computational needs of the process, making the classifier simpler and cheaper to implement, as any hardware requirements for an embedded system will be reduced as a result.

The SVM classifier improves greatly in terms of reliability, generalisation, and robustness after feature selection. This is due in the main to the removal of features that have a large variance within a class. By selecting features which

tend to have little variance within a class, the GA is able to create SVMs which define a boundary around data which will stay comparatively stable when faced with new data. The presence of data which contains a lot of outliers in the training set will cause the SVM training algorithm to orientate the hyperplane incorrectly with regard to real data. By selecting those features which vary little in a class, but are separable from different classes, the GA lessens the risk of this happening. It is possible to use a slightly different form of SVM, which can take into account nonseparable data through the use of "slack variables" [1, 85], and this can define the boundary correctly. This approach might improve the generalisation performance significantly; unfortunately time did not allow this to be checked.

By contrast, the ANN tends to be fairly robust whether feature selection is used or not; feature selection tends to improve the accuracy of classification, but the generalisation properties of the network tend to remain fairly constant before and after feature selection, with respect to the fact that the network tends to perform equally well on the training and test sets.

Using feature selection allows the dual class SVM to improve, recording very high performances on the data used here, and a significant improvement over the performance on the standalone datasets. In the multiclass case, the performance also improves to the point where the SVM is capable of matching or exceeding the performance of the MLP on the same data.

Perhaps the largest drawback to using the feature selection system is in the time it takes to perform training of the classifiers as part of the feature selection process. For the multiclass case, using the GA in conjunction with the SVM, the run time of the experiments was approximately 14 days, which is a long time to wait for results. By contrast, the slowest GA/ANN combination took about 9 days. However, the performance gains made through employing feature selection

mean that the computational costs are justified, or would be in a production, rather than research environment.

6.9 Summary

This chapter has introduced the use of a specific feature selection strategy for classifiers in the MCM context. It has been shown to be a successful approach, providing significant improvements in the performance of all classifiers used, while also managing to reduce the number of inputs required to achieve good classification performance by up to a factor of 39.

Using GA based feature selection, it has been shown that feature selection is a valid and useful technique in condition monitoring, and can provide extremely high levels of performance using very few inputs, reducing the computational workload required to preprocess data, while maintaining or improving performance.

It has been shown that both the SVM and the MLP are suitable for reliable classification tasks after feature selection has been undertaken, and that the performance of the two classifiers is comparable after feature selection has taken place.

Chapter 7

Conclusions

This thesis has examined the application of a number of different classification techniques in condition monitoring, and evaluated the usefulness of the techniques on the basis of the data available. The use of artificially intelligent techniques in the field of MCM is well established, however there is still much scope for the evaluation of new techniques. The goalposts are moving constantly, and research offers many possibilities for the use of new algorithms that may offer improvements over current approaches. Small improvements in reliability and efficiency are still attractive in MCM; a 1% improvement in the running efficiency of the large electric motors used in sewage pumping (approx. 4.5MW) can recoup the cost of manufacturing the motor in the space of a month [103].

ANN are in general robust classifiers; of the three ANN classifiers tested, all performed reasonably well, however, the MLP was found to be the most robus, with the best combination of desirable characteristics in terms of performance, generalisation, and compact size. The PNN is inferior to both the RBF and MLP for all cases except that of the data from machine B, where there are only 56 training patterns, and as a result there is little data for the training algorithms for the MLP and RBF to work with. The Parzen approximation is

probably a better solution for problems of this nature, as the sparse training data will improve the performance of the Parzen approximation in areas where there is overlap, and the PNN will act more like a nearest neighbour classifier. Whether this means that PNN is ideal for problems of this nature requires further work and investigation. The RBF offers fairly good performance, however there are also disadvantages, as the classifier seems slightly less robust than the MLP, and can be erratic. Additionally, the problem of finding a suitable value of kernel parameter can make a huge difference to the performance of the network. Overall, the MLP seems to be the most robust and compact ANN for use, which makes it ideal for use in MCM applications. There are also issues with the MLP, but they are mainly concerned with the choice of the number of neurons in the hidden layer of the network.

The SVM shows good performance with the modified kernels; however at the same time, the SVM seems to be much more sensitive to un-correlated data, and as a result can be less robust than the ANNs, with poor performance. A process of feature selection would solve this problem, and increase the performance of the classifier. The results from chapter 6 back this up. There are additional issues with the SVM, however much of this is concerned with the assumption that the training data given is statistically representative of the likely distribution of data within feature space; this is not always the case with MCM data. The modified kernels proposed seem to reduce the impact of this problem on the performance of the SVM, particularly in cases where there is overlap between different classes of data. Again, feature selection is capable of reducing much of the problems associated with this. Of the two modified kernels, both achieve similar performance results, although there is a tradeoff; the averaged kernel offers good performance, but is less computationally complex than the specific width kernel, which gives a higher performance at the cost of increased computation.

Performance of the SVM is good, however it is not quite as high as the MLP unless feature selection is used.

The experiments carried out with feature selection have shown the usefulness of the technique as an approach for optimising classification performance in the MCM area. The automated nature of the selection is also an advantage, as it allows the algorithm to find correlations between features itself, without preconceived ideas of the "best" choice of inputs for the classifier. The advantages of feature selection in this context are threefold; firstly, reducing the number of inputs required for the classifier reduces the amount of pre-processing that has to take place prior to classification taking place, reducing the computing burden; secondly, reducing the number of inputs to the classifier simplifies the classifier, and also tends to improve the performance of the classifier by removing redundant features which do not contribute to the overall solution. Thirdly, the reduced feature set selected will tend to be more robust, due to the smaller variance of the features chosen on a per class basis (i.e. data for each class is tightly localised in feature space, and does not tend to be subject to outliers). This makes the combination of classifier and feature selection attractive to many different areas, as the compact nature of the resultant classifier, combined with the high performance achievable makes it ideal for embedded applications where the computing resources available are small and computationally simple applications are needed, due to cost considerations. Using the GA for feature selection offers further advantages with the classifiers, as it is possible to allow the parameters of the classifier to be selected by the GA, allowing a (near) optimal solution to be found in one step. This allows the selection process to examine as many different features as are available, and select the best set of features required, without requiring any further supervision from an operator.

Using feature selection with the SVMs creates highly robust and accurate

classifiers which are able to perform comparably with that of the best ANNs even after feature selection. Removing the erratic features from the dataset greatly improved the performance of the SVM.

There is scope for further work on the basis of the work carried out thus far. The effectiveness of the two modified SVM kernels proposed needs to be evaluated on a broader range of non MCM problems. Examination of the classifier performance on some standard benchmarking problems would be a worthwhile exercise. An evaluation of the performance of the PNN after feature selection would also be worthwhile, as removing the features which are inseparable among the classes may improve the performance of the PNN significantly; unfortunately, time did not allow this work to be carried out. The performance of the different ANNs with other small size training sets (50-100 members approx.) could be investigated further, to determine whether the performance of the PNN with the multiclass data from machine B is unique to that dataset, or a general characteristic of the classifier.

With regard to feature selection, work remains to be done with other feature selection algorithms, to evaluate whether other alternative approaches may offer advantages not offered by the GA; additionally, it would also be interesting to try and apply the combined pre-processing/feature selection/classifier approach to problems from other areas than MCM, as there are many similarities with speech recognition and medical monitoring applications. The feature selection approach is of use in any area where classification has to be performed, and the features best suited to the problem in question are unknown.

The work carried out so far has been based around experiments performed on recorded data; it would be of interest to determine whether the "optimal" set of inputs chosen by the GA in the laboratory are capable of dealing with data in a real working environment, with all the different external noise sources

that are present, and other faults. Talking to industrial contacts has made it plain that one of the biggest problems remains in the detection of faults when there are no examples available of what a fault looks like. Work is required to develop techniques which are capable of detecting deviation from normal on a reliable basis, and this is an exceedingly ill defined problem within a classification context.

Bibliography

[1] V. N. Vapnik. *Statistical Learning Theory.* John Wiley & Sons, Inc., 1998.

[2] R. L. Kincaid. Evolution of condition monitoring and the management of maintenance. In Jones M. H., editor, *Condition Monitoring 87: Proceedings of an International Conference on Condition Monitoring held at university College of Swansea, 31st March - 3rd April 1987*, pp. 13–21. Pineridge Press, Swansea, UK, 1987.

[3] A. C. McCormick and A. K. Nandi. Real time classification of rotating shaft loading conditions using artificial neural networks. *IEEE Transactions on Neural Networks*, 8(3):748–757, 1997.

[4] D. C. Baillie and J. Mathew. A comparison of autoregressive modelling techniques for fault diagnosis of rolling element bearings. *Mechanical Systems and Signal Processing*, 10(1):1–17, 1996.

[5] T. Sorsa, H. Koivo, and H. Koivisto. Neural networks in process fault diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(4):815–825, 1991.

[6] P. Polycarpou and A. Helmicki. Automated fault detection and accommodation: A learning systems approach. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(11):1447–1458, 1995.

[7] T. I. Liu and J. M. Mengel. Intelligent monitoring of ball bearing conditions. *Mechanical Systems and Signal Processing*, 6(5):419–431, 1992.

[8] R. G. Silva, R. L. Reuben, K. J. Baker, and S. J. Wilcox. Tool wear monitoring of turning operations by neural network and expert system classification of a feature set generated from multiple sensors. *Mechanical Systems and Signal Processing*, 12(2):319–332, 1998.

[9] A. Starr, M. Desforges, and J. Esteban. Strategies in data fusion - a condition monitoring approach. In *Proceedings of COMADEM '99, University of Sunderland*, pp. 399–408, Kingham, Oxford, 1999. Coxmoor Publishing.

[10] C. Kirkham, A. Long, O. Taylor, and C. Isbell. Adaptive online systems for condition monitoring: The NEURAL-MAINE project. In *Proceedings of COMADEM '99*, pp. 317–325, Kingham, Oxford, 1999. Coxmoor Publishing Ltd.

[11] Z. Yangping, Z. Bingquan, and W. DongXin. Application of genetic algorithms to fault diagnosis in nuclear power plants. *Reliability Engineering and Safety Systems*, 67:153–160, 2000.

[12] L. B. Jack and A. K. Nandi. Genetic algorithms for feature selection in machine condition monitoring with vibration signals. *IEE Proceedings - Vision, Image and Signal Processing*, 147(3):205–212, 2000.

[13] B. D. Gemmel, J. R. McDonald, R. W. Stewart, R. N. T. Brooke, and B. J. Weir. A consultative expert system for fault diagnosis of turbine-generator plant. *Proceedings of the Institution of Mechanical Engineers: Part A*, 208:257–266, 1994.

[14] L. F. Pau. Survey of expert systems for fault detection, test generation and maintenance. *Expert Systems*, 3(2):100–111, 1986.

[15] C. Angeli and A. Chatzinikolaou. An expert system approach to fault diagnosis in hydraulic systems. *Expert Systems*, 12(4):323–330, 1995.

[16] T. C. Walker and R. K. Miller. *Expert Systems 1986.* SEAI Technical Publications, Madison, GA, 1986.

[17] A. Papoulis. *Probability, Random Variables and Stochastic Processes.* McGraw-Hill, 3rd edition, 1991.

[18] C. W. Therrien. *Discrete Random Signals and Statistical Signal Processing.* Prentice Hall, New Jersey, 1992.

[19] Cooley J. W. and J. W. Tukey. An algorithm for the machine calculation of complex fourier series. *Math. Computation*, 19:297–301, 1965.

[20] B. P. Bogert, M. J. Healey, and J. W. Tukey. *Time Series Analysis*, chapter 15, pp. 209–243. Wiley, New York, 1963.

[21] D. G. Childers, D. P. Skinner, and R. C Kemerait. The cepstrum: A guide to processing. *Proceedings of the IEEE*, 65(10):1428–1443, 1977.

[22] J. C. Hassab and R. Boucher. Analysis of signal extraction, echo detection and removal by complex cepstrum in presence of distortion and noise. *Journal of Sound and Vibration*, 40(3):321–335, 1975.

[23] J. W. A. Fackrell, P. R. White, J. K. Hammond, R. J. Pinnington, and A. T. Parsons. The interpretation of the bispectra of vibration signals-I. theory. *Mechanical Systems and Signal Processing*, 9(3):257–266, 1995.

[24] W. B. Collis, P. R. White, and J. K. Hammond. Higher-order spectra: The bispectrum and trispectrum. *Mechanical Systems and Signal Processing*, 12(3):375–394, 1998.

[25] C. L. Nikias and J. M. Mendel. Signal processing with higher order spectra. *IEEE Signal Processing Magazine*, 10(3):10–37, 1993.

[26] J. W. A. Fackrell, P. R. White, J. K. Hammond, R. J. Pinnington, and A. T. Parsons. The interpretation of the bispectra of vibration signals-II. experimental results. *Mechanical Systems and Signal Processing*, 9(3):267–274, 1995.

[27] C. K. Mechefske and J. Mathew. Fault detection and diagnosis in low speed rolling element bearings part i: the use of parametric spectra. *Mechanical Systems and Signal Processing*, 6(4):297–307, 1992.

[28] A. K. Nandi and K. Tutschku. Machine condition monitoring based on higher order spectra and statistics. In *Proceedings of ATHOS Workshop on Higher Order Statistics in Signal Processing*, University of Edinburgh, September 1994.

[29] A. C. McCormick, A. K. Nandi, and L. B. Jack. Digital signal processing algorithms in condition monitoring. *International Journal of COMADEM*, 1(3):5–14, 1998.

[30] W. Q. Jeffries, J. A. Chambers, and D. G. Ingfield. Experience with bicoherence of electrical power for condition monitoring of wind turbine blades. *IEE Proceedings on Vision, Image and Signal Processing*, 145(3):141–148, 1998.

[31] E. P. Wigner. On the quantum correction for thermodynamic equilibrium. *The Physical Review*, 40:749–759, 1932.

[32] J. Ville. Théorie et applications de la notion de signal analytique. *Cables et Transmission*, 2A:61–74, 1948.

[33] L. Cohen. Time-frequency distributions - a review. *Proceedings of the IEEE*, 77:941–981, 1989.

[34] H. Choi and W. J. Williams. Improved time-frequency representation of multicomponent signals using exponential kernels. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-37:862–871, 1989.

[35] J. C. Moss and J. K. Hammond. A comparison between the modified spectrogram and the pseudo-wigner-ville distribution with and without modification. *Mechanical Systems and Signal Processing*, 5(3):233–255, 1994.

[36] D. König and J. Böhme. Application of cyclostationary and time-frequency signal analysis to car engine diagnosis. In *Proceedings of ICASSP*, volume 4, pp. 149–152. IEEE, 1994.

[37] H. Oehlmann, D. Brie, M. Tomczak, and A. Richard. A method for analysing gearbox faults using time frequency representations. *Mechanical Systems and Signal Processing*, 11(4):529–545, 1997.

[38] B. D. Forrester. Use of the wigner ville distribution in detection of helicopter transmission fault detection. In *Proceedings of the Australian Symposium on Signal Processing and Applications - ASSPA89*, pp. 78–87, 1989.

[39] W. J. Wang and P. D. McFadden. Early detection of gear failure by vibration analysis - I: Calculation of the time-frequency distribution. *Mechanical Systems and Signal Processing*, 7(3):193–204, 1993.

[40] W. J. Staszewski, K. Worden, and G. R. Tomlinson. Time-frequency analysis in gearbox fault detection using the wigner-ville distribution and pattern recognition. *Mechanical Systems and Signal Processing*, 11(5):673–792, 1997.

[41] D. E. Newland. *An Introduction to Random Vibrations, Spectral and Wavelet Analysis.* Longman, 3rd edition, 1993.

[42] S. Mallat. *A Wavelet Tour of Signal Processing.* Academic Press, San Diego, Ca, 1998.

[43] S. T. Lin and P. D. McFadden. Gear vibration analysis by b-spline wavelet-based linear wavelet transform. *Mechanical Systems and Signal Processing*, 11(4):603–609, 1997.

[44] W. J. Staszewski and G. R. Thomlinson. Application of the wavelet transform to fault detection in a spur gear. *Mechanical Systems and Signal Processing*, 8(3):319–356, 1994.

[45] W. J. Wang and P. D. McFadden. Application of orthogonal wavelets to early gear damage detection. *Mechanical Systems and Signal Processing*, 9(5):497–507, 1995.

[46] B. A. Paya, I. I. Esat, and M. N. M. Badi. Artificial neural network based fault diagnostics of rotating machinery using wavelet transforms as a preprocessor. *Mechanical Systems and Signal Processing*, 11(5):751–765, 1997.

[47] W. Wu and R. Du. Feature extraction and assessment using wavelet packets for monitoring of machining processes. *Mechanical Systems and Signal Processing*, 10(1):29–53, 1996.

[48] A. Murray and J. Penman. Wavelets as an alternative to the fft in condition monitoring schemes using anns. In R. B. K. N. Rao, R. A. Smith, and J. L. Wearing, editors, *Proceedings of COMADEM '96*, pp. 177–186, University of Sheffield, July 1996. Sheffield Academic Press.

[49] G. Dalpiaz and A. Rivola. Condition monitoring in automatic machines: Comparison of vibration analysis techniques. *Mechanical Systems and Signal Processing*, 11(1):53–73, 1997.

[50] B. Liu and S.-F. Ling. On the selection of informative wavelets for machinery diagnosis. *Mechanical Systems and Signal Processing*, 13(1):145–162, 1999.

[51] J. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[52] K. DeJong. *An Analysis of the Behaviour of of a Class of Genetic Adaptive Systems.* PhD thesis, Dept. of Computer and Communication Sciences, University of Michigan, Ann Arbor, 1975.

[53] K. S. Tang, K. F. Man, S. Kwong, and Q. He. Genetic algorithms and their applications. *IEEE Signal Processing Magazine*, 13(6):22–37, 1996.

[54] D. S. Whitely. A genetic algorithms tutorial. *Statistics and Computing*, 4:65–85, 1994.

[55] D. E. Goldberg. *Genetic Algorithms in Search, Optimisation and Machine Learning.* Addison Wesley, 1989.

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

[57] T. Bäck. *Evolutionary Algorithms in Theory and in Practice.* Oxford University Press Inc., New York, 1995.

[58] K. F. Man, K. S. Tang, and S. Kwong. *Genetic Algorithms.* Springer Verlag, 1999.

[59] M. Wall. Galib: A C++ genetic algorithms library. available from http://lancet.mit.edu/ga .

[60] G. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In W. W. Cohen and H. Hirsh, editors, *Machine Learning: Proceedings of the Eleventh International Conference*, pp. 121–129, San Francisco, CA, 1994. Morgan Kaufmann.

[61] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis*, 1(3):131–156, 1997.

[62] P. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, C-26(9):917–922, 1977.

[63] A. Webb. *Statistical Pattern Recognition.* Arnold, New York, 1999.

[64] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.

[65] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Macmillan, 1994.

[66] P. D. McFadden and J. D. Smith. Vibration monitoring of rolling element bearings by the high frequency resonance technique - a review. *Tribology International*, 17(1):3–10, 1984.

[67] R. Rojas. *Neural Networks - A Systematic Introduction.* Springer-Verlag, 1996.

[68] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, chapter 8. MIT Press, Cambridge, MA, 1986.

[69] P. J. Werbos. *Beyond Regression: New tools for prediction and analysis in the behavioural sciences.* PhD thesis, Harvard University, Cambridge, MA, 1974.

[70] S. Roy and J. J. Shynk. Analysis of the momentum LMS algorithm. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38:2088–2098, 1990.

[71] R. A. Jacobs. Increased rates of convergence through learning rate adaption. *Neural Networks*, 1:295–307, 1988.

[72] M. T. Hagan and M. Menhaj. Training feedforward networks with the marquhardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993, 1998.

[73] H. Demuth and M. Beale. *Neural Networks Toolbox: Users Guide.* The Mathworks Inc., Natick, MA, 3rd edition.

[74] D. S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[75] L. B. Jack, A. K. Nandi, and A. C. McCormick. Diagnosis of rolling element bearing faults using radial basis functions. *Applied Signal Processing*, 6(25):25–32, 1999.

[76] I. W. Mayes. Use of neural networks for on-line vibration monitoring. *Proceedings of the IMechE: Part A*, 208:267–274, 1994.

[77] P. D. Wasserman. *Advanced Methods in Neural Computing*, chapter 3, pp. 35–55. Van Nostrand Rheinhold, New York, 1993.

[78] D. F. Specht. Probabilistic neural networks. *Neural Networks*, 3:109–118, 1990.

[79] L. A. Branagan and P. D. Wasserman. Introductory use of probabilistic neural networks for spike detection from an on-line vibration diagnostic system. In *Proceedings of the 1992 Artificial Neural Networks in Engineering, ANNIE'92*, volume 2, pp. 719–724, Fairfield, NJ, USA, 1992. ASME.

[80] Y. Kawabe, K. Maegawa, T. Toyota, and P. Chen. Diagnosis method of centrifugal pumps by rough sets and partially-linearized neural network. In *Proceedings of the 1997 IEEE International Conference on Intelligent Processing Systems, ICIPS. Part 2 (of 2)*, pp. 1490–1494, Piscataway, NJ, USA, 1997. IEEE.

[81] F. Mo and W. Kinsner. Probabilistic neural networks for power line fault classification. In *Proceedings of the 1998 11th Canadian Conference on Electrical and Computer Engineering, CCECE*, pp. 585–588, Piscataway, NJ, 1998. IEEE.

[82] V. N. Vapnik. On the uniform convergence of relative frequencies of events to their probabilities. *Soviet Mathematics: Doklady*, 9:915–918, 1968.

[83] J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of ESANN '99*, pp. 219–224, Bruges, Belgium, April 1999.

[84] E. Bredensteiner and K. P. Bennet. Multicategory classification by support vector machines. *Computational Optimisations and Applications*, 12(1):53–79, 1999.

[85] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):955–974, 1998.

[86] B. Schölkopf. SVMs - a practical consequence of learning theory. *IEEE Intelligent Systems*, 13(4):18–19, 1998.

[87] V. Vapnik. *Estimation of Dependencies Based on Empirical Data.* Springer-Verlag, New York, 1982.

[88] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *Proceedings of the IEEE Neural Networks for Signal Processing VII Workshop*, pp. 276–285, Piscataway, NJ, USA, 1997. IEEE Press.

[89] J. C. Platt. Fast training of support vector machines using sequential minimal optimisation. In Scholkopf B., Bürges C., and Smola A., editors, *Advances in Kernel Methods - Support Vector Learning.* MIT Press, 1998.

[90] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthi. Improvements to platt's SMO algorithm for SVM classifier design. Technical report, Control Division, Dept. of Mechanical and production Engineering, National University of Singapore, 1999. Technical Report CD-99-14.

[91] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theoretical Probability and it's Applications*, 17:264–280, 1971.

[92] S. Haykin. *Neural Networks: A Comprehensive Foundation*, chapter 6. Macmillan, 1994.

[93] C. Campbell and N. Christianini. Simple learning algorithms for training support vector machines. Technical report, University of Bristol, 1998.

[94] C. Campbell. Algorithmic approaches to training support vector machines: A survey. In *Proceedings of ESANN 2000*, Bruges, 2000.

[95] N. Cristianini, C. Campbell, and J. Shawe-Taylor. Dynamically adapting kernels in support vector machines. In Solla S. Kearns M. and Coln D., editors, *Advances in Neural Information Processing Systems*, volume 11, pp. 204–211. MIT Press, 1999.

[96] O. Chappelle and V. Vapnik. Model selection for support vector machines. In Leen T. K. Solla S. A. and Muller K. R., editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 2000.

[97] B. Schölkopf, J. Shawe-Taylor, A. Smola, and R. Williamson. Generalisation bounds via eigenvalues of the gram matrix. Technical Report NC2-TR2-1999-035, Espirit Working Group in Neural and Computational Learning II(Neurocolt2), http://www.neurocolt.com, March 1999.

[98] S. Amari and S. Wu. Improving support vector machine classifiers by modified kernel functions. *Neural Networks*, 12(6):783–789, 1999.

[99] M. Kudo and J. Sklansky. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, 33:25–41, 2000.

[100] H. Vafaie and K. de Jong. Genetic algorithms as a tool for feature selection in machine learning. In *Proceeding of the 4th International Conference on Tools with Artificial Intelligence, Arlington, VA, USA.*, pp. 200–204, 1992.

[101] J. Yang and V. Honavar. Feature subset selection using A genetic algorithm. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pp. 1–24, Stanford University, 13-16 1997. Morgan Kaufmann.

[102] O. Mangasarian and D. R. Musicant. Successive overrelaxation for support vector machines. *IEEE Transactions on Neural Networks*, 10(5):1032–1037, 1999.

[103] R. Patrick. Weir Pumps Ltd., private communication, July 1992.