The application of neural computation methods to forecasting and monitoring in an airline context.

Simon Nicholas Cumming

Ph.D.

University of Edinburgh

1998

Declaration

I hereby declare that this thesis is my own composition and that the work therein reported was performed by me, except where otherwise acknowledged in references.

To Fiona and Catriona

Abstract

This thesis examines the applicability of artificial neural networks (neural computation methods) to tasks of forecasting and condition monitoring involving real-world data in the context of a large airline business. The first chapter introduces artificial neural networks (concentrating on multilayer perceptron, radial basis function and self-organising map networks), and provides some motivation for their use in problems of statistical estimation and monitoring. The second chapter gives a case study in the estimation of airline booking take-up from booking attributes held in a reservations system, comparing multilayer perceptrons and radial basis function networks, used in a classification regime, with the statistical method of Automatic Interaction Detection (AID) and lookup table and moving average methods. Some consideration is given to how the outputs of the neural network should be interpreted and to application-specific issues. The third chapter introduces the task of aircraft engine condition monitoring, gives a literature survey of the use of neural networks and related methods in condition monitoring and considers the applicability of various neural network approaches to the aircraft engine monitoring problem. The fourth chapter gives results from applying neural networks in a classification and regression mode to operational aircraft engine condition monitoring data. In the fifth chapter a method of context-based novelty detection using a combination of self-organising maps is developed and illustrated on aircraft engine data. The sixth chapter gives concluding remarks on the use of artificial neural networks for data-driven forecasting and monitoring tasks using operational data.

Acknowledgments

Thanks are due firstly to my primary supervisor, Professor David Willshaw, for combining the benefit of extensive experience with a patient and encouraging approach. I would also like to thank Professor Keith Stenning of the Human Communications Research Centre at Edinburgh, and Alan Ainsworth, formerly of British Airways PLC, now at Imperial College Planning Applications Research Centre.

I should like to express my gratitude to Keith Rapley (now Innovation Manager) at British Airways for arranging sponsorship of this work, to Mark Raskino and Neil Morrison for early stimulation of ideas concerning neural networks; and to Dr Maurice Barr, Louis Busuttil, Paul Summerbell, Dr Himadri Chatterjee and Rupert Blackley for making possible the forecasting and attribute work. Thanks are due to Garry Copeland, Steve Mott, David Barnes and Rick Magaldi for open and helpful support on the engine condition monitoring projects.

A valuable contribution has been made by Dr Peter Cowley, Rachel Pearse and Colin Sylvester of the Computational Intelligence Group at Rolls Royce Applied Science Laboratories, Derby, with whom I have discussed the development of mapping methods for condition monitoring.

I would also like to acknowledge helpful advice from Dr John Pilkington of Scientific Computers. Dr Mike Wynne-Jones, Dave Cressy, Professor Chris Bishop, Mike Brinn, Dr Tom Harris and other past and present members of the committee of NCAF (Neural Computing Applications Forum) have over recent years provided the opportunity for convivial and high quality technical discussion.

I should like to thank Dr Bruce Graham, Dr Martin Simmen, and Rosanna Maccagnano at the CNS research group at Edinburgh.

Finally my thanks go to my wife Fiona and daughter Catriona for their patience and understanding.

Contents

1 Introduction

1.1 Goals and motivation.

The aim of this work is to provide a contribution to answering the question of whether artificial neural networks form a useful method for statistical data analysis and forecasting when applied to real-world data in the case of airline marketing and engineering applications. This can be seen as testing a null hypothesis that neural network techniques are no better than conventional ones in real-life applications.

1.2 Neural networks

An artificial neural network model consists of a set of simple processing elements or nodes, usually arranged in layers, with connections between the layers, each connection having associated with it a numerical weight. At each node i, the weighted sum $\sum w_{ij} x_j$ is computed, of the weights w_{ij} on the connections feeding into that node from input nodes j, multiplied by the *input values* x_j. The value of this sum is then passed through an activation function which often takes the form of a "squashing" or sigmoidal function, usually $f(x) = \frac{1}{1+e^{-x}}$ (the logistic function) or $f(x) = \tanh(x)$. The network is usually set up as an input-output model with the independent variables presented as the inputs and the dependent variables corresponding to the outputs. A feedforward network consists of a directed acyclic graph, that is there are no connections from a given node to the same node or a node further back (closer to the input layer) in the network. The weights in a neural network act as parameters which are iteratively adjusted according to a learning rule or algorithm as data is presented to the network. In the case of supervised learning, target values for the training data are presented at the output layer during the training phase and the weights adjusted to minimise an error function which is usually contains a term in the square of the difference between the target output and the network's output. The use of the sigmoid function introduces a nonlinearity in the parameters which enables the representation of nonlinear relationships but requires the use of an iterative nonlinear optimisation algorithm to determine the parameters. A multilayer perceptron (MLP) is a

feedforward network with sigmoidal activation functions. Multilayer perceptrons have been shown to have interesting properties as universal approximators of a wide class of functions (Hornik, Stinchcombe & White, 1988, Hecht-Nielsen, 1990; Hertz, Krogh & Palmer, 1991). The error back-propagation procedure refers to the use of the chain rule of partial differentiation to enable the "passing-back" of a term in the derivative of the error from one layer to the previous layer (Werbos, 1974, Bryson & Ho, 1969, Rumelhart, Hinton and Williams, 1986). Radial basis function networks use radially symmetric functions rather than sigmoids as activation functions. Very often these are spherically symmetrical gaussians of the form $f(z) \propto \exp(-z^2)$ .The network consists of two layers of weighted connections. The positions in space of the hidden units are determined by a clustering algorithm, for example, the *k*-means algorithm. The determination of the second layer weights is a linear optimisation process, which will usually result in a single, 'global' minimum for a given distribution of hidden units. (Broomhead & Lowe, 1988, Moody and Darken, 1989).

The multilayer perceptron and radial basis functions referred to above are examples of supervised learning where a target value is provided for each record of training data. Unsupervised learning proceeds without the provision of labelling or target values. Generally unsupervised learning algorithms perform a process analogous to principal components decomposition or dimensionality reduction. They can also be viewed as a stage in a clustering algorithm. One of the best known is Kohonen's Self-Organising (Feature) Map (SOM, Kohonen, 1982, 1989, 1995). In the SOM the input layer is fully connected to a grid (often two-dimensional) of units. The weights into each unit can be regarded as a prototype or template for a cluster of data. The learning rule finds the unit whose weight vector is closest in terms of some distance measure (often Euclidean distance) to the input data vector. This is known as the winning unit. The learning rule then updates the weight vectors of the winning unit, and its neighbours within a certain radius, with an increment directed towards the input vector. Though the SOM does not correspond to the minimisation of an error function in the same way as most other applicable neural network algorithms, it has found many applications, particularly in 'mapping' from a high-dimensional space to a low-dimensional one, and hence for data visualisation. More recently the technique known as the generative topographic map (GTM) is showing promise as a dimension-reduction method which is also a density model.

The increased flexibility in terms of functions which can be modelled, obtained by using a non-linear technique such as a neural network, is achieved at the cost of a loss of simplicity and parsimony of modelling, and often comprehensibility of model structure.

The parameters in a neural network are set by a process of "training" on a sample of data, the training set. The fact that many parameters need to be set gives a danger of overfitting the training set unless careful consideration is given to ensuring that the network performs adequately on data other than the training data. Overfitting is the situation where apparently good performance can be achieved on the training set but this arises from fitting the model to "noise" or random variation in the training set so that the performance on the test set is poor. This can be compared to the effect of too high a model order when fitting data with a polynomial. Obtaining adequate generalisation performance in a neural network usually requires some form of cross-validation, regularisation, or both. Cross-validation can take the form of either a leave-one-out (or leave-k-out) scheme of the kind familiar in statistical analysis (where the network is trained on all except k of the data records and tested on the remaining k and the process iterated for different subsets of the data), or the use of a validation set of data on which to evaluate the error function which is to be used to decide when to stop training a network, so as to avoid overfitting. A further limitation of the departure from the linear constraint is that the results of a non-linear optimisation algorithm are local minima and without exhaustive search, no guarantee can be given of producing a global optimum. Regularisation is an attempt to favour smoother, lower-complexity solutions, which will usually achieve better generalisation performance. The usual method for regularisation is to add to the error function a term in the sum of the squares of the weights.

1.3 Use of neural networks

Artificial neural networks have attracted research interest because of their pattern recognition and learning ability, their universal approximation properties and their ability to perform nonlinear statistical estimation tasks of various kinds. Supervised feedforward neural networks can be regarded as an extension of various regression and classification

algorithms from the disciplines of mathematical statistics and statistical pattern recognition. There are also parallels in filter theory and control theory in engineering.

Artificial neural networks have found application in a wide variety of fields, from computer vision, handwriting recognition and speech processing to system identification and process control. Building on the links to statistical methods, neural networks are widely applied in commercial and financial areas for forecasting, clustering and risk estimation. Neural networks have also been applied to control and learning in robots and other autonomous systems. Much of the development work in the neural network field has concentrated on the development of algorithms, often using for model comparison well-known and fairly limited data sets from standard sources, where the data has been previously cleaned up and to some extent pre-processed. In such cases neural networks are often claimed to "outperform" conventional (usually linear) techniques. There have been many studies attempting to compare the performance of neural network techniques (particularly multilayer perceptrons and radial basis function networks) to statistical pattern recognition or estimation techniques. These studies have often reported what appear to be conflicting results and taken as a whole can be confusing. Though many have sought simple measures to apply to data sets which will indicate which algorithms, for example for regression or clustering, would give good performance, the fact remains that results will be application-dependent.

1.4 Applying neural networks to "real-world" problems

Application of neural networks as a tool for analysing real-world data in a commercial or industrial environment comprises a task which is substantially different from assessing the performance of an algorithm on controlled data. The term "real-world" data is here intended to describe the situation where the analyst using the algorithms, neural network or conventional, does not have control over the form or quality of the data available. Real-world data may contain missing data elements, data errors or mislabelling. It may be statistically ill-behaved, for example containing outliers or being heteroscedastic (having a varying variance). It is often difficult to obtain clarity as to the exact definition of a data quantity. In addition to vagaries of data collection and character, it is necessary to address issues due to the character of real-world problems and the way in which they are specified. Real-world problems may not be well-specified; target data may not be reliable; the solution

of the modelling task inherent in the data may not be the required business solution; it may not be known whether the task at hand is tractable or what a "reasonable" expected error rate would be. These matters are discussed further in Chapter Six after results of case studies have been given.

1.5 Airline application areas

In a large airline company, large-scale, real-world tasks of forecasting, classification, monitoring and diagnosis arise in both the commercial and engineering activities.

1.5.1 Forecasting and capacity management

Forecasting problems can be treated by one of two orthogonal approaches: firstly, **time-series analysis,** where prediction is achieved from the autocorrelations and patterns in the stream of past data alone, and secondly **causal modelling,** where one is searching for the right *independent* variables, and the right mathematical model (in a *regression* or *classification* setting) in order to explain as much as possible of the systematic variation in a target (*dependent*) variable.

Artificial neural networks, viewed as trainable function approximators for input-output mappings, can be applied either to the time-series aspect of forecasting, for example in forecasting passenger demand (Hutchison & Stephens (1987), Morrison (1990)), or to the causal aspect. The first part of this thesis explores the use of neural networks, couched in a classification setting, for estimating the take-up rate of passenger bookings (the opposite of this is referred to as the no-show rate).

Capacity Management, or *Yield Management,* refers to the optimisation of revenue by controlling the extent to which seats are made available in different classes and at different prices, at various times before departure. Although historically most of the development in this subject has been carried out by airlines, more recently it has attracted interest from other fields of business and industry which can be seen as operating or dealing in a resource which needs to be booked, for example, hotels and vehicle hire.

Optimisation of capacity availability and yield management depend particularly critically on well-characterised forecasts and knowledge of statistical risks. Often, and particularly in this

case, the data are structured by a number of dimensions such as route, flight number, date and selling class; the history of bookings is kept and control decisions with respect to the number of seats in various classes to be made available are made as the booking history develops, based on some forecast of its future progress and the risks associated with various courses of capacity management action. This problem thus involves forecasting a noisy time-series which exhibits seasonality at the annual level, weekly level and in some cases daily level, is nonstationary and multivariate, is subject to the effects of "events", and has the additional complications of constraining of demand (by capacity itself or by capacity-management controls), "no-shows" (that is the case where passengers fail to register for the flight booked), cancelled bookings, "go-shows" (passengers travelling without a booking) and many other error-conditions and data-quality problems. In addition the temporal nature of the series is complicated by the fact that there are two "time-scales" simultaneously to be monitored: the time-line of travel and that of bookings.

For the purpose of this thesis the emphasis of the forecasting part will be on predicting "no-show" rates from booking attributes. Two case studies are presented. In one, a number of neural network structures are considered and compared. In the other, backpropagation neural networks are compared against a statistical partitioning algorithm and moving average methods.

1.5.2 Engine Condition Monitoring

Monitoring can be considered as a modelling task which at a conceptual level overlaps significantly with aspects of forecasting. Monitoring involves characterising a process with a mathematical model and constantly measuring the process against that model, with the aim of providing an alert to any unusual or novel circumstances and secondarily a prediction of the possible outcome of these circumstances.

In the second part of this thesis, neural networks are applied to the task of condition monitoring of jet aircraft engines. A variety of techniques and approaches are examined and some points raised about the concept of generalisation in this kind of application, as well as implications for experimental design and methodology in view of validation and verification requirements. A monitoring method involving the use of multiple neural networks is proposed and case studies reported using operational data.

1.6 Outline of thesis

The purpose of the work described in this thesis was to test the hypothesis that neural network methods give a consistent advantage over linear methods in real-world applications. In general, one can only expect this hypothesis to be true in application areas where the data relationship or mapping which is being modelled is non-linear. To demonstrate an advantage over linear methods, the benefit conferred by more flexible modelling which is obtained by a neural network needs to be shown to exceed the disadvantages of using a neural network, *i.e*, the loss of some statistical tools, tests and instruments for ascertaining error performance and confidence limits, and limitations such as the danger of overfitting or the algorithm converging to a local minimum of the error function. A further issue is the limited extent to which it is possible to give an "explanation" of the neural network model in terms which are meaningful in the user's domain. The issue of explanation will not be explored in this thesis.

In Chapter Two, the use of neural networks will be compared to other methods on the problem of forecasting the number of "no-shows" for a flight, that is, the number of passengers who do not check in for the flight they have booked. In Chapter Three, the use of neural networks for engine condition monitoring will be introduced. Some examples of the application of neural networks to engine condition monitoring using both a regression and a classification setting are given in Chapter Four. In Chapter Five the use of neural networks for novelty detection within a context is explored. A summary of findings and observations about the real-world use of neural networks is given in Chapter Six.

2 Attribute-based forecasting and classification: A Case study using airline booking data

2.1 Introduction

In this chapter, neural network methods are compared with the CHAID algorithm and lookup table methods for forecasting the take-up of bookings for a selection of flights, given information from the passenger booking details for those flights. The approach taken is to treat the problem as a classification task at the individual booking level, in order to calculate the expected number of passengers, for each class, on each flight, given the forward bookings in a booking database.

Firstly the background to the application will be given, then the abstraction of the problem as a classification task will be stated and different potential solution methods explained. A case study will be presented, consisting of two comparison exercises. The first comparison establishes the ability of neural network methods to carry out no-show forecasting and compares different neural network methods and the second compares neural network methods with the statistically-based tree-structured approach of chi-squared automatic interaction detection (CHAID).

2.1.1 Airline background

Firstly some definitions will be given, together with a brief explanation of how bookings and various special cases are handled by the reservations and departure control (check-in) systems.

Bookings, no-shows and cancellations

A *flight operation* is a particular flight number operating on a specified day. A *no-show* is defined as the case where a passenger makes a booking for a particular flight operation and then does not check in for, or travel on, that flight operation. The *no-show rate* is the number of no-shows on a flight operation divided by the number of bookings. Each booking record (sometimes known as a *passenger name record or PNR*) in the reservations system may refer to one or more people booked to travel together on the same flight. The aim is to estimate for each PNR the probability of that booking being taken up, and then to aggregate this information together for the whole flight operation to obtain a figure for the estimated *passenger load* [1]arising from the bookings, "net" of no-shows, that is, having taken the no-show rate into account. A *cancellation* is distinct from a no-show because in the case of a cancellation the information that the booking has been cancelled is recorded in the reservation system and the passenger is not expected. In this work, no account needs to be taken of cancellations, because the booking attribute data is captured in its final form before departure, in other words, all cancellations have been accounted for already. In practice if booking attributes were to be used for forecasting final passenger loads a certain number of days before departure, it would be necessary in addition to run a cancellations model to provide an estimate of the cancellation rate between when the forecast was run and the day of departure.

Selling classes, upgrades and downgrades

In addition to the well-known division of the aircraft passenger cabin into First class, business class and economy class (which distinction is referred to by the variable "*cabin*"), there is a further designation by which bookings are broken down for airline internal purposes. This is known as the *selling class*. There are about 25 selling classes[2], and they

---

[1] This will *not* be the final figure for the number of passengers carried, because of some special cases, such as passengers travelling without prior reservations..

[2] The actual number can vary over time, and between different flights.

are used to distinguish, for example, between full-fare tickets and advance purchase or special offer prices. Some of the selling classes refer to special cases such as groups, frequent-flyer bonus redemptions *etc.* In some cases the cabin in which the passenger travels is different from that in the booking or the ticket. Such cases are *upgrades* and *downgrades* and usually occur when one or more of the cabins is full but there is still some space in one of the others. These effects are ignored in this study for the main part of the modelling but formed part of a post-processing stage in which the effect of various levels of bookings was investigated. Details of this post-processing stage are not included in this work.

Hierarchical nature of the data

For the purposes of capacity management, it is necessary to have forecasts of the final number of passengers in each selling class for each flight number for each day of operation. There will be some cancellation of errors in individual no-show forecasts when these are aggregated to selling-class and flight-operation level. Similarly there is some cancellation of errors at the selling class level provided that the overall cabin loads are not affected and thus the number of upgrades or downgrades is unaffected. However, errors in the predictions at selling class level result in less efficient revenue optimisation. This case illustrates the general observation that business data is often hierarchical and that real-life cost functions often have subtleties associated with the hierarchical breakdown of the data.

Special cases

Some of the selling classes, which refer to special cases, contain only a very low number of bookings. For very low sample sizes, any statistical technique will be subject to large percentage errors. Therefore in the following analysis, usually only the selling classes containing the larger numbers of bookings have been included. An exception is the case of class 'F', *i.e.* first class, which has been included in some of the studies despite the fact that the capacity and number of bookings are both small, because the forecasting of passenger numbers in this class is important to the airline due to the high revenue involved.

2.1.2 Attribute-based classification tasks

Consider a classification application where the output (or *dependent variable*) is an indicator of positive or negative outcome and the input data (*independent variables*) are attribute variables drawn from the records of a database. Often this data will concern human activity and will be subject to a high level of random variation. The aims in this type of application can be stated as: to classify accurately the individual records, with respect to the class labels given in the training set, and to estimate a set of posterior probabilities for the outputs which can be used in further statistical processing.

In discussion of techniques for addressing this type of application the output variable is often referred to as a *response variable*. For example this might be, for each individual record, an indicator of whether the outcome was "positive", indicated by a '1', or "negative", indicated by a zero. The terms "positive" and "negative", referring to the outcome, are defined according to the application in question. In the airline application described in this chapter, the response variable is whether a booked passenger travels on the flight booked. Variables other than the response variable are called *predictor variables*.

Dimensions and dimension values

In this discussion I shall in some instances refer to the raw input variables as *dimensions*[3], and shall refer to the discrete values or categories which these variables may take on, as *dimension values*. For example, the dimension "sales area" might have the dimension values {UK, Europe, Americas, Far East, etc.}.

---

[3] This terminology arises in the use of multidimensional tables or online analytic processing (OLAP) systems.

"Data Mining"

The use of computational techniques, either those arising from mathematical statistics or from the Artificial Intelligence and Machine Learning communities, on the type of data described, is often referred to under the heading "*data mining*". See, for example, Holsheimer & Siebes, (1993), Adriaans & Zantinge (1996). Interest in "data mining" has grown with the increase in the amount of information stored in databases, and the desire by commercial organisations to characterise customer activity and changing business patterns. Data mining encompasses a range of database query, statistical, machine learning and neural network techniques which enable the automatic extraction of "knowledge" from data. This "knowledge" usually consists of clusters, predictive relationships and high-order correlations in the data. Applications which fall into this category are mostly in the commercial fields of banking, insurance and retail and have to do with customer selection (for marketing activity), creditworthiness estimation or risk estimation for insurance purposes. In these application types the data are often stochastic and frequently partially missing.

2.1.3 Comparison of methods

Recently there have been several attempts to compare the performance of various neural network methods with machine learning or rule induction methods and statistical methods of addressing attribute-based classification and regression problems. Some examples are the STATLOG project (Michie, *et al.*, (1994)) and the recent work of Ripley (1993, 1996).

This chapter is concerned with evaluation of neural networks, CHAID and lookup table method on real-world data for the no-show forecasting problem. A description of the statistical technique of Automatic Interaction Detection (AID, Hawkins & Kass, 1982) is given in section 2.2.1. Neural network approaches using the multilayer perceptron trained using the back-propagation procedure, and using radial basis function networks, are then introduced in section 2.2.2. The lookup table method used is documented in section 2.2.3. Results for the case study in the area of airline no-show prediction are introduced and discussed in subsequent sections.

2.2 Outline of approaches

2.2.1 Automatic interaction detection (AID)

AID (Automatic Interaction Detection, Hawkins & Kass, 1982) is a method for determining which variables have the greatest effect on the response variable and for constructing a model by recursively partitioning the space defined by the predictor variables. This is equivalent to building a tree structure with each level of the tree corresponding to a split according to one of the predictor variables. The objective is to determine partitions according to values (or groups of values) in the predictor variables, such that the value of the response variable within each region thus specified is significantly different from that in other partitions of the space as measured by some specified statistical test. Different statistical tests can be substituted depending on the type of data being analysed. The AID method is designed principally for applications where the predictor variables are *categorical* and may only take on a few possible values. SI-CHAID uses the *chi-squared* test to measure the significance of splitting on a variable value. [4]

The way in which SI-CHAID is used in practice is as follows. First there is a model-building phase with a training set of data. The algorithm determines appropriate splits and groupings of dimension values, within which the variation of the response variable is not significant, but between which it is significant, at a pre-determined significance level and according to the χ^2 test. The average (arithmetic mean) of the response variable in each of the subspaces is then calculated and entered into a lookup table. For unseen data, output values for the appropriate combination of dimension (attribute) values are retrieved from the table. If the match on the attribute set is not exact then defaults are implemented by reading from one level higher in the tree.

---

[4] SI-CHAID is an implementation of the CHAID technique developed by the company "Statistical Innovation" and which runs on the SAS(tm) statistical and database processing software.

2.2.2 Neural networks

Neural network methods were introduced in section 1.2. In this section we concentrate on the use of MLP and RBF networks in classification mode. With binary target outputs, under certain conditions it is possible to regard the outputs of the trained network as posterior probability estimates. Bishop (1995) makes the observation:

> "When we use a neural network to solve a classification problem, there are two distinct ways in which we can view the objectives of network training. At the simpler level we can arrange for the network to represent a nonlinear discriminant function so that, when a new input vector is presented to the trained network, the outputs provide a classification directly. The second approach, which is more general and more powerful, is to use the network to model the posterior probabilities of class membership. ... These probabilities can then be used in a subsequent decision-making stage to arrive at a classification. By arranging for the network outputs to approximate posterior probabilities, we can exploit a number of results which are not available if the network is used simply as a non-linear discriminant."

Benefits arising from this interpretation, which Bishop goes on to list, include the following:

(a) Minimum error-rate decisions can be achieved by assigning input vectors to the class having the largest posterior probability.

(b) Some degree of validation of whether the network is modelling the posterior probabilities accurately can be obtained by examining the difference between the average of the network outputs and the known prior probabilities for the training set. (Richard & Lippmann, 1991) .

(c) Compensation for the case when the prior probabilities expected when the network is in use *differ* from those which applied to the training set, can be achieved by using Bayes theorem and dividing the network outputs by the prior probabilities corresponding to the training set, multiplying them by the new prior probabilities and then normalising the results.

(d) Loss matrices other than the 0 / 1 case can be accommodated in that the posterior probabilities can be combined with a suitable matrix of loss coefficients to allow the minimum risk decision to be made. (Note that an alternative way of achieving this is to modify the error function).

(e) Rejection of input vectors (a "don't know" answer) can be achieved in a principled way, by using a suitable threshold on the posterior probability values obtained from the network.

Bishop gives general conditions under which the network's outputs can legitimately be regarded as probabilities. He derives[5] the class of error functions for which the assumption of the equivalence of outputs and probabilities is valid. The class of functions which satisfy this condition includes the *sum-of-squares* error function and the *cross-entropy* error function. In practice there is a constraint that the network is able to converge to a suitably low value of the error function. This requires that it has a sufficient number of parameters, *i.e.* a sufficient number of hidden nodes and connections.

In this work we will follow the principles outlined by Bishop insofar as regarding the output values of the network as probability estimates from which a subsequent decision stage is driven. It is assumed in the following work that there has been no material change in the prior probability of a no-show between the time frame of the training set and that of the recall set. No attempt is made to define a rejection threshold or a "don't-know" class. No distinction is made here between the value of the loss function for a no-show predicted as a show and for a show predicted as a no-show.

2.2.3 Lookup Table

In this application, given the encoding of the input data specified above, and in the absence of any new combination of attributes not experienced in the training set, the best results might be expected to be obtained from the use of an exhaustive lookup table, where for each unique combination of attribute values in the training set, a value is kept for the mean no-show probability of passengers with those attributes, on the combination of flight number, class and date under consideration. A lookup table of this sort lacks the ability to generalise to new cases. However, if the attribute data are hierarchical, then default values can be computed at a higher level of aggregation and used in the case of data points which lie outside the training set.

---

[5] The analysis used there is strictly true only for the limit of an infinite training set, where a sum over training patterns can be replaced by the corresponding integral.

2.3 The no-show forecasting problem as an attribute-based estimation problem

2.3.1 Data available

At the training stage the data available are the attributes from previous flights' bookings plus the historical no-show information, *i.e.* for each passenger record, whether the passenger was present for travel. The value of the function *y(x)* which the network learns from these data can be considered as an estimator for the probability that a passenger with those attributes will "show" or "no-show". Further details of the data sets are given in section 2.4.1.

2.3.2 Method

The chosen solution method (neural network or CHAID) computes conditional averages for sub-populations defined by the combinations of attributes. There are two levels of prediction which need to be considered: the "microscopic", that is, the prediction of the no-show behaviour of each individual, and the "macroscopic", that is the statistical behaviour of the population as a whole. On recall, the probabilities thus produced by the network at the individual booking-record level can be used to produce an aggregate forecast of the number of passengers at a flight-operation / class level or at a flight operation level. If the bookings are considered to be independent then the individual probabilities can be summed to give an expected number of bookings on which travel will occur. The sum, over all the bookings on a flight, of the product of the individual booking level probabilities and the number of passengers associated with each booking gives the expected number of passengers. This can be compared with the number of passengers booked to give an estimate of the no-show rate.

2.3.3 Assessment of prediction quality

As alluded to in section 2.3.1, in this application we are not exclusively concerned with the accurate prediction for each individual of whether or not that individual will no-show. Rather the real-world measure of performance is the incremental amount of revenue which can be made on a particular flight by using yield management controls, traded off against the risk of offloads or downgrades if the forecasting is substantially in error. These numbers are directly affected by the number of passengers who *travel*, by comparison to the number

booked, in the various classes. Thus some post-processing of the network results is necessary in order to determine at what level changes in mix affect the final outcome. Differences in proportion of passengers with bookings of different selling classes within a cabin will affect revenue but not passenger numbers; differences in mix of no-show rate will affect passenger numbers and may or may not affect revenue.

2.4 Comparison 1:

The first part of the case study examines the relative performance of various neural network methods on the estimation of the no-show rate by flight-operation, that is at the level of flight number by day, but without the data being broken down by selling class. The training data used in this case was four weeks' data for five classes on five North Atlantic routes.

The neural networks used are standard MLP / back-propagation and radial basis function networks. In addition an MLP network with restricted connectivity was constructed. This was intended to represent domain knowledge of the structure of the data in terms of the attribute dimensions. For the back-propagation and RBF networks, the effect of different numbers of hidden units, different training times, different epoch sizes, and different weight decay parameters was investigated. The results were compared against a lookup table method in which the set of unique records in the training set was formed and an exhaustive lookup table built from the training data, giving the conditional average for each unique combination of attribute inputs which occurred in the training set.

2.4.1 Description of raw data

The first comparison used four weeks' data on a sample of flights from London, UK, to five cities in the USA. This data set contained 28,222 records. There were 473 unique combinations of the attributes, and the overall no-show rate was 19.3%. The test data used were from 06-12 July 1992, consisting of 7877 records for comparison 1 (five destinations). In the test data there were 321 unique combinations of the attributes, and the overall no-show rate was 18.5%

The booking attributes used in this study are as follows:

Seat request: This is a binary variable indicating whether the passenger has expressed a wish at the time of booking for a particular seat, or a preference for a non-smoking or smoking seat.

Sales area: The airline's designator of the region of the world where the ticket was sold. This broadly corresponds to a continent-level split. The categories are: {UK, Europe, Americas, Africa, East (including Asia and Australasia)}.

Class of booking: This refers to the airline's designation of *selling class*. There are about 25 selling classes[6] which in general refer to different ticket-types (e.g. APEX, Super-APEX, group specials, *etc.*: see section 2.1.1). The selling class would be expected to influence the no-show rate because the flexibility of the ticket, that is the penalty for late cancellation or no-showing, varies according to selling class, as does the price of the ticket.

"Online" indicator: This is an indicator of whether the booking was made close to the origin or the destination of the current leg of the journey. It is set to 1 if the origin airport is in the booking sales area; set to 2 if the destination airport is in the booking sales area and set to 3 otherwise. [7] A value of 1 thus usually indicates the first leg of a passenger's itinerary; a value 2, the last leg and a value 3 an intermediate leg.

Other attributes included were **origin, destination, flight number** and whether an inbound or onward **connecting flight** was involved.

The **output** (target) variable indicates whether the passenger(s) travelled on the flight as booked. [8] A separate variable is used to give the number of passengers in the *party* associated with a particular booking. Thus we can measure the no-show rate in two senses: by booking record, and by passenger record. In this study, the no-show rate is measured by booking record.

---

[6] The split of {first class, business class, economy} is referred to as the *cabin*.

[7] Not to do with "online" in the electronic sense.

[8] In the data there are some records where the target variable is recorded as neither a show nor a no-show. These are special cases such as 'non-revenue' passengers, *etc.*, or errors in the data. In the case of the use of only one output these will get represented as "shows". There is therefore a small but noticeable error rate in the *target* classification of no-shows and shows (Austin, 1993, McGrath, 1995). For the purpose of studying the effectiveness of various algorithms, this can be treated as noise.

For the first comparison exercise, the inputs were as follows[9]:

1. Seat Request = Yes
2. Seat Request = No
3. Sales area: UK and Europe
4. Sales area: North and South America
5. Sales area: Far East
6. Sales area: Africa
7. Sales area: other
8. Online indicator = 1. See above.
9. Online indicator = 2
10. Online indicator = 3
11. Online indicator = other
12. Class: F (First)
13. Class J (Club World)
14. Class M or S (Full Fare Economy / World Traveller)
15. Class B or K (Apex)
16. Class Q ("6th freedom" i.e. connecting from other flights)
17. Class V (discounted economy)
18. Class O (staff firm bookings *etc.*)
19. Class - other
20. Destination: Boston or Pittsburgh
21. Destination: Detroit or Washington
22. Destination: Newark or Chicago
23. Destination: New York (JFK)
24. Destination: Philadelphia
25. Destination: other
26. Connection: inbound, or both inbound and outbound
27. Connection: outbound
28. No connection
29. Origin: London Heathrow
30. Origin: London Gatwick
31. Origin: other.

---

[9] It should be noted that some of these inputs are known to be redundant in the case of this application.

	Class					
	F	J	M	Q	S	Total
attribute 1 (seat request)	908	5103	794	1668	1998	10471
attribute 2 (no seat request)	567	2520	4666	8489	1509	17751
attribute 3 (sales area UK / Europe)	584	3844	1749	6222	2089	14488
attribute 4 (sales area N / S America)	656	2998	1261	3474	1211	9600
attribute 5 (sales area Far East)	206	599	2238	174	108	3325
attribute 6 (sales area Africa)	29	182	212	287	99	809
attribute 7 (sales area - other)	0	0	0	0	0	0
attribute 8 (online =1)	479	3283	1187	663	1910	7522
attribute 9 (online =2)	570	2629	1075	2784	1044	8102
attribute 10 (online =3)	426	1711	3198	6710	553	12598
attribute 11 (online = other)	0	0	0	0	0	0
attribute 12 (Class F)	1475	0	0	0	0	1475
attribute 13 (Class J)	0	7623	0	0	0	7623
attribute 14 (Class M or S)	0	0	5460	0	3507	8967
attribute 15 (Class B or K)	0	0	0	0	0	0
attribute 16 (Class Q)	0	0	0	10157	0	10157
attribute 17 (Class V)	0	0	0	0	0	0
attribute 18 (Class O)	0	0	0	0	0	0
attribute 19 (class - other)	0	0	0	0	0	0
attribute 20 (destination Boston / Pittsburgh)	312	1501	1145	1894	1032	5884
attribute 21 (destination Detroit / Washington)	154	600	420	867	284	2325
attribute 22 (destination Newark / Chicago)	166	995	869	2525	601	5156
attribute 23 (destination New York (JFK))	843	4527	3026	4871	1590	14857
attribute 24 (destination Philadelphia)	0	0	0	0	0	0
attribute 25 (destination - other)	0	0	0	0	0	0
attribute 26 (connection inbound)	488	2285	2964	9124	856	15717
attribute 27 (connection outbound)	95	606	138	71	293	1203
attribute 28 (no connection)	892	4732	2358	962	2358	11302
attribute 29 (origin London Heathrow)	1475	7623	5460	10157	3507	28222
attribute 30 (origin London Gatwick)	0	0	0	0	0	0
attribute 31 (origin - other)	0	0	0	0	0	0
shows	1202	6570	4293	8623	2947	23635
noshows	273	1053	1167	1534	560	4587
noshow rate (%)	22.7	16.0	27.2	17.8	19.0	19.4

Table 2.1

Table 2.1 shows the training data summarised by the attributes and by class, showing the prior noshow rates.

	Class					
	F	J	M	Q	S	Total
attribute 1 (seat request)	270	1332	219	269	371	2461
attribute 2 (no seat request)	135	685	1540	2263	384	5007
attribute 3 (sales area UK / Europe)	148	999	441	1776	403	3767
attribute 4 (sales area N / S America)	173	681	275	494	234	1857
attribute 5 (sales area Far East)	69	238	943	87	45	1382
attribute 6 (sales area Africa)	15	99	100	175	73	462
attribute 7 (sales area - other)	0	0	0	0	0	0
attribute 8 (online =1)	122	902	301	146	377	1848
attribute 9 (online =2)	148	586	241	356	198	1529
attribute 10 (online =3)	135	529	1217	2030	180	4091
attribute 11 (online = other)	0	0	0	0	0	0
attribute 12 (Class F)	405	0	0	0	0	405
attribute 13 (Class J)	0	2017	0	0	0	2017
attribute 14 (Class M or S)	0	0	1759	0	755	2514
attribute 15 (Class B or K)	0	0	0	0	0	0
attribute 16 (Class Q)	0	0	0	2532	0	2532
attribute 17 (Class V)	0	0	0	0	0	0
attribute 18 (Class O)	0	0	0	0	0	0
attribute 19 (class - other)	0	0	0	0	0	0
attribute 20 (destination Boston / Pittsburgh)	0	0	0	0	0	0
attribute 21 (destination Detroit / Washington)	0	0	0	0	0	0
attribute 22 (destination Newark / Chicago)	0	0	0	0	0	0
attribute 23 (destination New York (JFK))	405	2017	1759	2532	755	7468
attribute 24 (destination Philadelphia)	0	0	0	0	0	0
attribute 25 (destination - other)	0	0	0	0	0	0
attribute 26 (connection inbound)	112	473	846	2250	212	3893
attribute 27 (connection outbound)	21	140	50	15	40	266
attribute 28 (no connection)	272	1404	863	267	503	3309
attribute 29 (origin London Heathrow)	405	2017	1759	2532	755	7468
attribute 30 (origin London Gatwick)	0	0	0	0	0	0
attribute 31 (origin - other)	0	0	0	0	0	0
	0	0	0	0	0	0
shows	333	1736	1335	2196	589	6189
noshows	72	281	424	336	166	1279
	0	0	0	0	0	0
noshow rate (%)	21.6	16.2	31.8	15.3	28.2	20.7

Table 2.2

Table 2.2 shows the training data summarised by the attributes and by class, showing the prior noshow rates, for flight number BA175 only.

2.4.2 Preprocessing and input selection.

The input attributes are categorical, and were encoded as follows. For frequently occurring values of the attributes a separate input variable (with value 1 or 0) was generated for each value. Exceptional and infrequently occurring values are lumped into an *"other"* input for each attribute. The inputs labelled "other" are not always populated, *i.e.* the variable may contain all zeros. In some cases two or more values are "lumped" together into one input. This is based on *preprocessing* done using models previously set up in SI-CHAID to determine which dimension values are *not* significantly different. Table 2.1 gives the breakdown of attribute data by dimension value within the attribute dimensions, at class level, for all flights considered together. Table 2.2 shows this analysis for flight BA175 alone. Notice that for some of the attributes, no positive instances are recorded in the database. This is because this set of data is a sample from a larger data set.

It is interesting to consider whether the use of SI-CHAID as part of the pre-processing stage biases the subsequent classification experiment in favour of SI-CHAID. It is the author's opinion that the two processes (pre-processing, including the grouping of some of the input variables together) and the classification stage, can be considered separately.

One potential problem with the type of coding used above is that, in long-term use, new values could occur. If these are rare enough to be acceptably included in the relevant "other" category, then this poses no problem. If however there is a significant number of new values, then the network would have to be retrained with a different number of input nodes. There is not at present an efficient, practically usable technique for dealing with a variable number of input quantities.

2.4.3 Neural network structures used

Partially connected MLP

In this network the elements in the input layer were grouped into groups representing the "dimensions" in the original attribute data, and each dimension value within a dimension was linked to a dimension node in the hidden layer. Two variants of this network architecture were investigated:

(a) A network with one hidden layer containing nine hidden units. Of these, seven correspond to the seven dimensions {seat request, sales area, online indicator, class, destination, connection indicator, origin}. The two additional hidden units are fully connected to the input layer. The structure of this network is shown in Figure 2.1.

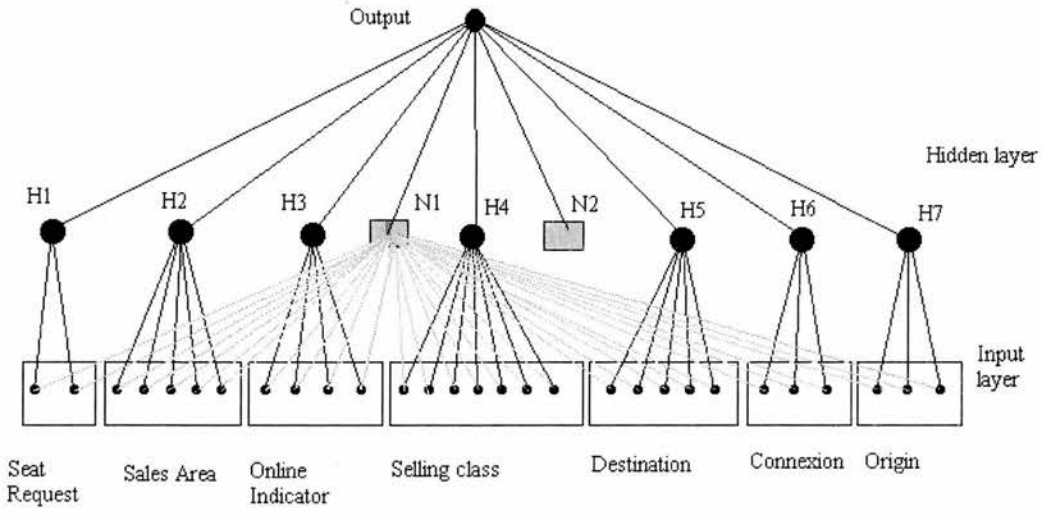


Figure 2.1: Structure of partially connected network (a). Note that the single hidden layer consists of seven nodes H1-H7 which are partially connected to the input layer (each being connected to a specific "dimensional" group of inputs), together with two extra nodes N1 and N2, which are connected to all the nodes in the input layer. Note that connections to N2 are omitted in the diagram for clarity. The hidden layer is fully connected to the output layer (there is no distinction between the H nodes and the N nodes except for their input connectivity).

(b) A network with two hidden layers. This was constructed as in (a) above (Figure 2.1) but with an additional hidden layer of 3 units between the first hidden layer and the output layer. These three units are fully connected from the first hidden layer and to the output unit. There are no "jump-layer" connections from the input layer to the second hidden layer or from the first hidden layer to the output layer. The results are presented in Table 2.3.

Fully connected MLP (Backpropagation) networks

The topologies of the fully connected MLP networks used were as follows: Networks (c) and (d): 31 inputs, 10 hidden nodes in first hidden layer; 3 hidden nodes in 2nd hidden layer; one output node (indicating 'show' or 'no-show' occurrence). Each of the hidden layers uses a *tanh* transfer function and the input data is normalised by the network software to lie between *-1* and *+1* . Network (e) uses 5 units in the second hidden layer. Networks (f) and (f1) use only one hidden layer. Network (f1) is the same as network (f) except that weight decay has been used, with weight decay parameters α, of 0.0001 for both the hidden layer and the output layer. Results for these networks are given in Table 2.3. For more detailed results on weight decay settings, see section 2.4.7.

Radial basis function networks

The implementation of the radial basis function network used in NeuralWorks (NeuralWare, 1993), follows Moody and Darken (1989), using Euclidean distance between the input records and the cluster centres, and the *k*-means algorithm to assign the cluster centres. Several different sizes of hidden layer (64, 80 and 100) were used, as reported in Table 2.3.

2.4.4 Calculation of error figure

Table 2.3 shows the Mean Absolute Percentage Error (MAPE) in the prediction of the number of bookings on which the passengers did present themselves for the flight. This number is calculated by summing the outputs of the neural network relating to all the bookings for each flight operation. The error is calculated as the absolute value of the difference between the predicted and actual number of bookings taken up, and this is expressed as a percentage of the actual taken-up bookings. Row (j) of the Table shows the mean of the predictions of booking takeup made by networks (a) to (i). The MAPE figure presented in Table 2.3 is given by:

$$E = \frac{\sum_C |y_i - t_i|}{\sum_C t_i} \times 100\%$$

where t_i is the target output for record i and y_i is the network output. C represents a combination of selling class, flight number and date. The figures presented in the table are averaged over classes F, J, M, Q and S, for flight numbers BA175, 179, 215, 223 and 297, for seven days of test data, for the week 06 - 12 July 1992.

Network	*Type*	*Units*	*Epoch size*	*Number of data record present-ations*	*Mean Abs. % Error*
(a)	MLP, reduced connectivity	31-9-1	1	100,000	6.8
(b)	MLP, reduced connectivity	31-9-3-1	1	125,000	5.1
(c)	MLP, full connectivity	31-10-3-1	16	40,000	5.3
(d)	MLP, full connectivity	31-10-3-1	4	100,000	4.8[10]
(e)	MLP, full connectivity	31-10-5-1	4	100,000	4.7
(f)	MLP, full connectivity	31-10-1	4	113,000	4.8
(f1)	MLP, full connectivity, weight decay=0.0001	31-10-1	4	100,000	4.6
(g)	RBF	31-100-1	1	75,000	4.9
(h)	RBF	31-64-1	1	50,000	4.6
(i)	RBF	31-80-1	1	50,000	4.8
	Average (arithmetic mean) of outputs of networks (a) to (i)	n/a	n/a	n/a	4.6
	Lookup Table	n/a	n/a	n/a	5.3

Table 2.3

Error rates (MAPE) for various MLP and RBF networks, on test set (data for 06-12 May 1992), for comparison 1 (using data from flights to five North American destinations).

10 Standard deviation, measured over 10 runs, = 0.25

2.4.5 Results

The comparison results are tabulated in Table 2.3. Results given are mean absolute percentage error (MAPE), of predictions of the number of bookings taken up, which is the aggregate of the individual no-show probabilities generated by the network or lookup table on booking records. Results are shown for two variants of the partially connected MLP, five variants of fully connected MLPs, three RBF networks, together with the error of the average of these network results, and the result of the lookup table. The first two rows in Table 2.3 show that network (b), with the additional hidden layer, performs substantially better than network (a); (MAPE of 5.1 against 6.8 for network (a)). Nevertheless, structuring the network in this way was less effective than using a fully-connected, standard back-propagation network (for which the smallest MAPE was 4.6, for network (f1)).

Comparison with Fully Connected MLP

With the exception of network (c), which used an epoch size of 16 (i.e. the weights were updated after the presentation of every sixteenth training record), the results from these networks were within the range MAPE=4.6 to 4.8%. The use of a second hidden layer (networks (d) and (e)) made only a marginal improvement, as did the use of weight decay (network f1). See Table 2.3. Further experiments using different weight decay parameters are presented in Section 2.4.7.

Comparison between MLP (backpropagation) and RBF networks

The results from the networks described above were also compared with those from three Radial Basis Function (RBF) networks, with 100, 64 and 80 hidden units respectively. The results are given in Table 2.3. The performance of the fully connected MLP networks and the RBF networks was similar to within 0.1 percentage points of MAPE. It will be seen from Figure 2.2 that the variations between the RBFs and the MLPs were of the same order as fluctuations in the performance of each: there was no systematic difference. Note that a higher number of hidden units was required with the RBF units than with the MLP, as would be expected, given that the RBF networks form a local representation of the data density. The best results were obtained using a RBF with 64 hidden units.

Graphical summary of results

Figures 2.2(a) and 2.2(b) show the neural network forecasts at flight level for the networks described in Table 2.1, for flight BA175 for a week of test data.

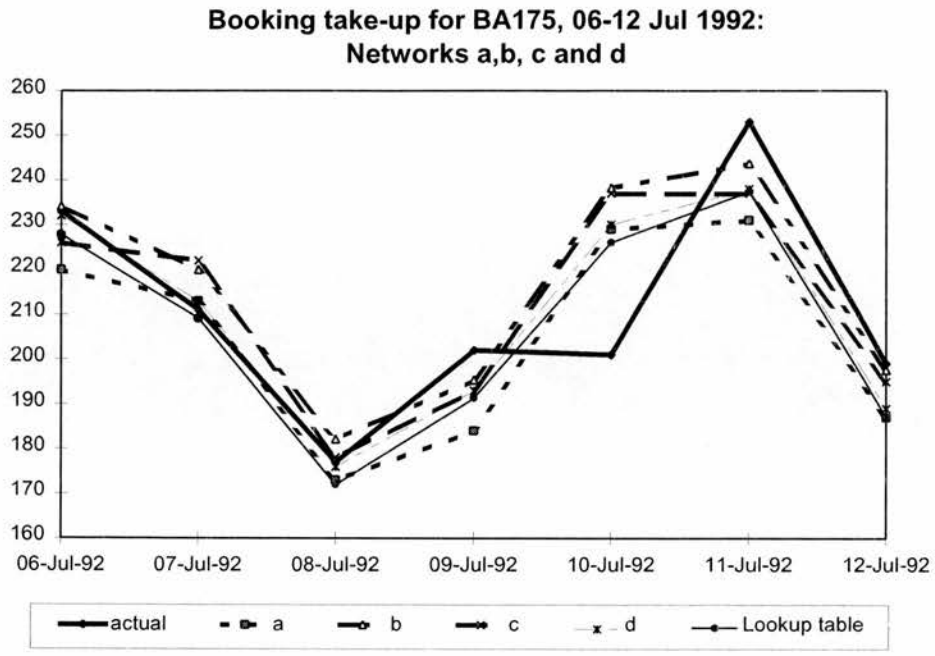


Figure 2.2(a): Forecast booking take-up for BA175, 06-12 Jul 1992: Results from Networks a,b,c,and d are shown together with actual figures and lookup table results.

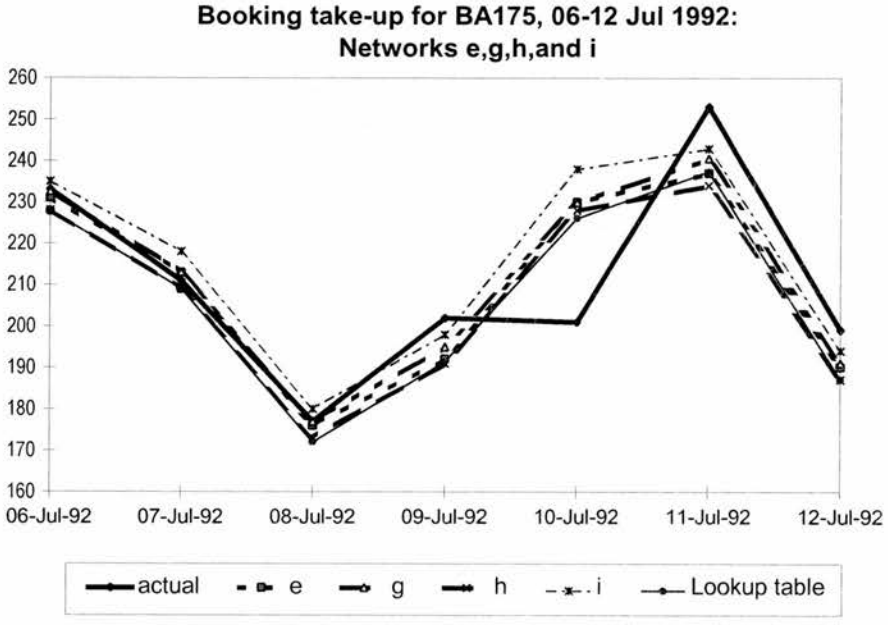


Figure 2.2(b): Forecast booking take-up for BA175, 06-12 Jul 1992: Results from Networks e,g,h and i are shown together with actual figures and lookup table results.

2.4.6 Effect of training time on error rate for MLP

The error rate on the test set is shown in Figure 2.3, as a function of the number of training records, for the MLP 31-10-5-1 network (with epoch size 4 and no weight decay). The curve flattens out at 100,000 epochs and starts to rise at 200,000 epochs where the network is considered to be overtrained. This result is in accordance with many other published accounts of the effect of training time on a MLP network.

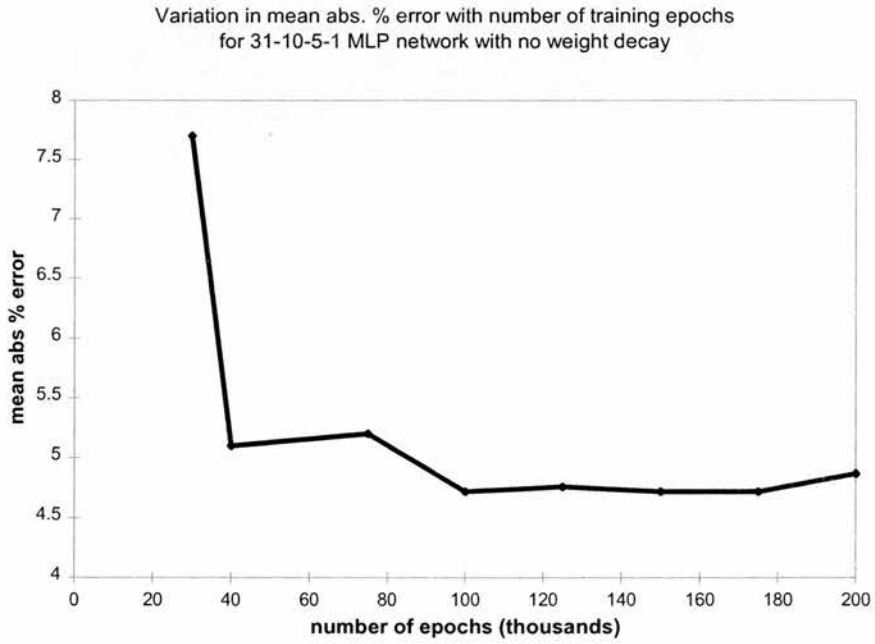


Figure 2.3: error rate (MAPE) on the test set as a function of the number of training records, for the MLP 31-10-5-1 network with epoch size 4 and no weight decay.

2.4.7 Effect of weight decay

Regularisation methods attempt to alleviate overfitting and increase generalisation ability in a neural network model, by the addition of a complexity penalty term to the error function. A common form of regularisation is the use of *weight decay* (Hinton, 1987, Bishop, 1995). In weight decay a term in the sum of the squares of the weights is added to the error function,

$$E = \frac{1}{2}\sum_{p}(y_p - t_p)^2 + \alpha \sum_{i} w_i^2 ,$$ where p indexes the training patterns, and α is the weight decay parameter.

Different parameters α can be used for different layers of the network. The purpose of weight decay is to encourage small values of the weights and thence to encourage the fitting of the data by the network with as smooth a function as is compatible with representing the data. Table 2.4 shows the effect of different weight decay values on different layers of the network. The results are for the 31-10-1 MLP network, and show that, in this application, weight decay can slightly improve generalisation performance of the network, but that the effect is sensitive to the weight decay parameters chosen. Of those studied, the only case in which the use of weight decay improved the network's performance was for the case of weight decay = 0.0001 for both hidden and output layers.

weight decay parameter for hidden layer	weight decay parameter for output layer	mean abs. % error (MAPE)
0	0	4.8
0.001	0.001	6.1
0.001	0.0001	5.4
0.0001	0.0001	4.6

Table 2.4
Effect of different values of weight decay parameters on MAPE for the case of the 31-10-1 MLP network.

2.4.8 Effect of random initial weights

Different random settings of the initial weights can have an effect on the solution to which the network converges. This can be investigated by running the same network for a number of different initial weight settings. For network (d), the standard deviation over 10 runs for the error figure recorded was 0.25. Assuming that this value is typical for the rest of the networks, the difference between networks (d) to (i) and the lookup table results is significant at the 95% level, whereas the differences between these networks themselves are not significant at the 95% level.

2.4.9 Summary of Comparison 1

At the flight-operation level, the multi-layer perceptron and radial basis function networks, as detailed in Table 2.3, were shown to give more accurate predictions of the number of passengers presenting for a flight, based on the booking attributes, than the method of using a lookup table.

The next section considers the comparison at a lower level of aggregation and compares the neural network results with those of a tree-based method.

2.5 Comparison 2

Section 2.4 examined the effects of different neural network architectures and parameter choices, at the flight-operation (flight number by day) level. In this section, comparisons are carried out at (flight number × day × selling class) level. The purpose of this section is twofold: firstly to assess the performance of the neural network methods at the lower level of data aggregation, which is the level at which the forecasts are required for subsequent use in the yield management process; secondly to compare the performance of selected neural network algorithms with conventional algorithms. A subset of the neural network approaches investigated in Comparison 1, is compared with three SI-CHAID models, and the results from using a lookup table at (flight number, day, selling class) level. The data for this study were restricted to flights from London to New York and Boston.

2.5.1 SI-CHAID models used for comparison

The following booking attributes were used to develop the SI-CHAID models:

Seat Request	(Yes / No)
Sales Area	(UK / N. America / S. America / Europe / East / Africa)
Card Usage	(Charge Card / Frequent Flyer / Executive Club / Other / Blank)
Class	(B, F, J, K, M, Q, R, S, V, Other)
Lead Time[11]	(>6 mo., 3 to 6 mo., 2 to 3 mo., 1 to 2 mo., 21 to 28 days, 14 to 28 days, 7 to 14 days, 0 to 7 days)
Departure Time	(0600-1200, 1200-1700, 1700-2359, 0000-0600)
Booking airline	(BA, Other)
Flight number	
Origin	
Destination	
Day of Week	
Connection indicator	(inbound connection, outbound connection, both, none)

[11] Elapsed time between when booking was made and departure.

Three CHAID models were configured, as follows: **Model 1** used eight weeks of training data (25 April - 05 July 1992) and allowed tree depth up to 6 levels. **Model 2** used the first four weeks of the training data and allowed tree depth up to 6 levels. **Model 3** used eight weeks of training data and allowed tree depth up to 5 levels.

2.5.2 Neural network models for Comparison 2

Five of the networks from "Comparison 1" are used for Comparison 2: the reduced connectivity network with a second hidden layer of three units (network (*b*)), fully connected MLP networks with one hidden layer (network (*f1*) and two hidden layers (network (*d*)) and radial basis function networks with 100 hidden units (network (*g*)) and 64 hidden units (network (h)). These are listed in Table 2.5. The implementation details are as in NeuralWare (1993).

2.5.3 Results on prediction of booking take-up by class

Network	*Type*	*Units*	*Epoch size*	*Number of data record present-ations*	*Mean Abs. % Error*
(b)	MLP, reduced connectivity	31-9-3-1	1	125,000	9.96
(d)	MLP, full connectivity	31-10-3-1	4	100,000	8.73[12]
(f1)	MLP, full connectivity, with weight decay=0.0001	31-10-1	4	100,000	8.94
(g)	RBF	31-100-1	1	75,000	8.13
(h)	RBF	31-64-1	1	50,000	9.07
	SI-CHAID model 1	8 weeks		6 levels	11.11
	SI-CHAID model 2	4 weeks		6 levels	11.11
	SI-CHAID model 3	8 weeks		5 levels	10.98
	Lookup Table	n/a	n/a	n/a	8.73

Table 2.5: MAPE values for neural network, SI-CHAID and lookup table methods on the second comparison problem, prediction of booking take-up by class on two UK-North American flights (BA175 and BA215).

[12] Standard deviation, measured over 10 runs, = 1.3

From Table 2.5 the following results can be seen. At the date / flight number / class level, the error rates in all cases are higher than for the date / flight number only split. The standard deviation across sets of ten runs is also higher, at 1.3 instead of 0.25. Thus, a comparison of the various techniques at this level is more difficult. However, the following consistent results were apparent: the neural networks in all cases gave lower error rates than the SI-CHAID models, according to the MAPE measure. Specifying additional levels for the SI-CHAID models made little difference to their accuracy. This shows that, for practical purposes, a five-level tree represents all the systematic variation of interest in this data set: the benefit of any further subdivision is outweighed by noise. Similarly, providing eight weeks' data rather than four weeks' data to the CHAID method did not produce any improvement, implying that either few combinations existed in the second four weeks' data which were not in the first four weeks' or that the first four weeks' data were sufficient as a training sample on which to build a model of the principal effects and sources of variation.

The sparsely connected multi-layer perceptron network still performed poorly compared to the fully connected networks, reflecting the importance of higher-order correlations in this task and the lower ability of the structure described in Figure 2.1 to represent them. The difference between the performance of the RBF networks and the MLP networks was not statistically significant at this level, but the network which gave best performance in the trial was the RBF with 100 hidden units. The advantage of using a radial basis function network can be understood in terms of the radial basis function representation being more local than that of the multi-layer perceptron. Notice that the 100-unit RBF outperformed the 64-unit RBF network in this task, whereas in Comparison 1 the opposite was true. This can be understood because at the lower level of aggregation there are more partitions of the data to be represented, so an RBF with higher capacity for representing local structure should be expected to do better.

Detailed examination of the errors by class and flight-operation show some differences between the models, and that the neural network models generally predicted higher no-show rates than the SI-CHAID models, *i.e.* the bias of the two types of model consistently differs. There are also, at this level, some "outliers", with anomalously high actual no-show rates (e.g. M and S classes on 10Jul). The specific cause for these is not known but a system error, operational disruption or the no-show of a large party could cause such results.

Results for class F are particularly sensitive to random variation at this level, as the sample sizes are small. On this basis, there would be good reason to suggest that the class-F data be treated separately, perhaps with a simpler model structure.

Only one of the neural networks at the date / flight number / class level achieved better performance by the MAPE measure than the lookup table, which for this comparison, was constructed from the training data at flight number / class / booking attribute-combination level. This shows that most of the pattern in the data is captured by the training data. If the number of unique combinations in the data is not too large, and if the unseen data do not vary too widely from the training data, then a lookup table is an effective solution to this type of conditional expectation task. The advantage of a neural network should become stronger as the generalisation requirement increases, given the supposed capabilities of neural networks in interpolation in high-dimensional spaces.

2.6 Discussion and conclusions

Results of the comparisons in this Chapter show that the neural network methods used in this trial (multilayer perceptrons and radial basis function networks) can perform at least as well as conventional methods in the task of no-show forecasting from airline booking attribute data, when the network structure and parameters are chosen appropriately. Networks with the best performance when the data was not broken down by selling class did not necessarily give the best results in the second comparison (though there were some other differences between the data in comparisons 1 and 2). Thus the variation in performance between different neural network structures, and types, is outweighed by application-specific details: in this case, the hierarchical nature of the data.

To summarise, it was found in this study, as in the case of other comparative studies of a similar nature, that the improvement obtained by applying the neural network method, though statistically significant at a 95% confidence level at a particular level of aggregation of the data, was not dramatic and depended on the level of aggregation at which the forecasts were made.

However, following the work of Richard and Lippmann (1991), as developed by Bishop (1995) and others, the outputs from the neural network are interpreted as posterior probabilities of a positive outcome. These can then be used in a further model or post-processing stage to arrive at an estimate, or distribution, of a value function which is more meaningful in terms of the domain, and can take into account real-world constraints which cannot easily be represented in the neural network model itself. In the airline domain the value function would be revenue or yield, and the constraints may be cabin sizes, upgrade policies, and other yield management "controls".

In a wider context, the no-show forecasting task can be seen as analogous to tasks such as the prediction of the outcome of a medical procedure or a commercial relationship (such as a loan or insurance policy) where the information available consists of historic instances characterised by a moderately large number (a few tens) of mostly binary attribute variables, together with a positive or negative outcome.

In terms of the aims stated at the beginning of this thesis, the evidence gathered in this chapter, taken as a whole, supports the hypothesis that neural networks are no worse than conventional systems when working with real-world problems and data.

In the next two chapters, the investigation of the effectiveness of neural network methods on real data will concentrate on the task of monitoring aircraft engine condition, using data from flight data recording apparatus. The condition monitoring task can be treated in terms of attributes (measurements taken in the engine and on the aircraft) and outcome (good or poor condition), with a neural network predictor giving a value which can be interpreted (within bounds) as a probability (in a way similar to the application given in this chapter and basically a classification approach but with continuous input variables). Alternatively, condition monitoring can be approached in terms of the concepts of regression or novelty detection, with the aid of a dimension-reducing scheme such as Kohonen's self-organising map. Chapter 3 gives an overview of such methods. Chapter 4 describes the application of neural networks in the context of nonlinear classification and regression to diagnosis and condition monitoring tasks using operational data. Chapter 5 describes a refinement to the approach of using neural networks in a context-based novelty detection capacity for general aircraft engine condition monitoring. Chapter 6 returns to the question of the real benefits of neural networks as a statistical tool in such applications.

3. Neural networks for Condition Monitoring

3.1 Introduction

Having stated, in Chapter 1, the null hypothesis that neural network techniques are "no better than conventional techniques" for handling real-world problems, and in Chapter 2 set out to provide evidence against this hypothesis for the case of no-show forecasting using booking attribute data, we turn in the second part of the thesis to another important area of airline data handling, in which we wish to test whether neural networks provide useful capability. This area is condition monitoring. The solutions of forecasting and monitoring problems both involve the characterisation of underlying relationships in data and a statistical description of the extent of variability of the data and of any trend behaviour. Once such a model is formed, the monitoring task entails a "prediction" of the normal behaviour of the system, or plant, under expected or known conditions, the measurement of any deviations from the nominal condition, and the raising of appropriate alerts to the user.

These tasks are routinely undertaken using linear statistical approaches, but the limitations of these are usually manifested in a restricted ability to deal with interactions between a number of variables. The use of a non-linear technique, for example, artificial neural networks, should extend the scope of condition monitoring systems. The second part of this thesis, that is, Chapters 3, 4 and 5, sets out to provide some evidence from processing of real data, to determine to what extent this is true in practice, and hence to provide further refutation of the null hypothesis stated in Chapter 1.

This chapter introduces the domain of condition monitoring, focussing the discussion particularly on computational techniques for aircraft engine condition monitoring. The first section outlines the condition-based maintenance and condition monitoring approach to the management of maintenance strategies, describes how neural net methods might be applied in this area and examines a sample of published work. Principles in applying neural networks to condition monitoring are stated and a number of different approaches using feed-forward networks are defined. The unsupervised approach is then introduced in the form of the self-organising map and its advantages and limitations are discussed. Descriptions of case studies and results are presented in chapters 4 and 5.

3.1.1 Maintenance strategies

Three strategies for deciding upon maintenance action are as follows:

(a) On breakdown.

While this is acceptable in many types of equipment and for components which are not safety-critical, this strategy is not acceptable for safety-critical plant or where a breakdown would mean a very large loss of revenue.

(b) Fixed lifetime.

In this case components are removed and replaced after pre-determined, planned times. This strategy has to be used for many components in accordance with regulations laid down by regulatory authorities.

(c) On condition.

In this strategy, the condition of the plant is monitored by inspection or automatically using sensors, and components repaired or replaced when this is necessary to ensure peak running condition of the equipment.

For modern jet aircraft engines a mixture of the "fixed lifetime" and "on-condition" strategies is used. Limits on intervals between maintenance checks are determined by the regulatory authorities. In addition, condition-based maintenance is practised, and a variety of inspection and data-driven techniques are used to assess condition.

3.1.2 Conventional approaches to condition monitoring

Physical methods for engine condition monitoring for gas turbine engines include visual inspection, using an instrument known as a boroscope to inspect inaccessible parts of the engine; Spectrographic Oil Analysis (SOAP) where a spectrometer is used to analyse chemical residues in the lubricating oil, in order to determine areas of wear, and Magnetic Chip Detection, where a magnetic plug is used to trap and identify tiny ferrous metallic fragments in oil, also as a means of identifying which components may be undergoing wear.

In addition to the physical inspection and mechanical measurement methods of assessing engine condition, modern jet aircraft engines are extensively instrumented with sensors for temperature and pressure at various points in the gas path and speed of the rotating

components.[1] As far as methods based on analysis of engine data are concerned, the conventional approach to computational or statistical engine condition monitoring is to generate nominal values for the parameters according to a thermodynamic model, then to analyse the differences of the measured values from the theoretical "book" values, signalling a warning when a difference exceeds a certain predetermined limit. (This is known as an *exceedance.*) Apart from checking for exceedances, condition monitoring systems are used to check whether engines are running at maximum efficiency. A rise in exhaust gas temperature, for example, is one indication of ineffiecient operation.

3.1.3 Use of neural networks for condition monitoring and fault detection

An artificial neural network can be used for condition monitoring in one of two ways: firstly *to diagnose and investigate a specific fault mode*; the monitoring function being to provide an early warning from any trends in the data which may point towards the development of a fault of known character. For this task a supervised learning approach is used if the data are available. The second mode of use is as a *novelty detector*; whereby one trains a network on a sufficient sample of "normal" operation that anything outside this envelope can be treated as an "alert" state.

The domain of aircraft engine condition monitoring in an operational context provides some interesting challenges to a monitoring algorithm. Unlike the case of monitoring of static machinery, the operating conditions are continually varying and this needs to be taken into account. The flight envelope is divided into phases consisting of "idle", "taxiing", "take-off", "climb", "cruise", "descent", and "landing". In searching for signs of incipient fault modes the greatest area of interest is the take-off phase. From the point of view of neural network analysis of the data, this time is when the parameters are varying at their greatest rate and transient effects are present due to the time lag, albeit small, between changes in control variables and the response from the engines and the airframe. For the purposes of long-term monitoring and analysis of trends, however, a more stable basis is required; this is provided by using cruise data, which is captured at stable cruise points which are determined from the data by a conventional condition monitoring system (for example, COMPASS: Kerry *et al*, 1988-91). This provides the opportunity for a separate analysis which is in some senses complementary to that using the continuous data.

---

[1] The extent of sensor provision on aircraft engines is not, however, as great as for ground-based gas turbines where many hundreds of parameters are recorded.

A desirable property in a trained neural network (or other mathematical model) for these purposes is to detect novel situations while being invariant to variations within normal envelope of operation. There is thus an "experimental design" challenge in the formulation of training data and strategies, to ensure that the network "sees" data which is representative of all normal aspects of a flight envelope (or that it has sensible defaults). This is particularly important where the data is highly multidimensional, or there are specific, non-fault related exceptional conditions which occur infrequently.

3.1.3 Literature survey

A number of workers have broached the idea of condition monitoring using neural networks. However, the majority of these have reported trials on artificial (simulation) data or on test rigs. Few studies have used operational data. Much of the reported work refers moreover to static gas turbines for example in power station or industrial plant usage. In such cases the operating environment is very much less variable and the task of learning a monitoring task commensurately more straightforward.

Many authors (for example, Witcomb & Skitt, 1990) concentrate on the Kohonen self-organising map (Kohonen, 1982) and auto-associative back-propagation methods (Rumelhart *et al.*, 1986). Several workers have used vibration data and first taken a fast Fourier transform (FFT, Cooley and Tukey, 1965), or more recently a wavelet transform (Daubechies, 1992) and used this, or some features derived from it, as the basis of the input to the neural network (Murray and Penman, 1996).

Patel and Kadirkamanathan (1996) have used a version of the Resource Allocating Network (RAN, Platt, 1991), a constructive network using radial basis functions, to monitor a gas turbine engine, using data from a detailed and extensive simulation. The RAN approach has the advantage that it allows for the construction of new hidden and output nodes to represent novel situations, and the formation of new nodes can be used as an indicator of departure from normal operation, once a stable model of the envelope of normal variation has been established. Patel and Kadirkamanathan used data from six sensors: High pressure compressor delivery pressure (P30), High Pressure Compressor Delivery Temperature (T30), Low Pressure Spool Speed (N1), High Pressure Spool Speed (N3), Turbine Gas Temperature (TGT) and Fuel Flow (designated WF or FF). Their *simulated* fault data were generated by simulating flying at 30000 *ft* at Mach 0.8 (representative of cruising conditions of an aircraft), and consisted of simulated fuel interruptions (decreasing the fuel flow by 10%, 30% and 50% over periods of 2 and 5 seconds), and simulated bearing faults. Their data was "sampled" every 100 milliseconds for 25 seconds. Patel and

Kadirkamanathan modelled the effects of the Electronic Engine Controller (EEC) and their results show how a simulated fault caused an excursion from a normal operating point in the space of N1 *vs* P30 and how the system returned to this point under the influence of the (simulated) EEC. This approach (looking at excursions from a control point) is suitable for the case of a 2-dimensional comparison, in particular as it is straightforward in two dimensions to visualise the excursion from the control point and the effect of the control system. This would not be the case for highly multivariate operational data without the additional use of some form of dimensionality reduction or visualisation aid.

Atkinson and Woollons (1996) have used a combination of prediction and classification networks arranged in a control-loop setup. Referring to the context of electro-hydraulic systems[2], they point out that although steady state faults can be diagnosed by classifier neural networks, steady-state errors are compensated for by feedback control systems, and that some faults may show effects only during transients. Atkinson and Woollons used two neural networks. The first network was trained as a one-step-ahead predictor to give an estimation of the system outputs assuming a normal, fault-free operation. The residuals between measured and predicted outputs were used to detect faults. The second network was trained to classify the system behaviour as normal or faulty based on features in the residual vector.

Azzam and Hazell (1996) used neural networks amongst other techniques in an appraisal of advanced diagnostic strategies for helicopter health and usage monitoring, using simulated faults, within a simulation of the aircraft dynamics. Eleven simulated fault types were diagnosed from 24 vibrational features.

Aznar Fernandez-Montesinos *et al*, (1994) have used real flight data. They used a hybrid approach involving neural networks and expert systems to perform condition monitoring on *CFM56-3* engines on *Boeing 737* aircraft operated by the Belgian airline *Sabena*. A Kohonen network was used to analyse single time-differences in the "deltas" (differences from book values) of Low Pressure Spool Speed (N1), High Pressure Spool Speed (N2), Exhaust Gas Temperature (EGT), Fuel Flow and throttle position. An expert system was used to aid the engineers in interpretation and analysis of the results from the neural network and to cope with step changes such as component or engine replacement.

---

[2] Atkinson and Woollons' experiments were performed on test rig "consisting of a hydraulic actuator system with positional feedback control".

3.2 Principles behind neural net formulations

Two main principles are at work in the use of neural-network and related methods for monitoring. These are *dimensionality reduction* and *novelty detection*.

3.2.1 Dimensionality reduction

In the case of engine condition monitoring one knows that the fundamental system underlying the data can be characterised by physical laws, and is thus basically deterministic. We are interested in any departure from the expected performance of an engine under specified or measured conditions. Modern instrumentation on recent aircraft types is capable of logging many dozens of variables, and although monitoring systems process this data and measure differences against standard models it is still necessary to inspect visually a large number of variables.

Apart from neural networks, methods of dimensionality reduction include Analysis of Variance, Principal Components Analysis (Jolliffe, 1986), Sensitivity Analysis, Multidimensional Scaling (Cox & Cox (1994), Goodhill, Simmen and Willshaw(1994), Ripley (1996); Lowe & Tipping(1996)) and the Sammon Mapping (Sammon, 1969).

Of these probably the most pertinent and widely used is principal components analysis, which consists of finding the eigenvectors and eigenvalues of the covariance matrix. The directions of the first k eigenvectors are (in rank order) the directions of highest variation.

Neural network methods for dimensionality reduction are usually based on an autoassociative network (Rumelhart *et al.*, 1986) or a Self-Organising Map (SOM: Kohonen, 1982 *etc*) An autoassociative network with only one hidden layer is effectively an iterative method of performing linear principal component analysis (Bourlard & Kamp, 1988; Baldi & Hornik, 1989). Extending to the case of three hidden layers (so-called 5-layer networks) allows for the "principal component" mapping to be *non-linear* and can result in a more efficient coding (Kramer, 1991). Self-organising maps are further discussed in section 3.5.

3.2.2 Novelty Detection

The elements of novelty detection are firstly that a distance measure must be defined and secondly that it is necessary to be clear about the manifold of space or population of data points from which the "novel" data is to be distinguished.

One approach would be to define novelty purely in terms of some generalised distance measure and relate unseen data points to the general population of training data by giving a probability level of the data lying within a certain distance according to a certain distribution. The concept of novelty detection is very closely related to the concept of outlier rejection in statistics. In the statistical framework, one would examine the distribution of the data, parameterise it and derive a confidence limit for a given confidence level. Then a statistical test would be used to determine whether a new data point was considered to belong to the same distribution as the previous data. This approach requires the distribution to be known or approximated over the whole data space, and preferably for it to conform to a well-known, tractable mathematical form. The neural network framework seeks a distribution-free procedure where one does not impose a form for the distribution. On a simple level, the distance itself or a frequency count could be used as a measure of novelty.

The issue of novelty detection also occurs when using labelled data and a *classification* approach. In this case data records which are substantially outside the manifold of the training data are highlighted when the class membership probabilities (network outputs) do not sum to one, or when two mutually inconsistent classes are both seen to be populated with appreciable probability mass. These effects can also arise as the result of a poorly trained network or a network with too little capacity, and so cannot be reliably used as a definitive measure of novelty. However, if suitable validation steps are carried out, this method can be used as an informal novelty detector. Tarassenko (1996), and Nairac *et al.* (1997a,b) discuss the practical issues associated with performing novelty detection in an engine condition monitoring context. They concentrate on the specific difficulties of working with a relatively small sample (around 50 engines) of data. Bishop (1994) also discusses novelty detection.

3.3 Issues in applying neural networks to condition monitoring

This section describes how some of the issues and considerations in applying neural networks, relate to the area of engine condition monitoring, and describes some particular features of condition monitoring applications.

3.3.1 Preponderance of one type of datum

Consider the case of a *classification system* in which the vast majority of the data (training and test data) belong to one of the classes in particular. In this situation the class prior probabilities are uneven. In the case of extreme unevenness of class priors, especially in the case of high noise in the training data, a supervised neural network such as backpropagation will achieve minimum error by ignoring the minority class altogether and classifying all the examples as the majority class.

A number of possible approaches have been outlined to tackle this problem (see section 2.2.2). Bishop (1995) advocates the practice of carrying out adjustments based on the class priors.

Problems which arise in training an MLP on data in which the frequency of different classes is very uneven can be circumvented to a certain extent by the use of a Kohonen map, in which the number of effectively different regions is determined not by the frequency of occurrence but by generalised distance. It is however necessary to identify outliers which are due to data errors or sensor faults for example.

3.3.2 Quantity of data necessary

It is preferable to view the quantity of data required to learn a mapping or task in terms of the *extent of the space which is covered* (particularly the extent of space covered by the data which overlaps with the region of interest). In the case of condition monitoring, however, particularly in the case where labelled category or classification data is not present, the only discriminant between the "good" and "bad" portions of the space is the extent to which the network models the data. That is we wish to model the good data "well" (that is, the residual being low) and the bad data "poorly" (that is with a high residual). Thus the answer to the question of "how much data?" cannot be addressed in the same way as in application areas where the interest is in the network's ability to generalise to unseen cases.

3.3.3 Overfitting

As noted above, the prevention of overfitting is not an overriding issue in this type of application, as one is concerned with controlling the extent to which generalisation is allowed to occur. In general the test and training sets will have closely similar statistical behaviour and it is any deviation from this which we *require* the system to signal by giving a high residual or distance measure. These issues are explored further with practical examples in section 4.4

3.3.4 Standard Day correction

Conventional condition monitoring systems for airborne engines adjust the thermodynamic data to a standard temperature and pressure framework. This is known as correcting to standard day conditions. In a neural network approach to engine condition monitoring, one has the choice of correcting the data to standard day conditions using an explicit formula or of leaving the data uncorrected and thereby implicitly including the learning of invariance to operating condition variations in the overall task of the neural network. For this work the full correction algorithm was not available so the latter approach was used. Further discussion will be given in terms of the results presented for the regression approach to condition monitoring in section 4.5.

3.3.5 Sensor faults

Engine data is recorded from a wide range of different kinds of sensor, each of which may exhibit fault modes. Sensor faults may show up in the following ways in the data. Firstly, values for some variables may be missing altogether. Secondly there may be a very obvious difference in the reading from the normal value. Such faults could be detected and filtered out in the pre-processing stage to reduce this case to the case of a missing variable. One option here is to discard the whole data set if this is the case; this would be unfortunate, as the ability to perform condition monitoring is required even in the presence of such (known) sensor faults. A further problem arises in the case where a sensor fault occurs in a dataset containing an engine degradation, where the sensor fault may mask the appearance of an engine-related fault mode.

The options available in dealing with sensor fault data are as follows:

- One may wish to *discard the affected records* from data set. This is an acceptable option if the frequency of sensor faults is low and in the absence of coincidence between the sensor faults and genuine fault data or other data showing a condition of interest which is infrequent in the training set.
- *Exclude the input variable* from the input set. This option is not viable if sensor faults are possible on all the input variables. For an individual variable, this is the only option if data is missing in a very high proportion of cases.
- *Use an average value* (for example, in the case of a four-engined aircraft, the average from the other three engines). This method can be used if data is missing in a low proportion of cases and there is good reason to believe the missing case will, in all other respects (for the purposes of the model), behave like the rest.
- *Impute the missing value* using linear regression or a separate neural network by modelling from all (or a subset of) the other inputs (for suitably sampled training data). Imputation using a neural network is the approach which has been used to overcome deficiencies in the P3 and DT25 variables. (see chapter 4).
- *Indicate the missing values* by hand (or by pre-processing) and teach the network to ignore particular input values as being meaningless. Here one runs across the general question of making a net invariant to a particular value or range of values of an input variable. One solution would be to train with constant output over the region concerned.

Detection of sensor faults

A neural network may be used to *detect* sensor faults in the data. Usually the first result derived from neural network analysis on a set of sensor data is an indication of the quality of the data and an identification of its deficiencies. However there may be simpler ways of performing this task on a set of data. Preliminary data analysis using standard linear techniques will usually identify data deficiencies.

Use in validating the network

Sensor fault data, once identified, can be used as "pathological" test data to explore the reaction of the network to data which is known to be out of range, as part of a validation process.

3.4 Feed-forward neural nets for engine condition monitoring

Condition monitoring applications fall into two broad categories. The first is where one does not impose any prejudgment as to the nature of fault modes. This method is tantamount to pure *novelty detection*, and the task becomes that of fully characterising the extent of variation of normal data, while regarding anything unfamiliar as abnormal until proven otherwise. The second approach is valid where one is investigating a *known or suspected problem* or fault mode. In this case models can be set up which are much more explicit and focussed. This section considers how feed-forward neural network architectures can be applied to condition monitoring, fault detection and diagnosis in these two senses.

3.4.1 Regression approach

The regression approach consists in modelling an output variable as a function of some input variables, where it is expected that a strong and consistent relationship will obtain. The example here is the modelling of Exhaust Gas Temperature (EGT) or Turbine Gas Temperature (TGT), as a function of a set of recorded parameters which include environmental parameters such as altitude, ambient pressure and temperature and Mach number, and engine parameters such as spool speeds (N1, N2, N3), Engine Pressure Ratio (EPR), and pressures and temperatures measured at various points in the engine gas path (P160, P25, T25, P3, T3). The method is as follows. A network (in this case a 2-layer, fully connected MLP trained using error back-propagation, is trained on data from one particular engine (say an engine which is known to be running normally). Data from the other engines are then passed through the trained network to give the value for the output variable which would have obtained if the engines had been functioning according to the same model.

This method is to be contrasted with just measuring the difference between the temperature reading for the two engines because although some of the input parameters in the two models will be the same the measured engine readings may well differ between engines in the normal course of operation due to ageing and differences in time since service checks, *etc*. Nevertheless in most cases the discrepancy between result of taking the difference between the network output and the measured temperature and the result of measuring the temperature difference between the two engines will be small.

Advantages of the regression approach are: that it is readily understandable to the engineers, the results are easily presented graphically, and trends away from the model induced in training are readily apparent. However, such a regression approach can really only be regarded as a first stage of a complete system for fault isolation, as hypothesis testing and thresholding mechanisms will be required in addition in order to make the results clear and useful from an engineering viewpoint. See section 4.4.

3.4.2 Auto-associative approach

After training an autoassociative network, two quantities provide monitoring information; the outputs from the hidden layer (and their differences from a reference model); and the reconstruction error. The reconstruction error for the training data can be used to give an indication of areas where the model is weak or unreliable and reconstruction error in test data gives a strong indication of an element of novelty. The advantage of an autoassociative approach is that all the inputs are being modelled simultaneously; giving a more general novelty detection capability than with the regression approach. Mappings obtained using autoassociative networks with two innermost hidden units and those obtained using a Kohonen map show extensive similarity. A disadvantage of the autoassociative approach is that these networks can be hard to train with local minima, in the non-linear case, seeming to be common.

3.4.3 Discussion of feed-forward methods

The above methods have the advantage that they are applicable in the instance where there is only a small number of data available or data are available for only one aircraft, and are hence suited for tracking down engine-specific fault modes. (Contrast the classification method described in the next section, where data representative of all the conditions of interest are required in order to obtain any answer). In the aircraft engine case, use can be made of the fact that there is more than one engine per aircraft and that all the engines are subject to the same environmental variables, the assumption being that the effect on the rest of the engines of any fault condition on one of the engines is negligible. (The latter is not necessarily a good assumption, as the pilot may well explicitly or implicitly alter the control settings in response to degradation of performance in one engine, and in any case the load on the other engines may well change as a result of an altered performance level.)

If one trains a network specifically for one aircraft, with the training set being the data from one engine, the fact that all the engines are subject to the same environment means that this "environment" or "context" is filtered out by the network and that we are effectively

working with far fewer parameters than it at first seems, that is, *the number of degrees of freedom of the underlying system is relatively small.* This should have some favourable consequences for the performance of neural network on this data.[3]

3.4.4 Classification approach

If enough labelled data is available (the word *enough* has to be quantified in some way - in particular sufficient fault data must be available), one can attempt to treat the monitoring task as a classification or diagnosis problem. One then has *classes* C_i corresponding to known conditions, and attempts to characterise a mapping from the input data to this set of classes using a neural net (or other classification or diagnosis method). An unknown quantity in the practical construction of such a network is the extent of training data necessary. In this study one is attempting to train the network on a data set which for some of the classes only contains a small number of examples. Use is made of the fact that the relationships in the data are strong and deterministic (relatively noise free) and there is a high level of redundancy in the data.

3.4.5 Figure of merit approach

Rather than classify engine condition according to discrete output classes, one could attempt to define a figure of merit such that this figure could be estimated by neural net regression from the data. In this case the output is one dimensional, and represents a general "health level" (or, looked at the other way, "alarm level"). Problems could arise if there is no suitable projection of the variation in the data onto one dimension, or in trying to treat sensor and data faults in an integrated way. Some representational difficulties can be avoided by representing the "normal" state using a number which is not at the extreme of the range. Alternatively, one could allow further "figures of merit" so that the result was not constrained to be one-dimensional. Effectively the classification approach does this, as the intermediate probability values can be read as figures of merit.

---

[3] The fact that there are relatively few degrees of freedom but that there is a large number of input variables also suggests that a *latent variable* approach may be fruitful.

3.5 Self-organising map

The self-organising feature map of Kohonen (1982) was introduced briefly in section 1.2. This algorithm is worthy of mention here in that is has been the basis of a number of attempts at using neural network methods for condition monitoring, particularly in early studies (Witcomb, Skitt and Hewitt, 1989; Skitt & Witcomb (1990)).

Kohonen's self-organising map algorithm is well documented and many references can be found in Kohonen[1988] and other sources. The structure of the network consists of an input layer and a grid layer (usually, but not necessarily, taken as a square, two-dimensional grid), which can be regarded from a neural network point of view as consisting of processing elements each of which is fully connected to the input layer with a vector of weights. A winner-take-all rule operates on this layer, the winner being the element with the lowest distance (however that is defined in the implementation, usually Euclidean) between its weights and the input vector. Weights in a neighbourhood of the winning node are then adjusted to move in the direction of the input vector. As training progresses it is usual to reduce the size of the neighbourhood within which the updating occurs. The result of the training process, if successful, is to form a mapping from the (n-dimensional) space of the input vectors to a k-dimensional (usually 2-dimensional) space of the processing elements in the grid layer, with the interesting property that (with certain reservations, as described below) it is topography-preserving in the sense that points which are close together in the input space are mapped to points close to one another in the output space.

The grid layer in the SOM can also be regarded from a "vector quantisation" point of view as a set of *prototypes* or *codebook* vectors. Alternatively, Ripley (1996) for example, views the self-organising map as "just a specific type of clustering algorithm". That is, the prototypical cases which are represented by the grid vectors can be considered as cluster centres.

3.5.1 Drawbacks of SOMs

In this section we enumerate some of the practical and theoretical deficiencies of the SOM.

Lack of "meaning"

The axes of the 2-dimensional Kohonen grid do not usually correspond to any identifiable real-world quantities. The fact that the axes do not correspond to easily visualised quantities is also a problem in techniques such as principal components analysis, although in the latter they are defined as linear combinations of the original dimensions. However in the SOM, there is no fixed definition which can be assigned to the dimensions of the grid. Sometimes it is the case that meaningful dimensions do occur, but this is the exception rather than the rule. In some applications where labelled data is available it is possible to extend the SOM idea to cover the inclusion of labelling or prior knowledge; these extensions move the concept of the SOM more towards that of supervised learning. See section 3.5.2.

Order dependence

The "mapping" obtained by training the network can be dependent on the order in which the training examples are presented, thus making consistency difficult to achieve. Ripley(1996) (among others) points out that "the result will depend on the random initialisation, the order of presentation of the examples and the tuning of the constants".

Twists and folds

In almost all of the applications described in the literature, the key quantity is the relative position of the mapped points. Problems can arise because of topological "twists" and "folds" in the mapping. In certain applications there are of necessity discontinuities in the mapping, and unless these are understood, they could be a source of misleading results.

Interpretation issues : artifacts, subjectivity and ambiguity

In addition to the mathematical weaknesses outlined above, there is a more subtle danger with the SOM arising from human factors, and the fact that interpretation of a trained Kohonen map relies heavily on subjective visual inspection. The danger is that the visual effect of the grid is very "appealing". The human mind will attempt to assign meaning and impose pattern where such things may have no statistical significance. That is to say, there is a danger of difficulty in separating artifacts of interpretation from true structure in the data.

The interpretation issue outlined above is a particular concern in the case where dependability is sought. In such cases there is a principle and a requirement that the interpretation of the output of the system be *unambiguous*. With a network such as Kohonen's, the output is ambiguous in the sense that it could be interpreted differently by different users or the same user under different conditions. This concern is real but at first sight appears somewhat paradoxical in the light of the usual criticism of neural network techniques as being "black boxes" and open to too little analysis. However it can be seen that both objections can be traced to the desire for an explanation of the "reasoning" by which an answer was obtained. This criticism can be overcome by adding further processing of the output which attempts to resolve the ambiguity by means of thresholds, rules or statistical analysis.

Deficiencies in the theory

The history of SOMs has seen practical applications of the technique flourish despite reservations concerning the theoretical underpinning of the method. Analyses of the SOM have been attempted by various authors on a number of levels.

Cottrell, Fort and Pages (1994) describe the Kohonen algorithm as "surprisingly resistant to a complete mathematical study", meaning that the self-organisation property is hard to prove (certainly for grid dimensionality greater than one; specifically there is difficulty in defining a potential function and finding absorbing sets so as to prove convergence). They note that the state of a Kohonen network is a special case of a stochastic process or Markov chain, and suggest considering the Kohonen algorithm as a Robbins-Monro algorithm (Robbins & Monro, 1951).

Furthermore, Luttrell (1989, 1994) points out that the SOM does not minimise an error function: "the Euclidean error function is not minimised when the nearest neighbour prescription is used. Further studies have shown that exact minimisation requires that the nearest neighbour prescription be replaced by the minimum distortion prescription in which the choice of winner is influenced by the reference vectors to which it is connected by the SOM neighbourhood function" (Luttrell, 1994).

Luttrell has worked extensively on the theoretical underpinning of self-organising maps and relates them to the Kalman filter, to a Bayesian formulation and to folded Markov chains. (Luttrell, 1993). The principal result contained in his paper is a Bayesian derivation of the properties of SOMs.

Noise sensitivity

Care is required when using the Kohonen map with noisy data in the sense that there is no easily drawn distinction between templates which represent noise and those which represent a statistically significant departure from normal conditions, unless the templates are explicitly labelled according to domain knowledge.

3.5.2 Extensions of the SOM concept

Many extensions of the basic SOM concept have been formulated. Often the motivation for these has been to make use of partially known structure or partial labelling in the data, or to provide a means of validating and analysing the clustering performed by the self-organising map.

Array of SOMs

Some workers have used an array of SOMs each trained on a different set of labelled data. The recall process is then to run the unseen data through each of the SOMs and measure which gives the lowest distance. This method has been used to impart a partial "explanation" property to the SOM. Nevertheless it should be noted that this explanation ability is relative to the labelling of the data which has to be performed by hand or by some other method.

Hierarchical SOMs

In the hierarchical SOM described by Luttrell, (Luttrell, 1989b) the input space is split by the network into a number of lower dimensional subspaces and a SOM trained on each subspace. "Simultaneously another SOM is trained on the outputs of the above SOMs to produce the final network output. This type of multi-layer SOM can be refined by introducing an error function which measures the average Euclidean error between the input vector and a vector that is constructed from the final network output. This forces the optimisation of the layers to be tied together in such a way that they have to be trained co-operatively.. by sending backpropagation signals from the second layer to the first layer.

However these backpropagation signals are not derived from an external supervisor so this type of training algorithm is called '*self-supervised*'" (Luttrell, 1994)

Labelled Kohonen maps

An elegant extension of the mode of use of a SOM is obtained by including target or label data in the *input* layer. In training this method operates exactly as for the conventional Kohonen map. In recall mode we do not have the label information. If the labels are binary (or the class labels form a sufficiently small set) then by recalling on all the admissible values of the label variable in turn, adjoined to the recall data, the self-organising map can be used as a "what-if" or exploratory tool. Analysis of the pattern of connections (the *fan-out*) from the relevant input node can also yield useful insights into multivariate associations in the data. (Pilkington, 1995).

Additional layers

Another method of extending the SOM is to add a supervised-learning layer (or layers) with connections from either the whole grid or from units which represent the x and y co-ordinates of the winning cell. Backpropagation, or in favourable cases a linear optimisation method, can be used to set the weights in the additional layer. This approach has limited use from a regression point of view in that some accuracy is inevitably lost by the process of quantising the data on the Kohonen grid. One can, however, view the grid as a *feature detector*, thereby possibly providing some further "explanation" of a network's decision than is evident from a straightforward supervised learning network.

3.5.3 Self-organising maps as a monitoring tool

Given adequate, representative data and a suitable grid size, the algorithm should ensure that the codebook vectors span the space of variation that is expected within the data. Then the distance between the current input and the nearest of the codebook vectors can be used as a measure of novelty. A reduction of dimensionality is achieved and good visualisation ensues once the areas of the grid have been labelled according to some calibration data.

The key points in the above paragraph are "adequate, representative data" and a suitable setting of the model parameters, and this statement of the method begs the question of by what methodology this is achieved systematically in practice.

3.5.4 An alternative

Recently Bishop *et al* (1996) have proposed the Generative Topographic Map (GTM), which is based on the concepts of Gaussian mixture models (Titterington, *et al.*, 1985; McLachlan & Basford, 1988) and is trained by the Expectation - Maximisation (EM) algorithm (Dempster, Laird and Rubin, 1977. The GTM is a *generative* model, that is, it models a probability density and overcomes many of the problems of the SOM. Bishop *et al.* claim that the algorithms used for training the GTM require less hand-crafting than Kohonen's SOM.

3.5.5 Discussion

Although Kohonen's SOM is useful as an exploratory data analysis tool and as the basis for representing a state space in two dimensions, it will usually be found that more focussed methods are required for condition monitoring problems with operational data, often in combination with a Kohonen network or similar algorithm. In the next chapter, classification and regression applications of neural networks for condition monitoring are explored, and in Chapter 5, a context-based novelty detection method is introduced in which two Kohonen networks are trained on different subsets of the input, in order to provide a context dependent measure of novelty.

4 Monitoring and diagnosis of a specific engine component

4.1 Introduction

The objective in this Chapter is to assess the feasibility of remotely inferring the condition of the combustor module in the interior of the engine by remote sensing from readings taken from engine sensors which are further downstream in the gas path.

Combustors in commercial jet aircraft engines are subject to frequent, regular, visual inspections using a piece of equipment known as a *boroscope*[1], so information on any deterioration in condition in this area is captured at an early stage.

4.2 Data sources and methods

4.2.1 Data sources

Routine data from engine sensors is recorded in service and stored in the Quick Access Recorder (QAR). Engine data for this application is available in two forms. A time series of raw data can be read from the QAR, with records taken every second. In this study, data is captured from when the throttles are engaged up to an altitude of about 3000 feet. Data is also available from the output of the condition monitoring system, COMPASS. [2] This system determines for each flight a "stable cruise point", in which a snapshot of data is taken. COMPASS has quite extensive logic and statistical processing which determines the point of optimal stability at which to extract this reading. In developing an approach to this kind of monitoring, several ways of applying neural networks have been considered, resulting in the selection of two complementary views: a prediction setup which looks at the continuous-time QAR data and a labelled classification approach for dealing with the snapshot data from COMPASS, recorded in the 'cruise' phase of flight.

---

[1] A boroscope is a fibre optic probe which enables the engineer to carry out visual inspection of an otherwise inaccessible area of an engine.

[2] COMPASS was developed by Rolls Royce and EDS-Scicon.

For the Quick Access Recorder (QAR) data, the following fields are available:

SAT	Static Air Temperature
PALT	Pressure Altitude
CAS	Computed Air Speed
P2	Ambient Air Pressure
TRA	Throttle Resolver Angle (degrees)
EPR	Engine Pressure Ratio
N1	Low Pressure Spool Speed
N2	Intermediate Pressure Spool Speed
N3	High Pressure Spool Speed
FF	Fuel Flow
TGT	Turbine Gas Temperature
EGT (or T50)	Exhaust Gas Temperature

"COMPASS" data, which has been processed by the condition monitoring system, includes derived parameters such as differences ("deltas") from model values. Smoothed values of these deltas are also available.

ALT	Altitude
MN	Mach number
T2A	Ambient air temperature (adjusted)
EPRA	Engine Pressure Ratio (adjusted)
N1	Low pressure spool speed
N2	Intermediate pressure spool speed
N3	High Pressure Spool speed
FF	Fuel flow
PINT	Intermediate Pressure
P2	Ambient Air Pressure
EPR	Engine Pressure ratio
T2	Ambient air temperature
TRA	Throttle resolver angle
P300	HP Compressor Exit Pressure
WBIPA	Air Conditioning Bleed from IP Compressor
GLE	Generator load
DGLE	Delta GLE
DP25	Delta IP Compressor Outlet Pressure
DT25	Delta IP Compressor Outlet Temperature
DT3	Delta HP Compressor Outlet Temperature
DN1	Delta N1
DN2	Delta N2
DN3	Delta N3
DP160	Delta Fan Exit Pressure
DFF	Delta Fuel Flow
DTGT_L	Smoothed value of delta TGT
DFF_L	Smoothed value of delta Fuel Flow

DN1_L	Smoothed value of delta N1
DN2_L	Smoothed value of delta N2
DN3_L	Smoothed value of delta N3
DTGT	Delta TGT
DEGT	Delta EGT
TGT	Turbine Gas Temperature

Data sets available.

One set of QAR data was available for an aircraft with an engine which had been found to fail the boroscope inspection. A further thirteen sets of take-off QAR data for fault-free engines were available, together with further, unlabelled data.

COMPASS data was available for a wider selection of aircraft (25 in all) including one with an engine which had suffered a fault of this type, and three which had failed the visual (boroscope) inspection.

4.2.2 Methods

The preliminary approach was to couch the model in terms of a *regression* or estimation setup, with the EGT and TGT variables as dependent variables (outputs) and the rest of the engine parameters as inputs. The network was trained on data from one of the engines on a given aircraft. Data for the other engines was passed through the trained network. The discrepancy between the actual engine TGT the neural network's prediction was measured for each engine.

This method is advantageous were there is relatively little data and shows immediately any deviation from the nominal condition. The *noise*, *i.e.* random variation, in the engine data is small and the neural network is able to form an accurate model of the relationships in the data given only a relatively small amount of data. The danger of *overfitting* should be noted however in such a case.

As further data sets became available it was possible to extend the neural net analysis and some alternative approaches were considered, as follows:

(a) Autoassociative networks

A multi-layer perceptron (MLP) network was trained to reproduce the input data and the reconstruction error was examined, to give an indication of "novelty". Some information can also be obtained by examining activation levels of the hidden units. This method has two drawbacks. One is that it is susceptible to local optima in training; we found autoassociative backpropagation networks difficult to train adequately to convergence. If the network dos not converge on the training data, it is then unclear whether high reconstruction error is due to novelty in the data or to an inadequate model. Secondly, the interpretation of activity in the middle layer is not straightforward and can be misleading.

(b) Classification setup of a back-propagation network.

This approach is described in section 4.3.

(c) Figure-of-merit approach.

This is similar to the classification approach except that instead of assigning distinct classes to the various condition descriptions, they are scored using a simple integer scoring scheme.

4.3 Classification study

4.3.1 Description

The neural network setup is as follows: a standard backpropagation network was used, with initially 41 inputs and 4 outputs, the outputs representing categories relating to engine condition with respect to the fault mode under consideration. The categories were: "new", "OK" , "boroscope inspection failure" (abbreviated to "boroscope") and "fault" condition.

The network was trained using sets of output data from the condition monitoring system COMPASS taken at a snapshot point during the cruise phase for each flight. Training data were collected on each of the four engines from each of six aircraft, selected to give a variety of conditions from some brand new engines to ones which had been removed following an inspection, and one which had developed a full instance of the fault mode.

First attempts at modelling the fault condition in this way were confounded by the presence of sensor faults (see section 4.3.2). After removing six of the more unreliable input variables and using a neural network (MLP trained using backpropagation) to impute missing values of the variable P3, which was highly susceptible to sensor faults, a consistent classification was achieved which could be tested on unseen data.

4.3.2 Treatment of sensor faults

An early stage of the analysis consisted in identifying, with the help of neural network runs, sensor faults in the data. A number of sensor fault modes were found in the course of the analysis. Examples of sensor fault modes with different properties are as follows:

(a) Variable Inlet Guide Vane readings (VIGV and dVIGV).

These parameters were found to be missing in about half the data sets, and would be difficult to infer from other variables. It was therefore decided to leave this variable out of the analysis. In practice, poor VIGV scheduling affects N2 (Intermediate pressure spool speed) and is tolerated as part of the normal variation in the system.

(b) Pressure readings P3 (or P300) & P3DEL

These were found to be missing in about 20% of cases. It is possible to predict P3 from a subset of the other input variables (Altitude, Mach number and EPR) with reasonably high accuracy (RMSE=1.2% in training data; 1.6% in test data). However P3DEL was not predictable from these variables (results gave an average of 4 with very little variation (+/- 0.05) where the true values lay between 2.7 and 4.6).

(c) DT3 (*delta* T3, deviation from book value for temperature reading T3).

This case illustrates a difficulty in this type of condition monitoring work, in that the only occurrence of this sensor fault coincides with the only serious instance in the data of the fault mode under study. One option would be, as only one engine out of four is affected, to take the average of the other three. However, values for this parameter vary greatly between the other three engines. The alternative options are to impute the missing data from the other input variables, or to ignore the variable. It was decided to impute the missing data in this case although the sensor fault data is known to be non-typical.

After pre-processing as described, to remove the effects of sensor faults, the training data consisted of 35 input variables and the 4 condition outputs.

4.3.3 Networks used

Network "A": 35 inputs, no hidden layer, 4 outputs.
Network "B": 132 inputs (transforms of 35 original inputs); no hidden layer; 4 outputs.
Network "C": 35 inputs, 5 hidden units, 4 outputs.

The options for Networks A and B were set up in NeuralWorks *Predict* as given in table 4.1:

Network ID	prediction or classification setting	Network structure	noise setting	transformation extent setting	variable selection extent setting	network search extent setting
A	classification	35-0-4	clean	superficial	none	moderate
B	classification	132-0-4	clean	comprehensive	none	comprehensive

Table 4.1: details of networks A and B set up in "NeuralWorks Predict"

Network "C" was set up in *NeuralWorks Professional II*, with four hidden units, full connection between the inputs and the hidden layer and between the hidden layer and the output layer, but no connections between the inputs and the output layer. The activation function in each layer was *tanh,* and the learning algorithm was set to *delta rule* with normalisation and an "epoch" size of 4 records between weight updates.

4.3.4 Results on training data

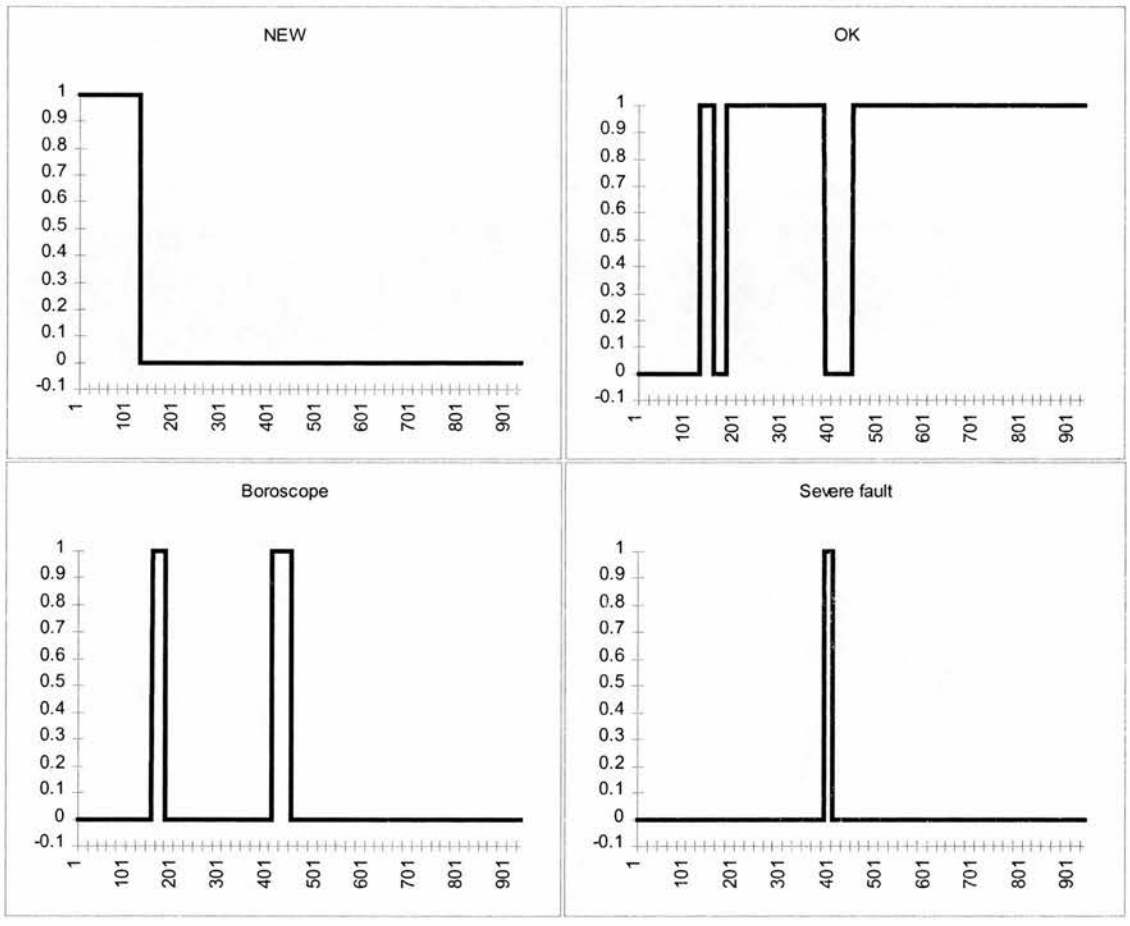


Figure 4.1: Training data used for classification networks.

The graph shows the target values in each of the four classes.

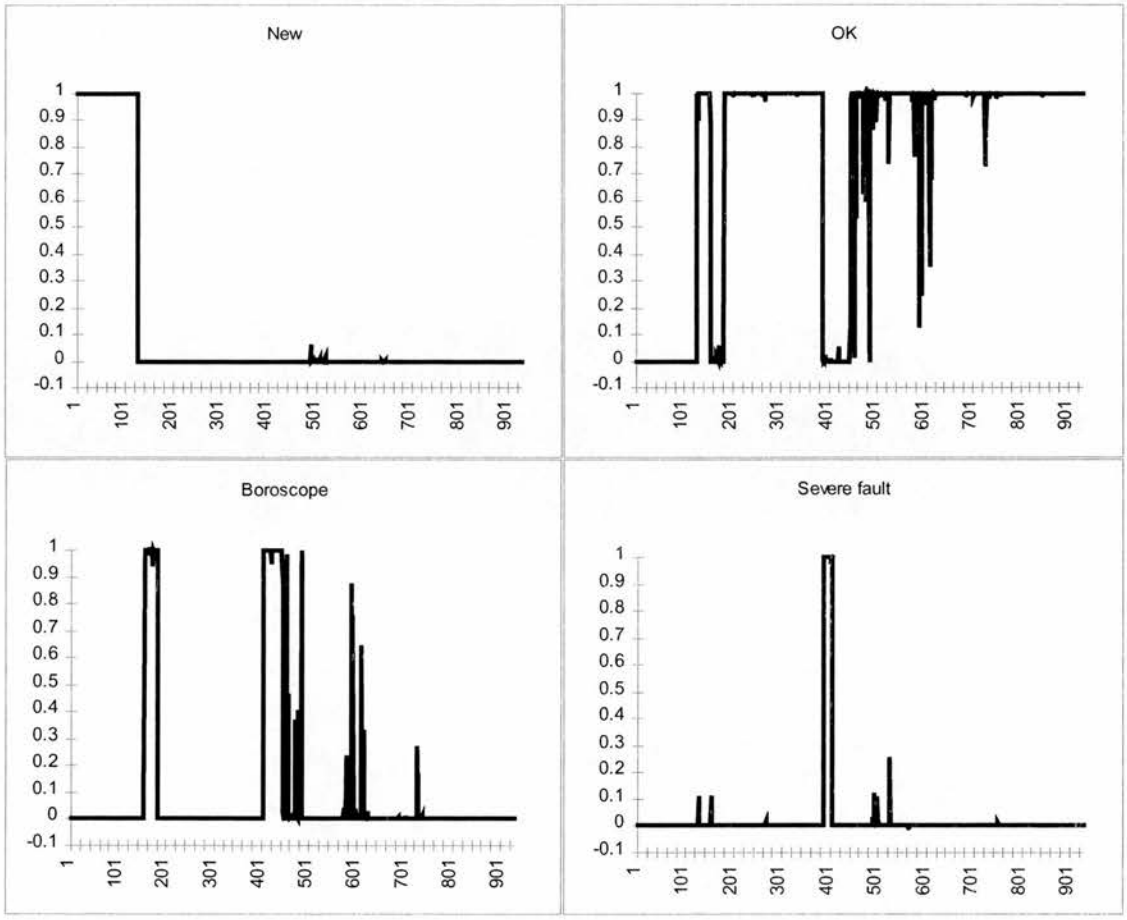


Figure 4.2 Results of network 'A' recalled on the training data.

Figure 4.1 shows the training data itself. The training data consists of cruise snapshot sequences for six aircraft. These sequences have been concatenated together into one series. Figure 4.2 shows the outputs on the training data for network (A) as an example. A small amount of overlap of classes is evident at around records 480 and 580-620.

4.3.5 Results on Test Data

Test data was gathered for 23 aircraft (that is, 92 engines) for the four month period between 09Dec95 and 04Apr96. The network classified most of the data into the "OK" category (class 2); in a few cases some probability mass was assigned to both classes 1 and 3 or occasionally 1,2 and 3. These combinations indicate novel situations not encountered in the training data. Usually this will require retraining of the network, but cases should be scrutinised for the presence of fault modes which had not been encountered in the training data.

The following graphs show a selection from the results. Data shown are moving averages of the network outputs taken over ten flights. Figures 4.3, 4.4 and 4.5 show the effect of an engine "running-in" over a period of time. The value on the "new engine" category decreases gradually towards zero as the value on the "OK" category increases gradually towards unity. The changes do not show on the graphs as monotonic because there is some "noise", *i.e.* random variation, probably due to the fact that the input data is a discontinuous sequence of "snapshots" representing different flights, with correspondingly different environmental conditions. Figures 4.3 - 4.5 represent the results from three different "networks";

Figure 4.3:	Network "A":	35 inputs, no hidden layer, 4 outputs.
Figure 4.4:	Network "B":	132 inputs (transforms of 35 original inputs); no hidden layer; 4 outputs.
Figure 4.5:	Network "C":	35 inputs, 5 hidden units, 4 outputs.

There is some variation among the models in the values of the network outputs. The networks agree on the extreme values (0 or 1) for each of the classes, but the scaling of the intermediate values is seen to vary according to the network model chosen. With the intermediate values of the network outputs it is difficult to assign a quantitative *a priori* meaning to a particular output value. In this respect the models need to be calibrated by a lengthy process of data gathering, labelling and validation. Without this calibration, it is difficult on this data set to construct an evaluation function according to which to compare the results of the different networks. In the absence of this, there is no reason to prefer a particular one of the three architectures discussed. Hence at this stage the model outputs can only be used qualitatively.

Figure 4.6 shows an example of a trend developing in the "boroscope" category, over a number of weeks.

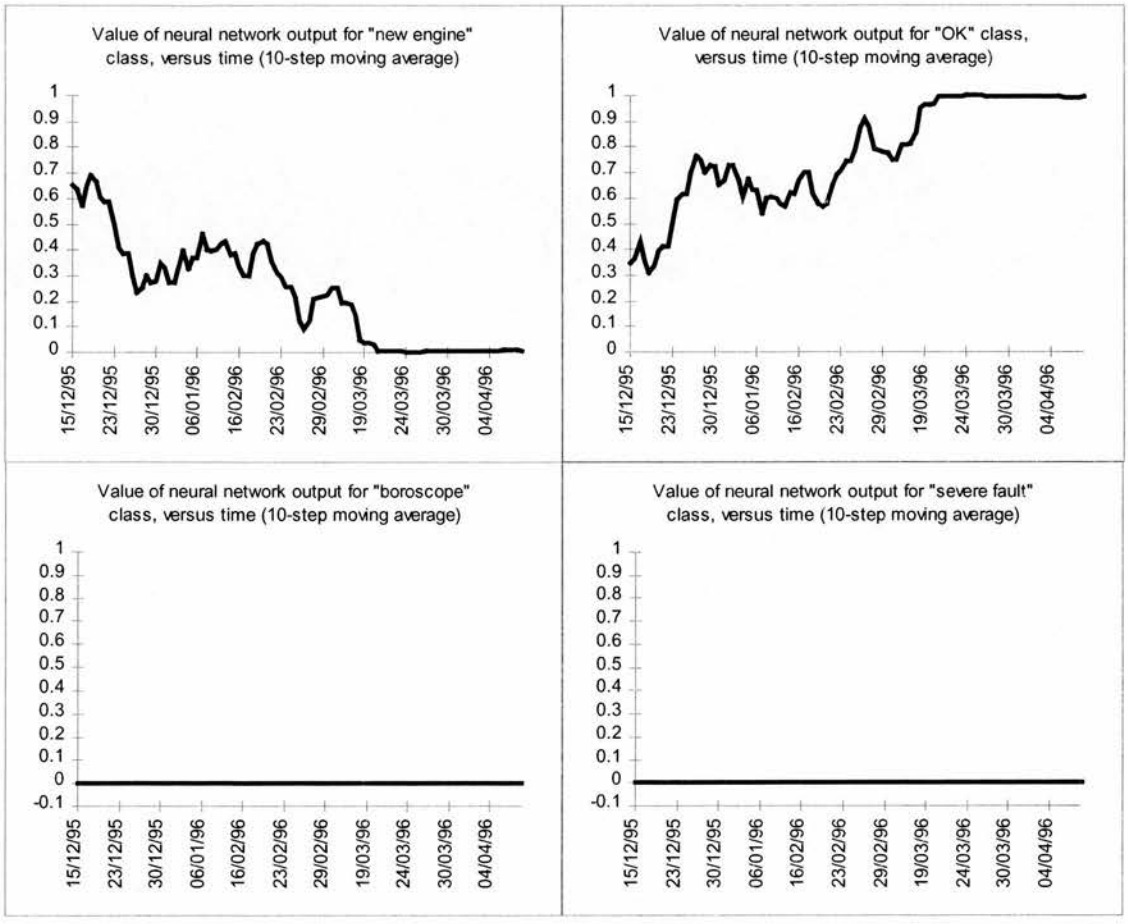


Figure 4.3 Engine serial number 13235 (fitted 10/11/95): results from network A

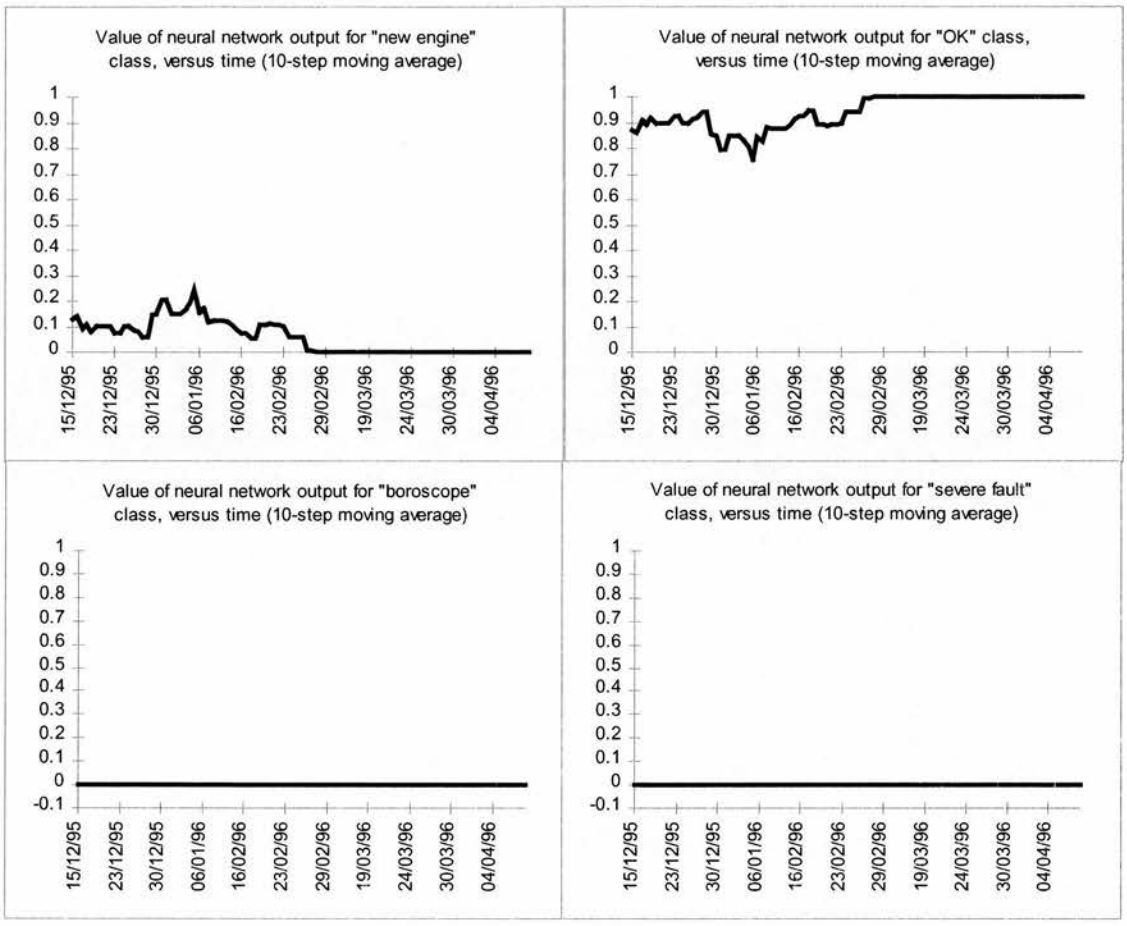


Figure 4.4 Engine serial number 13235 (fitted 10/11/95): results from Network B

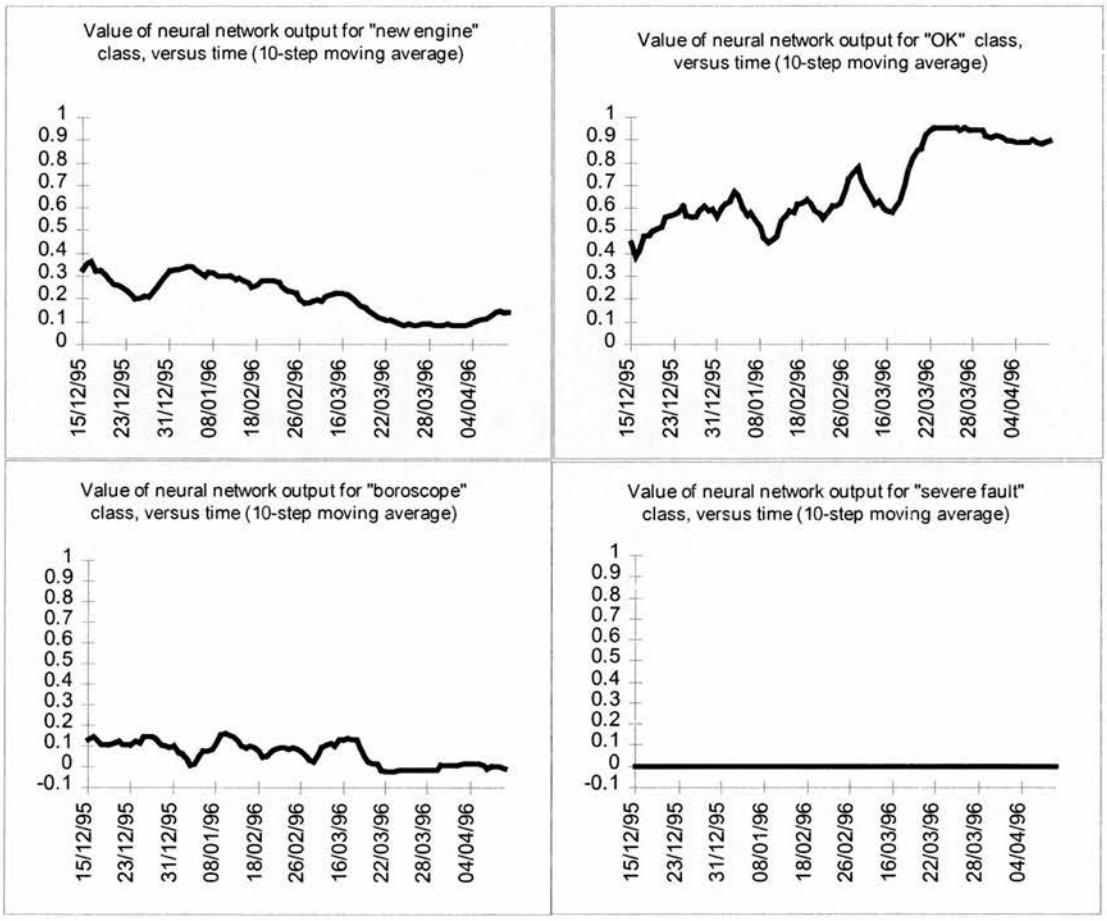


Figure 4.5 Engine serial number 13235 (fitted 10/11/95): results from network 'C'

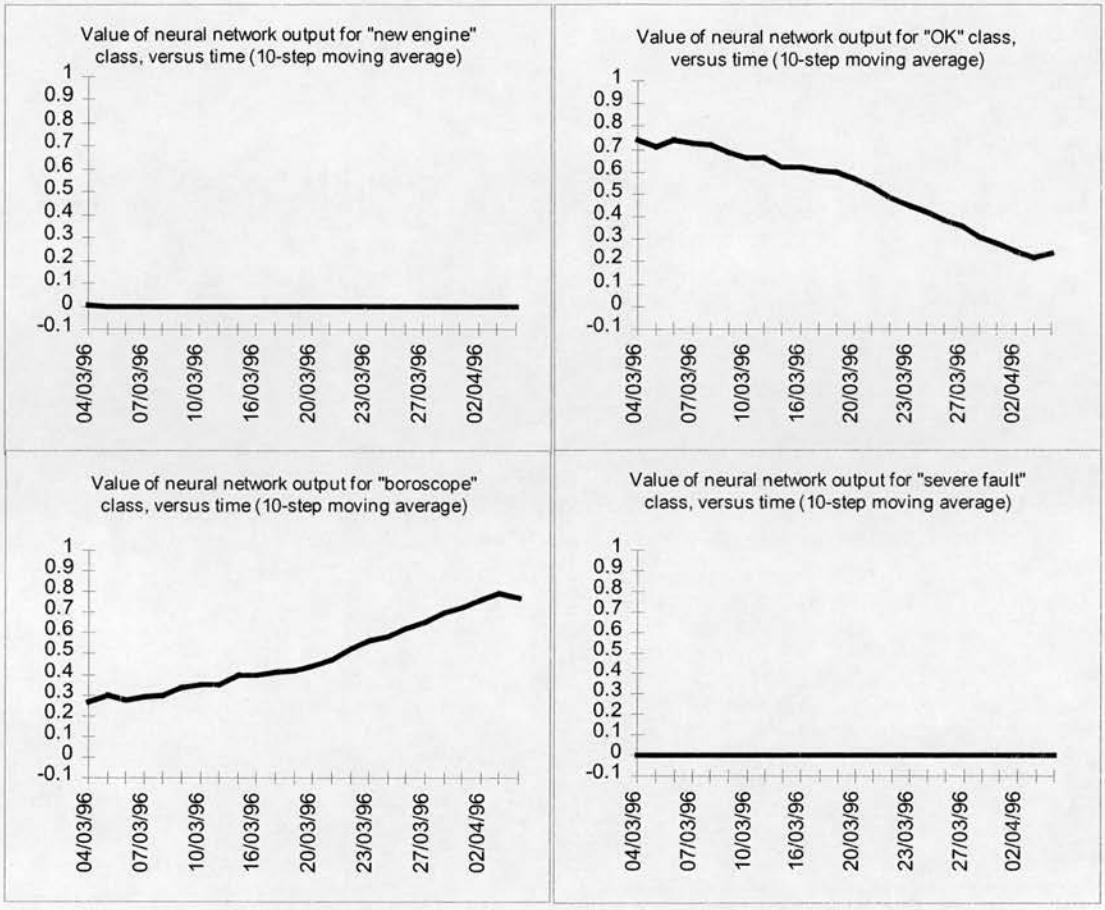


Figure 4.6: engine serial number 13093 (network 'C'): example of increasing trend in "boroscope" category.

4.3.6 Discussion

When calibrated over a test set of around 90 engines over a period of around four months, plus some subsequent data sets which were saved following boroscope inspection failures, the classification method appears overall to be overestimating the fault rate. The probable reason for this is that, in the encoding of the target data in the training set, each record is labelled with a "1" in the "boroscope failure" or "fault" class if the sequence of data for that engine results ultimately in a boroscope failure or fault. Thus, early in such a data sequence, prior to the development of the fault, it is likely that non-faulty conditions are being "mis-labelled" as faulty. In addition, there are some fault instances where the neural network classifies the data as normal. Therefore although this method cannot be used on its own as a reliable predictor of incipient degradation of combustor performance, it can serve as part of a suite of approaches including physical inspection and expert analysis of the data.

4.4 Regression study

This section describes an alternative approach to condition monitoring, using data from the quick access recorder (QAR), logged at one-second intervals in take-off. In the regression framework (as introduced in section 3.4.1), the neural network is used to model one of the engine parameters as a function of a suitable set of upstream engine parameter readings and environmental data. In the case of this study, turbine gas temperature (TGT) and exhaust gas temperature (EGT, otherwise known as "T50") were chosen as the dependent variables.

4.4.1 Data

For the QAR data, the following fields are available:

SAT	Static Air Temperature
PALT	Pressure Altitude
CAS	Computed Air Speed
P2	Ambient Air Pressure
TRA	Throttle Resolver Angle (degrees)
EPR	Engine Pressure Ratio
N1	Low Pressure Spool Speed
N2	Intermediate Pressure Spool Speed
N3	High Pressure Spool Speed
FF	Fuel Flow
TGT	Turbine Gas Temperature
EGT (or T50)	Exhaust Gas Temperature

Data were collected every second for each of a sequence of take-offs, starting with the aircraft on the runway about to accelerate, and ending at about 3000 ft. altitude above the airfield. Take-off sequences were recorded for eight different aircraft (each aircraft having four engines, 1300 records per engine). Additionally, data for a sequence of five take-offs for one particular aircraft (850 records), and a sequence of thirteen take-offs for a further aircraft (2175 records) were available. How these data were partitioned into training and test sets is described in section 4.4.2.

Network structure and parameters

A back-propagation network was used with 11 inputs and one output. In the cases reported here, one hidden layer was used, containing 5 hidden units using the *tanh* activation function. The output units used a linear activation function. The values for Turbine Gas Temperature (TGT) and T50 were predicted in two separate exercises, each using an 11-5-1 network.

4.4.2 Training strategies and results

There are three modes in which the training data can be used:

(a) train on data for one engine

(b) train on data for all four engines

(c) train on a sequence of takeoffs for the same aircraft.

(a) The network is trained on data for one engine. Data from the other three engines is passed through the network as unseen data and any discrepancies between the results for the different engines is noted. This is equivalent to treating one engine as a reference model or template. A disadvantage of the method would be that false positives could be generated if the model were overfitted to one set of data.

Figure 4.7 shows the effect of training on a set of reference engines (in this case engine position 1 for a sequence of eight take-offs (one each for eight different aircraft), then recalling the network for the other three engines for each aircraft. In the training set, and in the graphs in Fig 4.7, the data for the eight take-offs have been concatenated together to form one set. each take-off is characterised by a "hump" as the turbine gas temperature rapidly rises as the throttles are opened, then stabilises, then falls off as the throttle setting is reduced after take-off. Figure 4.7 shows for each of the four engine positions the measured ("actual") turbine gas temperature and the network's estimate ("prediction") of TGT for that particular engine based on having been trained on the data for engine 1 for the same conditions.

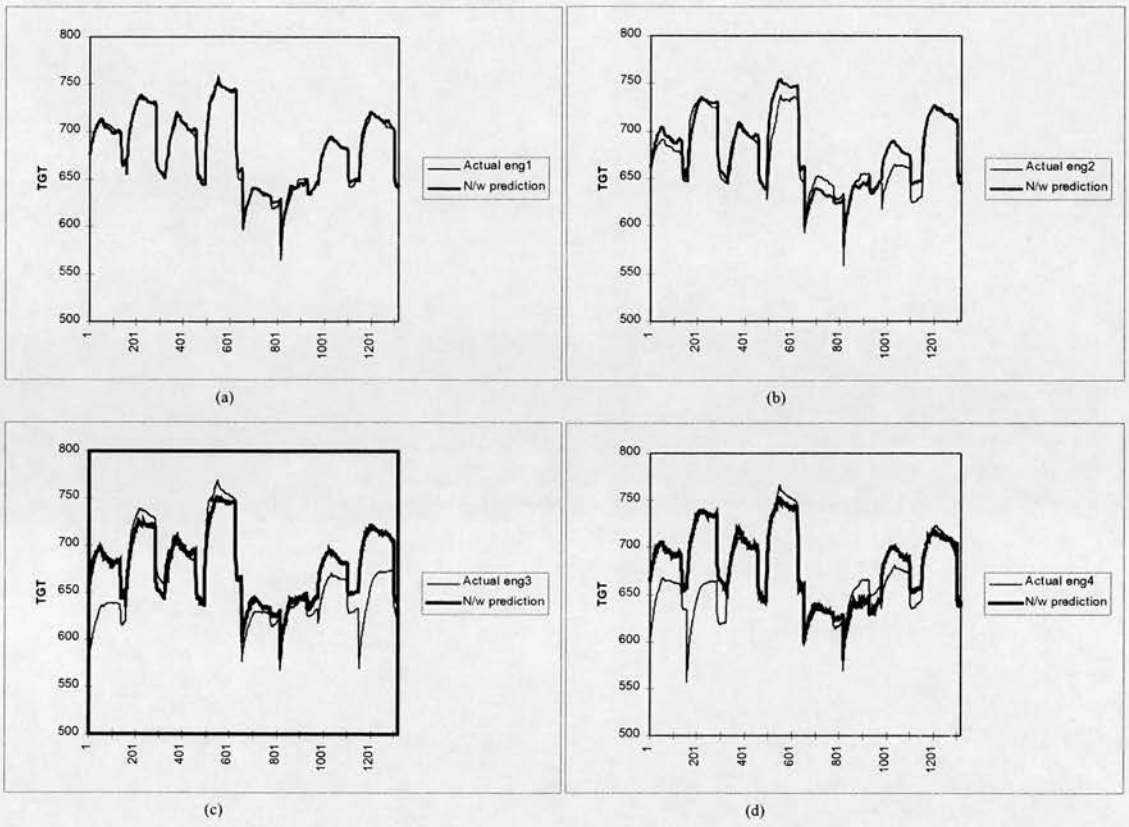


Figure 4.7: Turbine Gas Temperature prediction: network trained on position-1 engine and recalled on the same flights' data for the other engines. ("n/w prediction" stands for "network prediction").

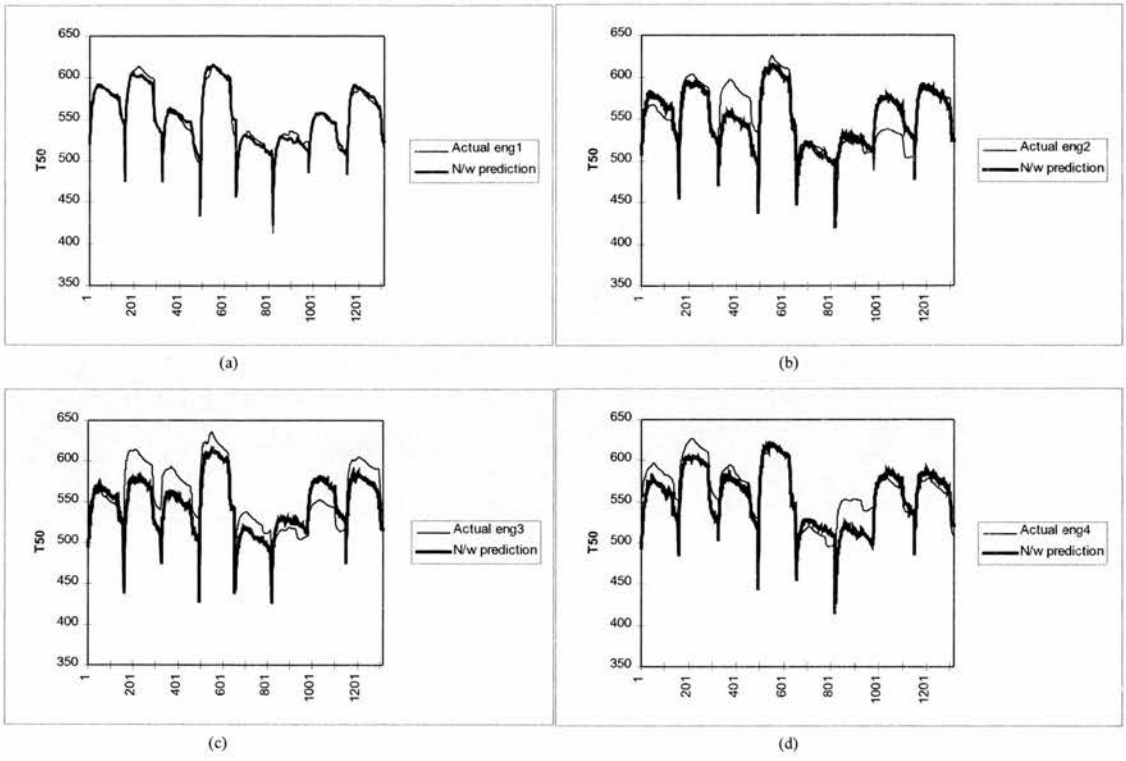


Figure 4.8: Exhaust Gas Temperature (T50) prediction: network trained on position-1 engine and recalled on the same flights' data for the other engines.

Figure 4.8 shows the results of the same analysis performed for the variable T50 (Exhaust Gas Temperature). It is believed that the presence of opposing discrepancies in TGT and T50 can give some indication of engine condition. Although Figs. 4.7 and 4.8 appear to show an overfitting effect where the network is performing well on the training data, the variation which is observed on the data for the position-2,3 and 4 engines on the same set of flights, is indicative of the normal variation between engines, and reflects known differences in efficiency.

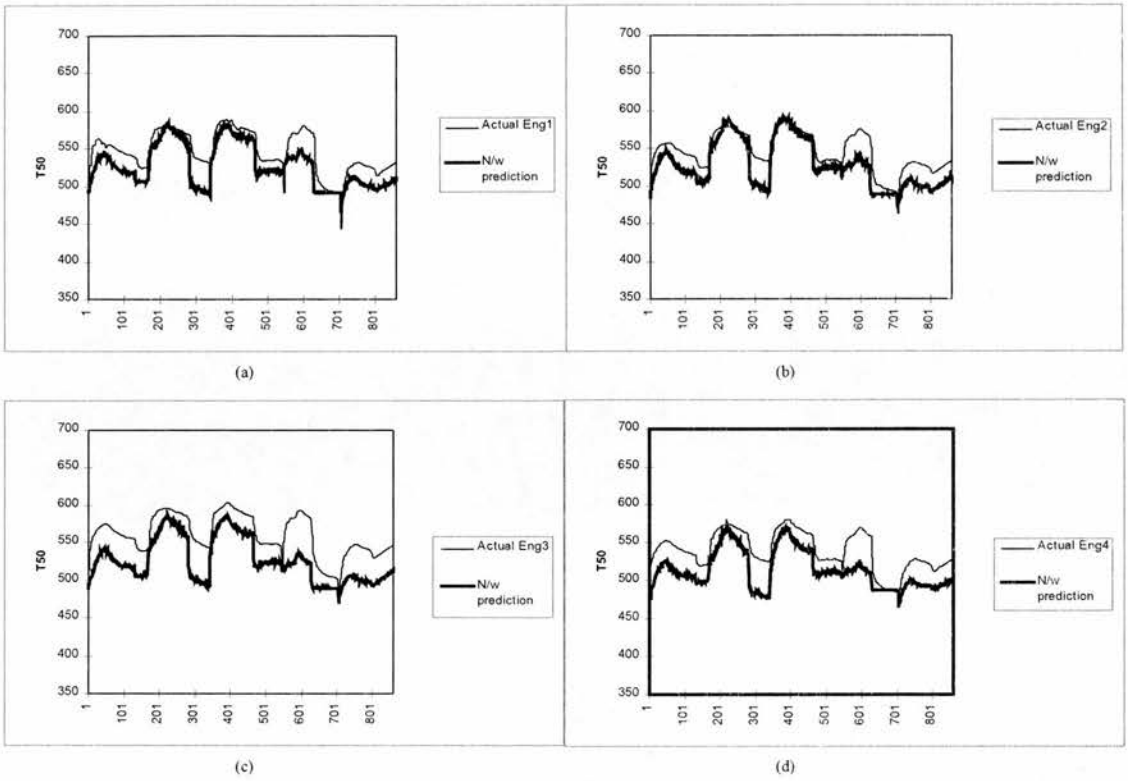


Figure 4.9 : T50 result for five concatenated take-off data sets for an aircraft that was not in the training set.

Figure 4.9 shows the results of recalling the network on a set of data from four engines from one aircraft which was not included in the training set, and here a consistent difference between the neural network results and the measured exhaust gas temperature is observed. This is probably a result of the fact that a small number of different aircraft were used in constructing the training set. See Figure 4.12.

(b) The second method of training is to use the data from all four engines in the training set, and examine the variation in the residuals for the training data. The advantages of this scheme are that it gives a larger training set and avoids biasing the network towards one reference engine. However this method may fail to detect a fault or trend, and thus produce a false negative result, if the training set is poorly chosen. Figures 4.10 and 4.11 show results for turbine gas temperature (TGT) and exhaust gas temperature (T50) prediction, shown for the training data. Figure 4.12 shows the results of the network trained using regime (b), and recalled on the same test set data as in Fig. 4.9. Note that the residuals are much smaller in Fig. 4.12, and that there is less bias, than in the case of regime (a) above.

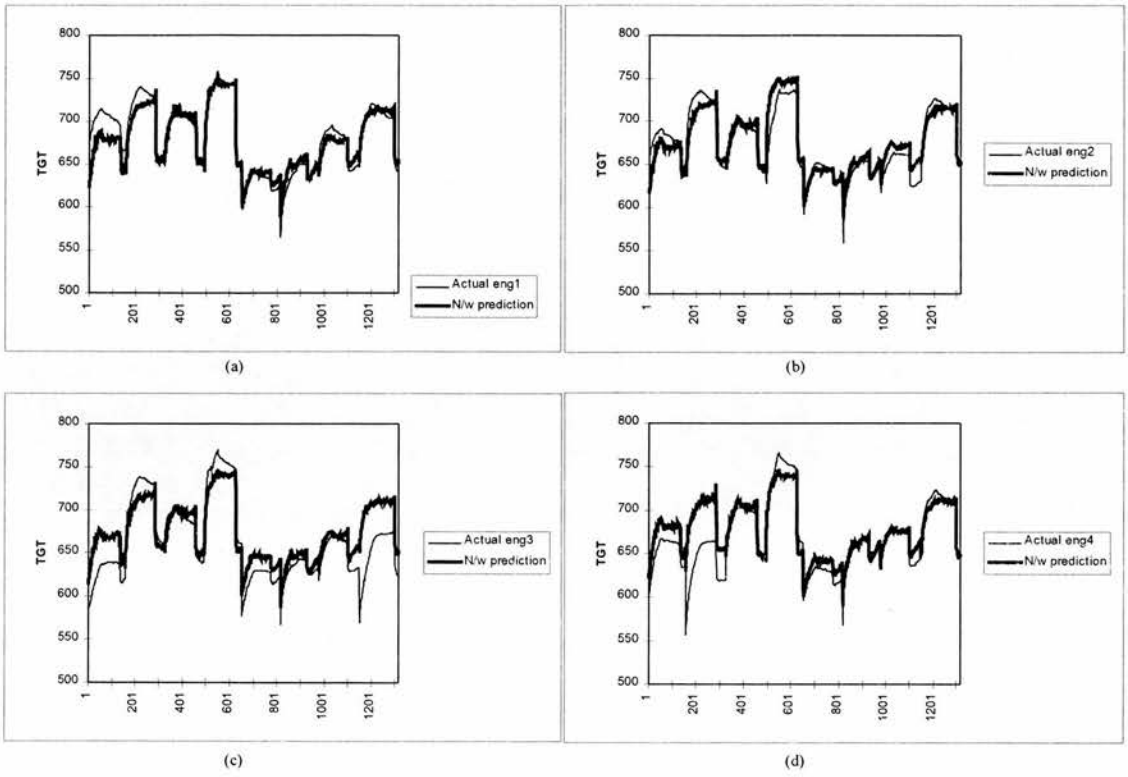


Figure 4.10: TGT prediction: network trained on all 4 engines' data; recalled on training set.

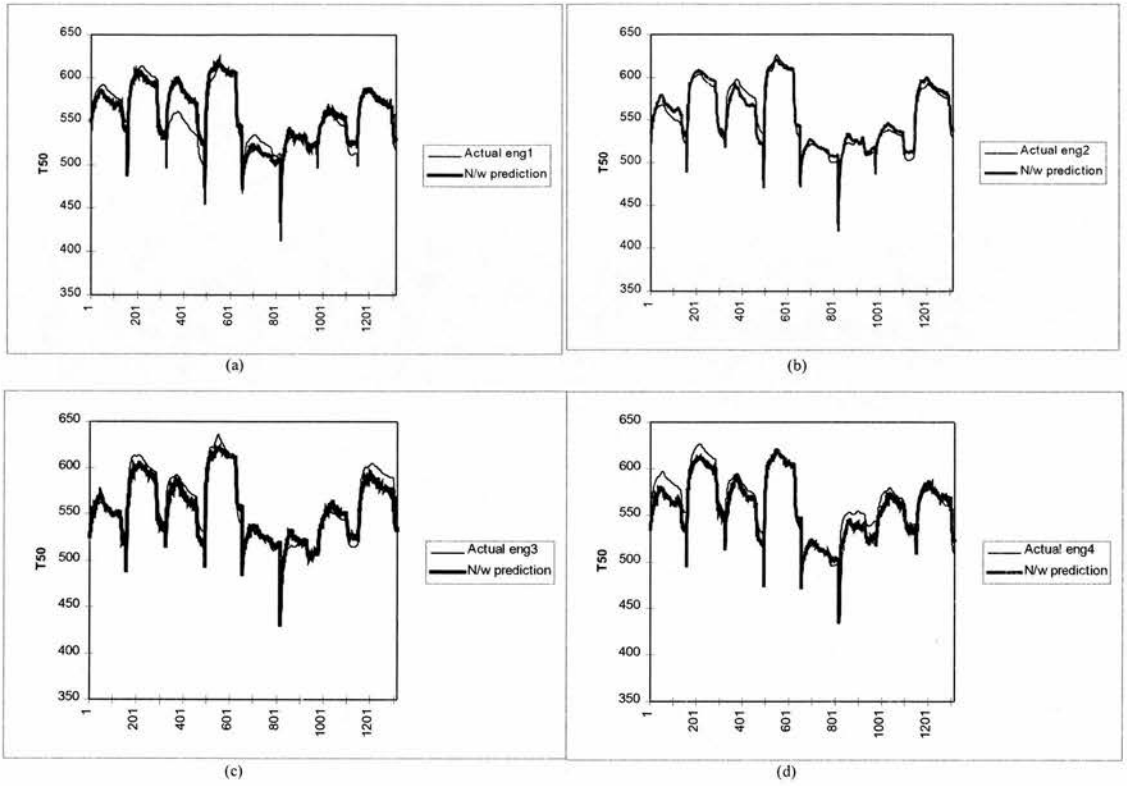


Figure 4.11 T50 prediction: network trained on all 4 engines' data; recalled on training set.

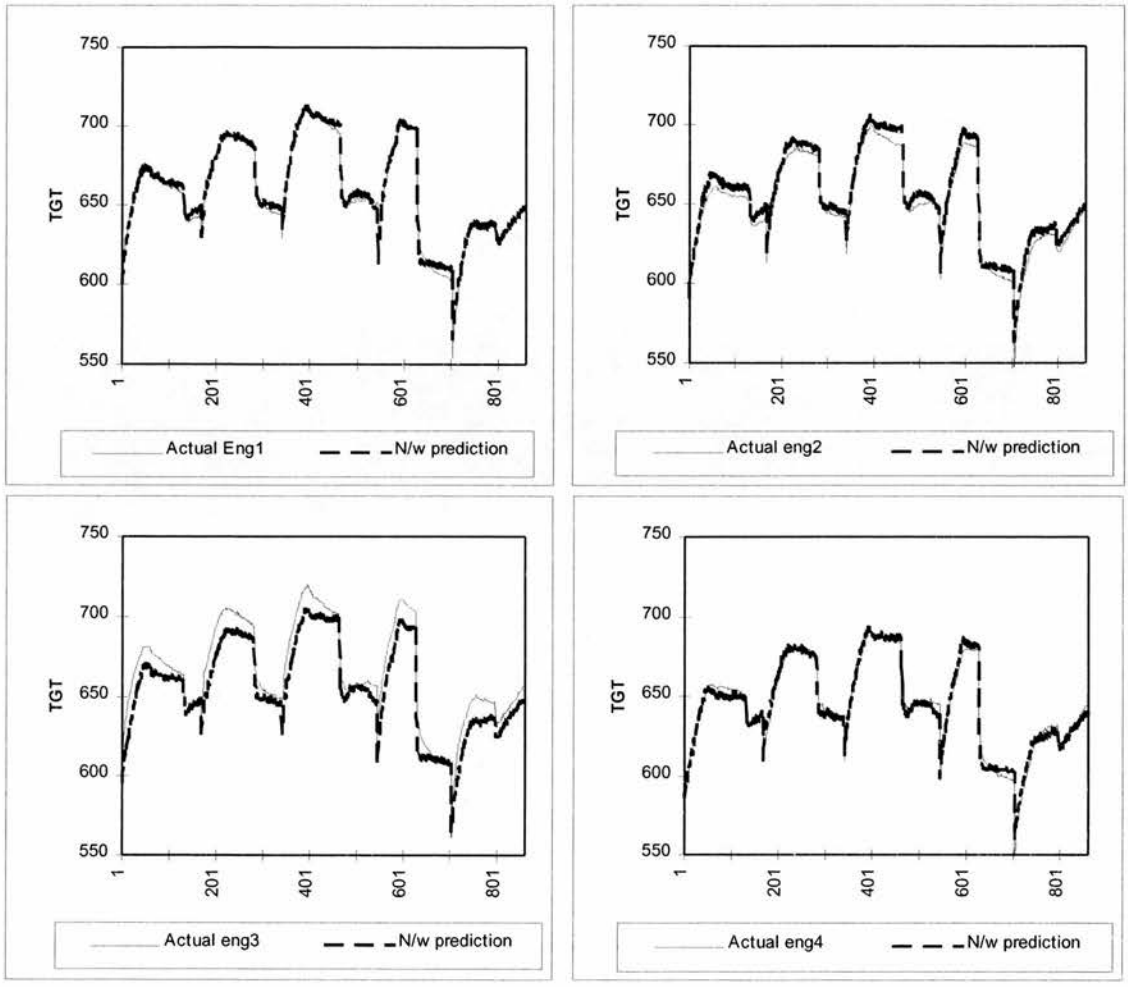


Figure 4.12: TGT Prediction: network trained on engine position-1 data and recalled for all four engines for 5 take-offs for aircraft 'O'.

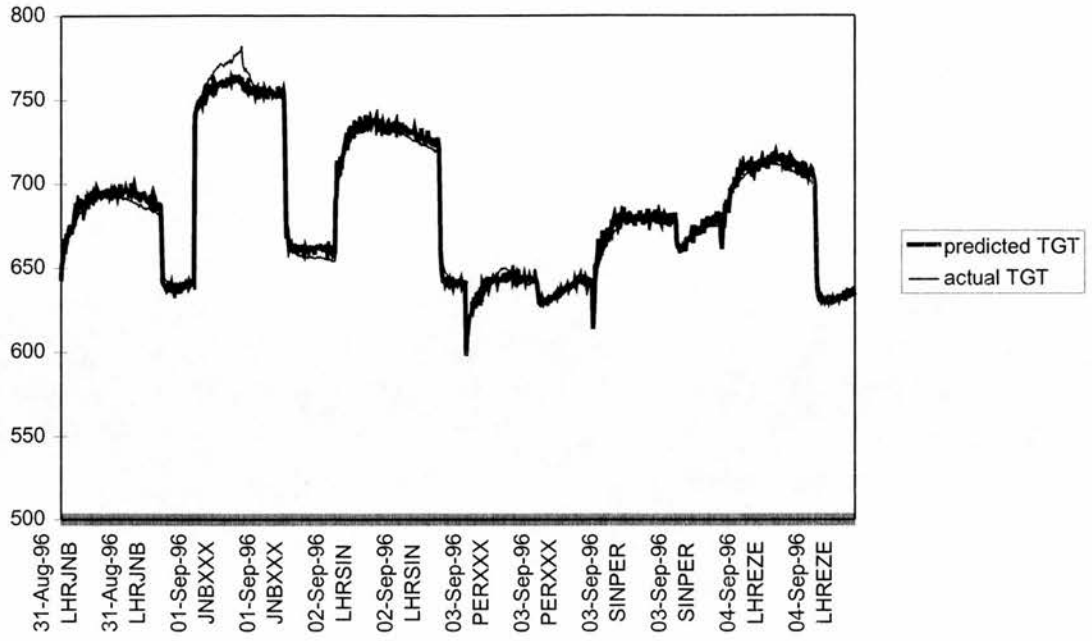


Figure 4.13: Network trained on seven take-offs and recalled for six subsequent take-offs for aircraft 'A', engine 1.

Figure 4.12 shows the effect of comparing the four engines on one aircraft, with the network trained on a variety of take-offs, for the same aircraft. Figure 4.13 shows results from recalling on unseen data, the network having been trained on all four engines for a sequence of take-offs for a particular aircraft. It can be seen that the accuracy of modelling is sufficient for slight variations in turbine gas temperature from the neural network model to be made apparent (see for example the second take-off in Figure 4.13).

4.4.3 Discussion

Using a neural network as a nonlinear regression model for this kind of data easily identifies portions of unseen data where there is a departure from the model which has been learnt from the training data. Visualisation is straightforward, as only one variable is considered at one time, and an x-y plot can clearly show the deviation.

The regression approach is useful particularly for four-engined aircraft (for example, 747-400 data) as for each engine under study there are three others in an identical environment, and a high modelling precision for individual engine parameters such as turbine gas temperature can be achieved. This method allows us to pick out differences in engine behaviour. (A more superficial type of monitoring can be implemented merely by comparing the true engine outputs, but would not take into account possible variation due to differences in upstream engine parameters and control parameters). It does not require data to be collected on a large sample of aircraft (unlike the classification approach) and so can be used to study uncommon fault modes.

Shortcomings of the regression method as follows. Firstly, although the residual gives a quantitative measure of deviation from a model, there is no indication of how applicable that model is in the context of the present data. Secondly, there is no direct measure of statistical significance. Thus, a large amount of interpretation of the results is left to the user. One of the requirements of this type of engineering system is for the results to be as clearly and unambiguously interpretable as possible. There is still difficulty in distinguishing natural variation in the parameters (due to variables which have not been included in the model) from that caused by a fault. This ambiguity is due to a lack of contextual information which would show the expected behaviour of the system (the engines) in a particular configuration of environmental parameters (the current state of the aeroplane, in terms of altitude, speed etc., and the surroundings, in terms of ambient temperature, pressure, etc.). This is addressed in the next chapter, where the concept of context-dependent novelty detection is introduced.

5 Context-based novelty detection

5.1 Introduction

In this Chapter, a method will be introduced for detecting novelty in data sets where the input variables can be logically partitioned into dependent and independent variables. In the case of the data considered here, which is aircraft engine data from the "Quick Access Recorder" (QAR), the "dependent" variables are considered to be the engine parameters (spool speeds, fuel flow, gas path temperatures) while the "independent" variables are the variables giving the aircraft's altitude and airspeed, and the ambient temperature and pressure of the air, together with throttle position and power setting (engine pressure ratio). In principle the method could be applied to any set of data where one set of the inputs could be considered to depend on the values of another subset of the inputs, and where one wishes to monitor departures in the expected values of the dependent variables given the independent variables. The independent variables can be regarded as specifying the "context" in which novelty detection on the dependent variables is carried out. The difference from multivariate regression is that a self-organising map is used to perform dimensionality reduction on both the dependent and independent variable sets. The method is thus suited to highly multivariate and nonlinear problems. This chapter explores issues in setting up such a model and using it in practice for monitoring and novelty detection on aircraft engine data.

5.1.1 Description of the method

The approach taken here forms a three-stage monitoring system which allows for novelty detection in a variety of senses as described below and is based on the concept of dimensionality reduction. The input variables are divided into two sets. In this application these sets correspond, broadly speaking, to airframe-related and engine-related variables. "Airframe" data are defined here as are quantities which will be the same for all engines on the same aircraft, such as {altitude, air speed, air temperature and pressure}. In the case study, control parameters such as throttle resolver angle or parameters which specify a required level of performance, such as EPR (engine pressure ratio) have been included in this set. Engine data is taken to include the spool speeds (N1, N2, N3) of the three rotating shafts, the fuel flow (FF), turbine gas temperature and exhaust gas temperature (see Table 5.4). A self-organising map (SOM, Kohonen, 1982, 1988) is trained for the airframe data

and a separate SOM for the engine data. The effect of this is that the processing elements in the two SOM grids represent different *states* or points on the flight *trajectory*.

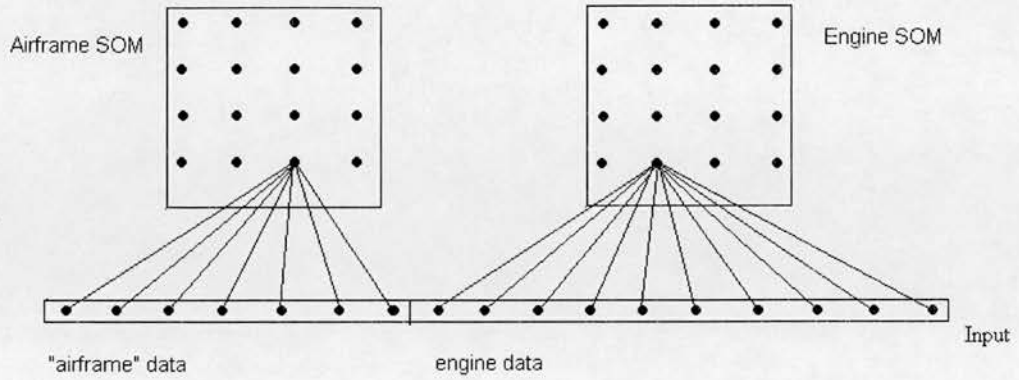


Figure 5.1

Figure 5.1 shows schematically the arrangement of the two SOMs. Note that there is full connectivity between the "airframe" part of the input layer and the airframe SOM, and between the "engine" part of the input layer and the "engine" SOM.

Each SOM can be used to detect "novelty" (expressed as a function of distance between the current record in the data and the nearest of the templates in the SOM grid), for its own subset of the input variables. Thus, the two Kohonen SOMs provide two levels of monitoring or novelty detection capability: *viz.* novelty with regard to the airframe parameters*: "Has the system seen the aircraft flying in this sort of way before?"* and novelty with regard to the engine parameters (*"Has the system seen the engines behaving in this sort of way before?"*) respectively.

The third level of novelty detection or monitoring ability, as alluded to above, arises from having a mapping between the two SOMs, which maps from the airframe states to the engine states. This mapping, and its associated residual errors, give us information about the extent of novelty in the engine parameters *in the context of* the airframe parameters, that is, answering the question, *"Given that the system recognises the state of the airframe and environment parameters (the airframe data), has it within this context seen 'this' behaviour from the engine(s)?".* It is this question which is the nub of effective engine condition monitoring, and which forms a natural extension of standard methods of engine condition monitoring where a thermodynamic model is derived based on combinations of simple environmental variables such as ambient pressure and temperature (P2 and T2), altitude and air speed (or Mach number).

This mapping is in general a one-many mapping. It can be specified in terms of the conditional probabilities of a particular node in the engine data SOM winning, given the airframe data and the airframe SOM node activations. These probabilities can be evaluated by enumerating cases or by training a neural network. In the case of small SOM grids there is no particular advantage in using a neural network for this mapping. In the case of a large SOM, some advantage might be gained from using a feedforward neural network in this case, as the complete set of combinations of states would not then have to be enumerated. possible combinations of the three outcome measures are described in Table 5.1.

Minimum distance on airframe SOM	Minimum distance on "engine" SOM	Residual on SOM-SOM mapping	Explanation
Low	Low	Low	Normal condition. This engine behaviour has been seen in these flying conditions.
Low	Low	High	Familiar flying conditions; familiar engine behaviour, but this engine bahviour has not been seen for these flying conditions (perhaps because of a change in some external parameter not represented in the model)
Low	High	Low	This engine behaviour has not been seen before. Although the minimum distance is high, the nearest node is one which has been visited before in this airframe context.
Low	High	High	This engine behaviour has not been seen before. The minimum distance is high, and the nearest node is one which has not been visited before in this airframe context
High	Low	Low	Although this engine behaviour has been seen before, the current airframe conditions are novel. The *nearest* airframe state is one where the current engine behaviour has been previously observed.
High	Low	High	Although this engine behaviour has been seen before, the current airframe conditions are novel. Also, the nearest airframe state is one where the current engine behaviour has *not* been previously observed.
High	High	Low	Both airframe and engine data are showing novelty, but the nearest engine state correlates with the nearest airframe state in the past data.
High	High	High	Both airframe and engine data are showing novelty. The nearest engine state has not been observed in conjunction with the nearest airframe state in the past data.

Table 5.1: Possible combinations of responses from the three components of the context-based novelty detection system.

Note that even in the case of the combination (high, high, high), this is not necessarily indicative of a *fault,* as such; merely a novel situation. Therefore the algorithm described above constitutes only part of a condition monitoring system, and to be of practical use would have to be accompanied by an inspection of the input data or the use of some other technique to arrive at a diagnosis of engine condition. This "other technique" could consist

of confirmatory analysis using a neural network in another mode (classification or regression, for example, as described in Chapter 4.

5.1.2 Possible variations

Options for the SOM-SOM mapping.

There are several ways in which the detail of this mapping can be implemented. In particular one can choose whether to perform the mapping on a state-by-state basis (a separate mapping for each state in the airframe map, predicting the distribution of states in the engine map), or on a distribution-to-distribution basis (from the distribution of states in the airframe data to the corresponding distribution of states in the engine data).

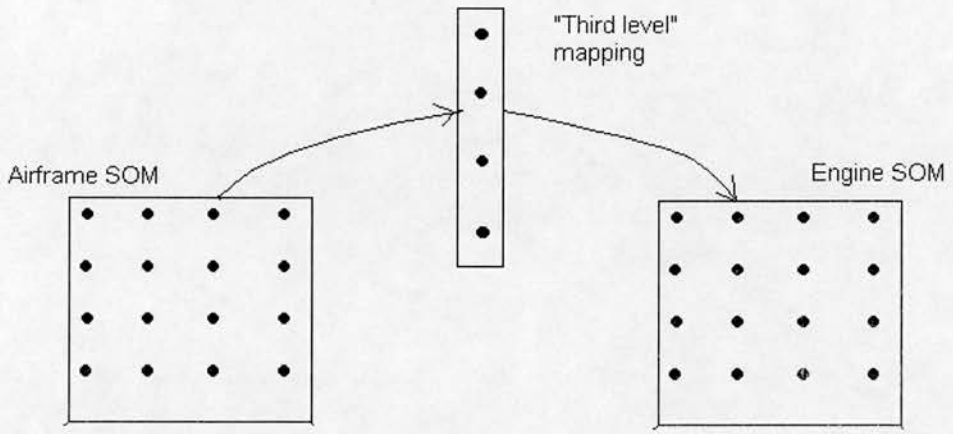


Figure 5.2: schematic of the arrangement of the third level or "trajectory" mapping (which may be implemented as a lookup table of frequencies or probabilities rather than a neural network as such).

Using a feedforward neural network

The output values of the Airframe SOM would be used as the inputs of a feedforward network (which could be implemented as an MLP or an RBF network, for example), and the outputs of the Engine SOM would be used as the target outputs. The same training data that has already been used to train the SOMs would be used, the SOMs being run in recall mode. In the case of a Winner-Take-All rule, the input and output encoding would be a 1 for the winning element and 0 (or -1) for the others. An alternative method is to use the Euclidean distances of each template node from the current data record as inputs and targets for the network.

Using a lookup table

The simplest method of relating the output of the airframe map to the output of the engine map is to tabulate the output data and form a lookup table, such that for each airframe state the frequency of occurrence of each engine state is calculated, and then normalised to give a probability figure, $p(s_E|s_A, D_{tr})$, where s_E is the engine state (here represented by the number of the winning node in the engine SOM); s_A is the airframe state (here represented by the winning node in the airframe SOM) and D_{tr} represents the training data.

5.2 Implementation

The level of detail at which this representation occurs depends on the size of the grid and on how much of the flight envelope (or space of variation of the input parameters) is covered by the data. The winning nodes tell us where in the profile or trajectory the data point is (or in fact the closest point to it). The values of the distances tell us how novel the data point is compared to the data previously seen by the network, if the following conditions are satisfied: the network having been trained adequately, having enough nodes to represent the data adequately at the level which is necessary for this analysis, the mapping being "reasonably smooth" (note the points made in section [3.5.1] about discontinuities in the mapping), and the training data being suitably representative of the conditions we wish to study (that is, what are treated as "good data", really are good data!).

The main guiding principles here are to ensure the network's invariance to routine variation due to such factors as weather (some of the measured engine and intake parameters may be, for example, critically dependent on ambient air temperature and wind speed), variations in origin, destination, routing, and payload, also air traffic control and noise abatement restrictions; while at the same time ensuring that the network is sensitive to trends which may give an early warning of events requiring attention, and sensitive enough to detect anomalies which may be evident only for a short time, though nonetheless important in engineering terms.

False alarm rate is also an important concern. Because we are dealing with a system that would be, for operational purposes, in *secondary* usage, (*i.e.* it would not replace the existing monitoring system or be relied on for safety purposes), the cost of misclassifying bad data as good is moderated, though if the misclassification rate in this direction were too high, obviously one would seriously question the validity of the system. However the cost of

false alarms (identifying good data as bad) could be high both in terms of time and money spent in "no fault found" cases, and in customer service considerations.

These requirements and principles can be seen as an extension of the idea of generalisation ability and impose a stronger set of conditions on which to evaluate the behaviour of a trained neural network than the 'ability to generalise to unseen data'.

5.2.1 Parameters of the system

The framework outlined above can be implemented in any of a number of ways. It is worthwhile at this point to outline the 'parameters of the system', that is which things may vary, or be chosen.

- The true dimensionality of the input manifold, *i.e.* how many dimensions are required to represent the variation in the airframe data.
- The true dimensionality of the output manifold, *i.e.* how many dimensions are required to represent the variation in the engine data.
- Which variables are included in the context (airframe) input set.
- Which variables are included in the engine data input set.
- The nature of the input and output topographic mappings (for this one could consider using one of the following techniques: SOM, autoassociative network, recirculation network (Hinton & McClelland, 1988), or generative topographic map. (In the following discussion this mapping is referred to as a "SOM", but similar principles would apply were one of the other methods to be used.)
- Whether, after training, the SOM layers are implemented with a *winner-take-all* rule or whether the *raw distances* are used.
- The size of the input (airframe) mapping SOM grid.
- The size of the output (engine) mapping SOM grid.
- A threshold value, relating to the distance in the input (airframe) SOM, which represents the level at which we say that novelty in the airframe data has been detected.
- An equivalent threshold value for the output (engine) SOM. (The level at which we say that novelty in the engine data has been detected).
- Whether the "third" mapping (from the airframe SOM to the engine SOM) is implemented on a *state-by-state* or *distribution-to-distribution* basis.
- The method used to implement the third mapping. For this purpose we consider options of: {multi-layer perceptron, radial basis function or lookup table}.

- The model complexity of the third mapping. If the third mapping is being implemented as a neural network, the choice is over the size of the hidden layer, and this determines whether the network acts as a "bottleneck" or as a content-addressable memory.
- The level of error expected on the SOM - SOM mapping.
- Invertibility: whether it makes sense to perform the mapping in the reverse direction also.
- Whether the Kohonen grids are inspected record-by-record or whether they are used to represent average or accumulated behaviour over a number of data records.

The above list can be used as a checklist to guide the design process for a network of this type, and to ensure that domain knowledge is used in the best way to match the network design to the application in hand.

5.2.2 Possible outcomes of training the network

The following section shows diagrammatically the configurations of the system which may arise. From a design point of view, it is important that the overall system produces the appropriate action or response for each of these cases.

With **winner-take-all** networks, situations (a) and (b) are possible.

(a) The case where the prediction from the SOM-SOM mapping is accurate, *i.e.* the actual winning node in the engine SOM is the same as the predicted winning node, as shown in Figure 5.3.

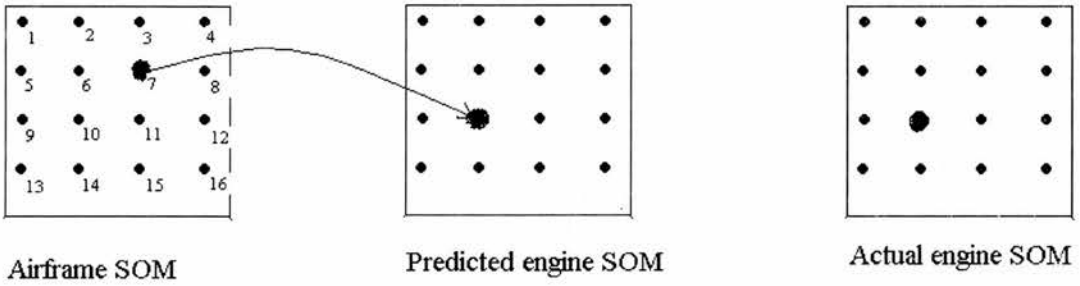


Figure 5.3: example of correct prediction of engine SOM winning node from airframe SOM

In Figure 5.3, the airframe SOM has calculated that the closest template to the current airframe data record is node 7 (let us write s_A=7). The SOM-SOM mapping (or lookup table) gives the most likely value for the active engine SOM node to be 10. ($\hat{s}_E = 10$). In this case the engine SOM gives that the nearest template to the current engine data record is node 10 *(s_E=10).*

(b) The case where there is a discrepancy between the predicted and actual engine SOM winning node co-ordinates, as shown in Figure 5.4

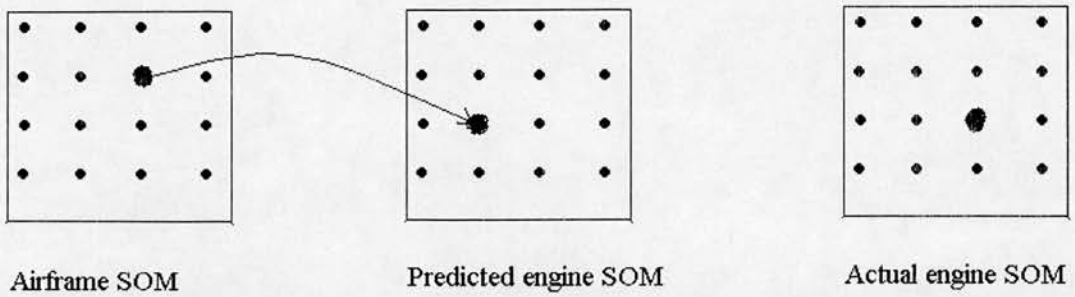


Figure 5.4 discrepancy between the predicted and actual engine SOM

In Figure 5.4, we also have s_A=7, $\hat{s}_E = 10$. In this case the engine SOM gives that the nearest template to the current engine data record is node 11 *(s_E=11).*

Use of raw distances:

By relaxing the Winner-Take-All rule and considering activations of the *k* template nodes nearest in euclidean distance to the current record in the data, situations (c), (d) and (e) are also possible. These situations can also arise when considering a sequence of data, where the winner-take-all rule has been applied.

(c) This is the case where the airframe data always maps onto a single point but the engine data maps onto a distribution over a number of states, and results in a situation such as is shown in Figure 5.5.

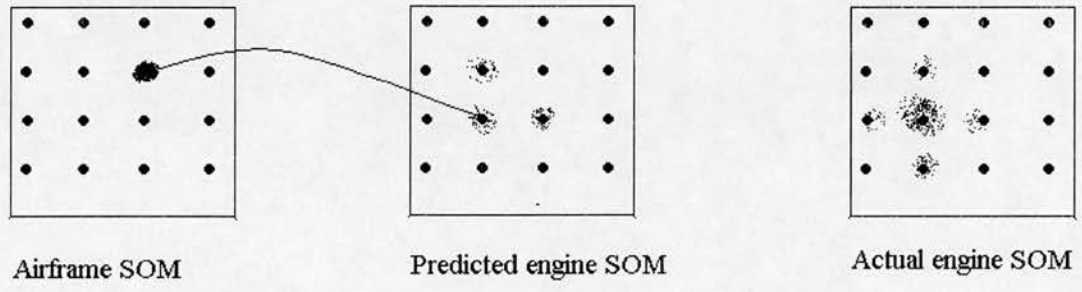


Figure 5.5: distribution in engine SOM.

Worked Example

In Figure 5.5, an artificial example is shown, again using s_A=7; In this case, rather than a single value of $\hat{s}_E$, predicted and actual distributions are shown over the states (6, 10, 11). The actual SOM trained on the engine data gives a slightly different distribution centred on

state 10 but with some occurrences of neighbouring states 6, 9, 11 and 14. In tabular form, this might be represented as shown in Table 5.2.

Airframe state (from Airframe SOM)	*Predicted engine state*	*Engine state from engine SOM*
s_A	$\hat{s}_E$	s_E
7	6	6
7	6	10
7	10	9
7	10	10
7	10	11
7	11	11
7	11	14
7	10	10
7	10	10
7	6	6

Table 5.2: hypothetical example of a mapping from one airframe state to a variety of engine states

Table 5.2 shows a hypothetical example of a case where a mapping from one airframe state (in this case labelled 7) to a variety of engine states, and the way in which this might be approximated by the SOM-SOM map. The table shows typical values for predicted and actual states for a sequence of data which is classified by the airframe SOM as belonging to one airframe state.

Engine state	*Conditional probability of expected engine state from SOM-SOM map*	*Actual conditional probability calculated from results of Airframe SOM and Engine SOM*
n	$p(\hat{s}_E = n \mid s_A = 7)$	$p(s_E = n \mid s_A = 7)$
6	0.3	0.2
9		0.1
10	0.5	0.4
11	0.2	0.2
14		0.1

Table 5.3: conditional probabilities of the engine states corresponding to each airframe state for hypothetical example.

Table 5.3 shows, for the above example, conditional probabilities (calculated as normalised frequencies of occurrence of the engine states corresponding to each airframe state), both as expected from the results of the SOM-SOM map, and as observed by comparing the Airframe and Engine SOMs.

(d) In Figure 5.6, an example is shown where a diffuse region in the airframe SOM maps to single state (or very narrow region) in the engine (output) SOM. The zero-error case is illustrated. Non-zero error between predicted and actual is also possible.

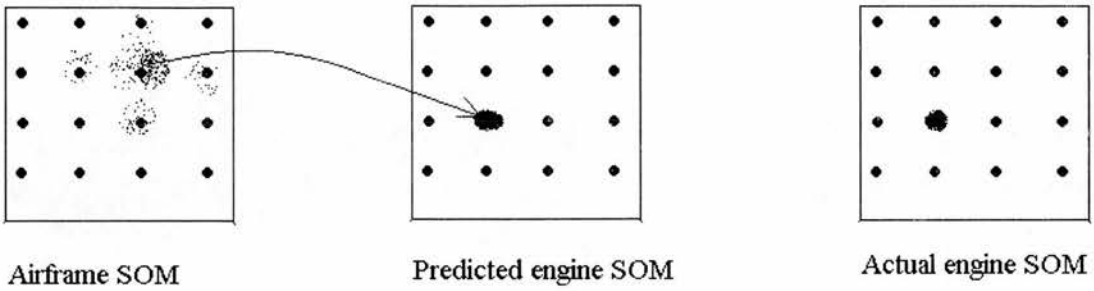


Figure 5.6: sequence of data with different airframe states but a constant engine state.

(f) Figure 5.7 shows the case where a diffuse region in the input SOM maps to a diffuse region in the output SOM. Once again, the zero-error case is shown; error between predicted and actual is also possible. The question arises of how to define a suitable error metric in this case. Note that one can avoid this situation by splitting the data sequence into shorter sub-sequences.

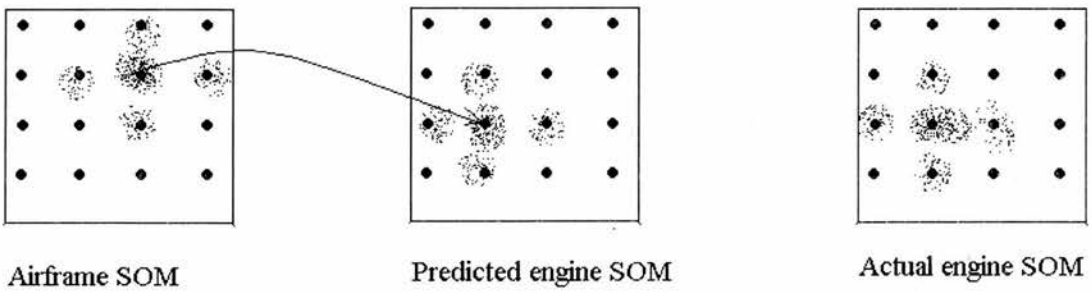


Figure 5.7: sequence of data with a distribution over airframe states and a distribution over engine states.

Using test data

Consider next the cases which can arise when dealing with unseen data where there is some difference from the training set.

(g) Consider a test set point where the input layer maps to a narrow region in input (airframe) SOM but one which has not been visited in the training set. This represents one type of novelty in the input SOM. However, if the training set has been constructed appropriately and the SOM has been trained correctly, this situation should not occur in practice. See case (h).

(h) A more usual case in this situation would entail complete novelty on the input SOM, that is, with the distances to all the template nodes high. The output SOM could show: a well-defined point, a diffuse region or all distances high.

Error conditions

In addition certain error conditions have to be addressed:

(i) For the input or the output SOM (or both), novelty may be present but not detected. That is, the distance measure being used may be invariant to the kind of discrepancy which we wish to detect.

(j) Spurious novelty may be detected (that is, the distance measure varies but in response to a change in which we are not interested).

Consideration of the previous two cases provides some justification for adjusting or weighting the distance function according to the application. Alternatively the required tuning of sensitivity could be achieved by a suitable normalisation or transformation of the input data before passing through the SOM.

Furthermore, if a neural network is used to learn the mapping between the two SOMs, then there may be some suboptimality in the SOM-SOM mapping network, caused by:

(k) the fact that the SOM-SOM mapping network may not converge, or may converge to an unacceptable local minimum of the error function.

In such cases, if the number of SOM nodes is small enough, one could still use a lookup table to represent the conditional frequencies. Throughout the above discussion, the assumption has been made that the representations of the two data subsets, formed by the two Kohonen networks, are acceptable. Some thought also needs to be given to limitations which arise from the Kohonen method itself (see chapter 3).

5.3 Condition monitoring case study

We shall now address the use of the context-based novelty detection framework, as described above, with operational data, with the intention of making some observations and judgments about its strengths and limitations in real-world use.

5.3.1 Introduction: Case study using 747-400 data

This section describes a case study in which the algorithm described above was tested on some real data gathered in flight on Boeing 747-400 aircraft in normal scheduled service at British Airways.

5.3.2 Description of the input data

SOM 1 ("airframe"):	TAT	Air temperature
	ALT	Altitude
	CAS	Air speed [of aircraft]
	P2	Ambient atmospheric pressure
	TRA	Throttle resolver angle [engine 1]
	EPR	Engine pressure ratio [engine 1]
SOM 2 ("engine"):	N1	Low pressure spool speed
	N2	Intermediate pressure spool speed
	N3	High pressure spool speed
	FF	Fuel flow
	TGT	Turbine gas temperature
	EGT	Exhaust gas temperature [T50]

Table 5.4: input variables belonging to the airframe data set and engine data set, these being the inputs to the airframe SOM and engine SOM respectively.

Effects of the Flight Management Computer

The values of throttle resolver angle (TRA) and engine pressure ratio (EPR) can be governed by an automatic control system known as the Flight Management Computer. Given values for the altitude, ambient pressure and temperature, required airspeed, and length of the runway and weight of the aircraft, the flight management computer (FMC) calculates the power required from the engines, in terms of engine pressure ratio (EPR). Effects due to this fact will be reflected in the values of the other engine parameters. Note that as aircraft weight and runway length are not used as inputs in the neural network models described in this thesis, this will give rise to some "random" variation or noise, due to variations in these parameters which have not been taken into account. Given that the automatic throttle advancement stops when the required EPR is obtained from one of the engines, it will usually be the case that the EPR values for the four different engines (on a four-engined aircraft such as a Boeing 747) will be slightly different. The approach which has been taken here is to select one of the engine positions (in this case the number-1 position) as an arbitrary "standard". The variables TRA and EPR are, for the purposes of this model, treated along with the environmental variables of altitude, mach number, air temperature and pressure as part of the "airframe" input set. A case could be made for treating them separately, in a category of "control parameters". This would introduce further complexity into the model, as there would then be a necessity to model the relationship between three sets of variables (one would have a system with up to three SOMs and three inter-SOM mappings or lookup tables). This option has not been explored further in the current work.

5.3.3 Training strategies and training set construction

In constructing a training and test set so that the neural network achieves the desired generalisation properties, there are several possible training strategies (see also Section 4.4.2):

(i) The network is trained on one engine position (in this case position number 1) and recalled on all the others. Although one would expect a high degree of uniformity given that the operating conditions are the same for the four engines, the network in fact assigns the engines to different states quite often. Sometimes this is transient and lasts only a second or two and usually the states involved are close together in state space. If it is known that the

data set consists only of good engine data one can use an exercise such as this to calibrate clusters of states which behave similarly.

(ii) Alternatively one can train the network on data for all engines on each aircraft, recalling on subsequent data for that same aircraft. This removes the danger of overfitting to the behaviour of one particular engine but increases the possibility of overfitting to the operational conditions represented by the training set data.

In practice there may be many such options based on the dimensionality of the *domain data* (here I have used the term 'domain data' to refer to the ensemble of possible training / test set combinations which could be chosen). Some of these domain dimensions could be represented by having multiple neural networks, for example. Here one is reminded of issues in classical experimental design: of the requirement to keep as many variables as possible constant or controlled. In the light of the fact that neural network methods are almost invariably a "method of last resort" when the dimensionality of the data space is very high, it will be noted that there is a fundamental difficulty here. In practice it will often be found however that there is sufficient continuity over time and collinearity in the inputs that the underlying dimensionality of the domain is not as high as some input data selections may suggest. Recent and continuing work on the application of the concept of latent variable models to the neural net modelling domain could potentially be used as a basis of further research in this area.

(iii) A third option is to train on data from a number of aircraft and to generalise to different individual aircraft within a fleet. This is the most desirable option, in that a standard set of training data can be set apart; a separate training set would not have to be constructed for each aircraft. It would also make the resulting trained system more robust to engine changes.

Training Set

The training set consisted of Quick Access Recorder (QAR) data from eight 747-400's (with Rolls-Royce RB211-524G engines) collected at one second intervals for takeoff and the first part of climb for 12 flights in all. The data set contained 2176 records. A 3x3 SOM was used for the airframe and engine data. The SOMs were trained for 20,000 cycles, on data for the position-1 engine.

5.3.4 System parameter choices for the case study

Although six airframe variables have been used {altitude, mach number, ambient air pressure, ambient air temperature, throttle resolver angle, engine pressure ratio}, TRA and EPR are closely correlated, so that effectively there are 5 degrees of freedom in the "airframe" data. The engine data {N1, N2, N3, fuel flow, turbine gas temperature, exhaust gas temperature} in the normal state, are well correlated but not linearly related. For the dimension reduction, 2-dimensional Kohonen SOMs have been used in both cases[1], with both the *winner-take-all* rule and the use of *raw distances* being investigated. The size of the input (airframe) mapping SOM grid and the output (engine) mapping SOM grid for this exercise were set to 9 (3 x 3 units). Experiments using 16 units were also carried out but are not reported here. In a full-scale application, it would be necessary, after some exploratory work on a sample of data, to set threshold values relating to the distance in the airframe and engine SOMs, to determine the level at which we say that novelty in the airframe and engine data has been detected. The "third" mapping (from the airframe SOM to the engine SOM) is implemented using a lookup table. Experiments were also conducted using neural networks for this purpose.

5.4 Results

Results of training the 3x3 SOMs on these data are shown in Table 5.6 and Table 5.7. It will be seen from Table 5.6 (by examining the values of altitude, for example) that the nodes representing the airframe data show an approximate ordering from 9 to 1 in increasing order of altitude. In the case of the "engine" SOM in Table 5.7. It can be seen that there is an ordering in terms of fuel flow (and approximately, in terms of spool speed) from 1 to 9, state 1 being highest, and state 9 being lowest. This implies that he relationships in the data are close to linear and that a one-dimensional SOM may be sufficient to model the engine data.

---

[1] The implementation of Kohonen's algorithm used was that of the package NeuralWorks Professional II (NeuralWare, 1988-1995).

5.4.1 Airframe State Templates

Using a 3x3 SOM, Table 5.5 shows the templates, or prototypes, formed by the nodes of the "airframe" SOM after training. The columns labelled '1' to '9' correspond to the 9 SOM nodes, and are referred to later as airframe states 1-9.

Input	SOM node number :	1	2	3	4	5	6	7	8	9
TAT	Air temp /deg. C	22.07	24.30	25.11	22.96	24.22	24.68	24.82	24.13	24.08
ALT	Altitude / ft.	2355	1811	1531	1409	1206	878	401	440	329
CAS	Computed air speed / kts.	225	211	207	195	189	180	149	161	153
P2	Air pressure	14.84	15.01	15.14	15.10	15.19	15.31	15.30	15.39	15.41
TRA	Throttle resolver angle / deg.	67.3	71.7	76.5	69.8	73.2	76.9	71.8	74.9	77.5
EPR	Engine pressure ratio	1.530	1.567	1.617	1.550	1.586	1.624	1.569	1.607	1.635

Table 5.5: Airframe State Templates

airframe state number	1	2	3	4	5	6	7	8	9
1	0.	0.73	1.49	0.67	1.14	1.73	1.37	1.66	2.02
2	0.73	0.	0.77	0.39	0.45	1.02	0.88	1.	1.33
3	1.49	0.77	0.	1.05	0.55	0.42	1.03	0.72	0.79
4	0.67	0.39	1.05	0.	0.56	1.18	0.71	1.02	1.42
5	1.14	0.45	0.55	0.56	0.	0.62	0.56	0.55	0.89
6	1.73	1.02	0.42	1.18	0.62	0.	0.88	0.4	0.38
7	1.37	0.88	1.03	0.71	0.56	0.88	0.	0.54	0.95
8	1.66	1.	0.72	1.02	0.55	0.4	0.54	0.	0.43
9	2.02	1.33	0.79	1.42	0.89	0.38	0.95	0.43	0.

Table 5.6 shows the Euclidean distances between pairs of airframe states.

5.4.2 Engine State Templates

Using a 3x3 SOM, Table 5.5 shows the templates, or prototypes, formed by the nodes of the "engine" SOM after training. The columns labelled '1' to '9' correspond to the 9 SOM nodes, and are referred to later as engine states 1-9.

input		1	2	3	4	5	6	7	8	9
N1	Low pressure spool speed %	104.0	101.6	99.5	99.4	97.8	95.8	95.9	94.5	93.4
N2	Int. pressure spool speed %	106	104	104	103	102	101	100	99	98
N3	High pressure spool speed %	94	94	92	93	92	91	91	90	89
FF	Fuel flow	21522	20249	19161	18631	17988	16932	16612	15792	15496
TGT	Turbine gas temperature / deg.C	725	703	683	696	674	653	668	650	632
EGT	Exhaust gas temperature / deg. C	607	586	563	587	564	544	564	546	529

Table 5.7 : templates, or prototypes, formed by the nodes of the "engine" SOM after training.

5.4.3 Relationship between airframe and engine data

Having trained the two separate neural networks on the airframe and engine data, respectively, it remains to examine the relationship between the two mappings produced, to characterise the extent to which an identifiable mapping exists between the two SOM representations of the data, and to determine how useful this is as a condition monitoring method and any advantage which it confers over alternative methods of treating engine data.

The correlations between the airframe states and engine states identified by the two SOMs are shown for a real example in Tables 5.8 and 5.9. Frequencies of occupation of the various states (that is, the frequencies with which the various template nodes in the SOM are the winners) are shown in Table 5.8; the corresponding *normalised* frequencies are shown in Table 5.9 (that is the values in Table 5.8 divided by the total for the relevant airframe state).

s_A=1		
a	b	c
d	e	f
	1	24
g	h	i
35	291	240

s_A=2		
a	b	c
	1	1
d	e	f
65	3	
g	h	i
34		

s_A=3		
a	b	c
77	72	30
d	e	f
30		
g	h	i

s_A=4		
a	b	c
d	e	f
		40
g	h	i
23	27	74

s_A=5		
a	b	c
	18	44
d	e	f
50	1	
g	h	i
10		

s_A=6		
a	b	c
80	32	
d	e	f
g	h	i

s_A=7		
a	b	c
	8	23
d	e	f
113	53	70
g	h	i
6	19	106

s_A=8		
a	b	c
14	93	70
d	e	f
g	h	i

s_A=9		
a	b	c
178	92	28
d	e	f
g	h	i

Table 5.8 : correspondences between the "airframe" states and the "engine" states after the respective SOMs have been trained. The table gives results for the training set. The arrangement of states in Table 5.7 and the following tables is shown in the following order:

1	***2***	***3***
4	***5***	***6***
7	***8***	***9***

Thus, the top left 3x3 matrix gives the count of occupation of the engine states 1-9 (labelled a-i in the table) for airframe state s_A=1, the next 3x3 matrix to the right gives the engine states for airframe state 2 and so on. Looking at the top-left 3x3 matrix, for records where the airframe state was classified as '1' by the airframe SOM, there was one case where the engine state was classified as '5' (cell 'e') by the engine SOM, 24 cases where it was '6' (cell 'f') and so on.

Examination of Table 5.8 shows that the results of the airframe and engine SOMs are indeed correlated, as would be expected. There is some overlap (that is, more than one airframe state mapping onto one engine state and *vice versa*), as would also be expected.

s_A=1		
a	b	c
d	e	f 0.04
g 0.06	h 0.49	i 0.41

s_A=2		
a	b 0.01	c 0.01
d 0.63	e 0.03	f
g 0.33	h	i

s_A=3		
a 0.37	b 0.34	c 0.14
d 0.14	e	f
g	h	i

s_A=4		
a	b	c
d	e	f 0.24
g 0.14	h 0.16	i 0.45

s_A=5		
a	b 0.15	c 0.36
d 0.41	e 0.01	f
g 0.08	h	i

s_A=6		
a 0.71	b 0.29	c
d	e	f
g	h	i

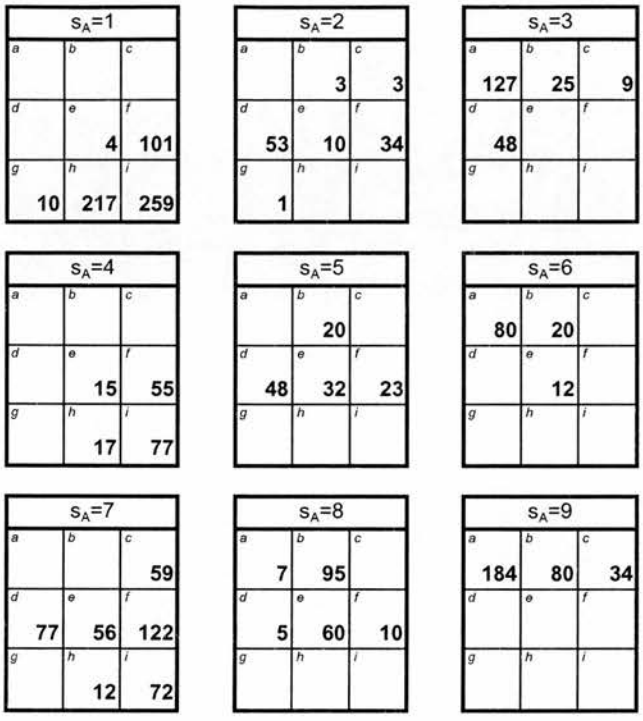
s_A=7		
a	b 0.02	c 0.06
d 0.28	e 0.13	f 0.18
g 0.02	h 0.05	i 0.27

s_A=8		
a 0.08	b 0.53	c 0.4
d	e	f
g	h	i

s_A=9		
a 0.6	b 0.31	c 0.09
d	e	f
g	h	i

Table 5.9 : engine state posterior probabilities for engine number 1, given the airframe state (also, but implicitly, conditioned on the training data and the SOM training).

5.4.4 Comparison of engines on the same aircraft



s_A=1		
a	b	c
d	e 4	f 101
g 10	h 217	i 259

s_A=2		
a	b 3	c 3
d 53	e 10	f 34
g 1	h	i

s_A=3		
a 127	b 25	c 9
d 48	e	f
g	h	i

s_A=4		
a	b	c
d	e 15	f 55
g	h 17	i 77

s_A=5		
a	b 20	c
d 48	e 32	f 23
g	h	i

s_A=6		
a 80	b 20	c
d	e 12	f
g	h	i

s_A=7		
a	b	c 59
d 77	e 56	f 122
g	h 12	i 72

s_A=8		
a 7	b 95	c
d 5	e 60	f 10
g	h	i

s_A=9		
a 184	b 80	c 34
d	e	f
g	h	i

Table 5.10: frequencies of state occupation for the data from the number-2 position engine from the same flight data set as Table 5.8

Given the same airframe data, differences in engine parameters are apparent *i.e.* the patterns of winning nodes in the Kohonen network when trained on engine 1 and recalled on engine 2 show some noticeable differences from the result when recalled on engine 1 data. The probable reason for this is that in this example, information on the control parameters *throttle resolver angle* and *engine pressure ratio* for engine 1 has been included in the "airframe" data set.

5.4.5 Use of trajectories or sequences of states

Using continuous FDR data which contains a sequence which passes in time order through various flight phases, *viz.* taxiing, throttle opening, takeoff and early climb, and applying the 2-SOM method, one obtains a sequence, or trajectory, through predicted and actual (airframe, engine) state pairs. This is illustrated for the 747-400 take-off data in Table 5.11.

				sample of data at start of takeoff sequence				sample of data at end of takeoff sequence				
take-off no.	aircraft	origin [2]	destin-ation	TAT[3]	TRA[4]	EPR[5]	N1[6]	TAT	TRA	EPR	N1	sequence of (airframe, engine) state pairs[7]
1	i	BKK	SYD	33	73	1.58	98	33	67	1.49	93	(7,4)(5,4)(2,4)(1,7)
2	ii	LHR	KUL	24	79	1.69	102	24	66	1.53	94	(9,1)(6,1)(3,1)(1,8)
3	iii	LHR	NRT	25	76	1.65	101	23	66	1.53	94	(9,1)(8,1)(6,1)(3,1)(1,8)
4	iv	EZE	LHR	25	80	1.75	107	28	66	1.53	93	(9,1)(6,1)(3,1)(1,8)
5	v	LHR	BOS	16	70	1.58	94	18	70	1.53	92	(7,9)(4,9)(1,9)
6	vi	LHR	JFK	22	70	1.57	95	26	68	1.51	93	(7,9)(7,6)(4,6)(4,9)(1,9)(1,8)
7	vii	LHR	GRU	18	73	1.62	98	22	66	1.53	93	(7,3)(8,3)(5,3)(1,9)(1,6)
8	viii	LHR	NRT	20	71	1.6	99	20	66	1.54	94	(7,5)(5,7)(4,7)(2,7)(1,9)
9	viii	NRT	LHR	29	73	1.6	98	25	67	1.51	93	(7,4)(5,4)(2,4)(1,8)
10	viii	LHR	NRT	22	74	1.65	100	20	66	1.54	94	(8,2)(5,2)(3,4)(1,8)
11	viii	CPT	LHR	10	78	1.7	101	9	67	1.51	90	(9,2)(6,2)(1,9)
12	viii	PER	SIN	15	70	1.56	93	21	71	1.54	94	(7,9)(7,8)(4,8/9)(1,9)

Table 5.11 : trajectories through the state pairs for each of the twelve take-offs

Table 5.11 gives, for the number-1-position engine in each case, for the training data used above, the trajectories through the state pairs for each of the twelve take-offs in the training set, together with the values of a subset of the variables in the input data, sampled at the start and end of the data sequence for each take-off. The "state pair sequences" given in the right-hand column are consecutive pairs of (airframe, engine 1) states. (In this table, some

[2] Codes given are airport codes, with meanings as follows: BKK-Bangkok; SYD - Sydney; LHR - London Heathrow; KUL - Kuala Lumpur; NRT - Tokyo; EZE - Buenos Aires; JFK - New York; GRU - Sao Paolo; CPT - Cape Town; PER - Perth, Australia; SIN - Singapore.

[3] Air temperature (in deg. C)

[4] Throttle resolver angle (in degrees)

[5] Engine pressure ratio

[6] Low pressure spool speed. (% of nominal)

[7] The ordered pairs in brackets give the context (airframe) template followed by the engine template. The sequence of state pairs gives an indication of the "trajectory" of context and engine parameters in the take-off phase.

of the state pairs which arise purely as transients are omitted for clarity, but their presence is indicated by some of the low frequency counts in Tables 5.8 and 5.9). [8]

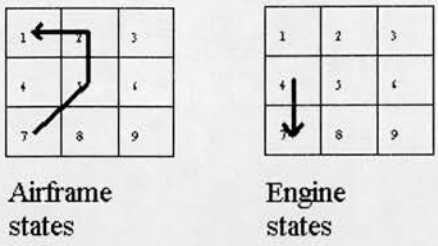


Figure 5.8 : trajectory through the airframe and engine state spaces represented by the sequence for take-off number (1) in Table 5.11.

A qualitative evaluation of the results presented in Table 5.11 can be gained by visual inspection of the right-hand column of the table together with the data samples given, and shows that the combination of two SOMs has correctly identified some consistency and some structure in the data, and has corroborated the observation that there is more variability in the data at the start of the takeoff sequences than at the end. Figure 5.8 shows an example. In the data shown, airframe sequences start with state 7, 8 or 9 and always end at state 1. That is, templates 7, 8, and 9 represent ground activity, while template 1 represents climb. See Tables 5.5 and 5.6.

Some common patterns emerge and it is seen that once the engines have been throttled back slightly, after about 1000 ft. altitude, the engine parameters look more homogeneous and this is reflected by the number of states visited at that stage (nearly all (1,8) and (1,9)). The variation which is evident at the take-off stage corresponds to known variation in throttle and power settings usually referred to as "take-off derate bands", and is confirmed by observing that take-offs on particularly long haul flights, (for example between London (LHR) and Kuala Lumpur (KUL), Tokyo (NRT), Buenos Aires (EZE) and Cape Town (CPT)), because of the extra fuel weight, require higher throttle resolver angle settings and correspondingly higher engine pressure ratio and spool speeds, than the comparatively short journeys London to New York (JFK) or Boston (BOS).

[8] Note that the values in Table 5.2 are not state-*transition* probabilities (as would be used in a Markov chain, for instance), but rather state co-occurrence probabilities.

5.4.6 Use of neural networks for the third mapping

In the above work, the mapping between the outputs of the two SOMs was performed using a lookup table. Neural network methods of implementing this mapping were also examined. A radial basis function network with 36 hidden units was compared with an MLP with 18 hidden units.

For this purpose, in the case of winner-take-all SOM networks as input and output, generalisation to new data is not a requirement. It was found that the MLP with 18 hidden units was able to learn the mapping with a very small residual error (RMSE 0.3%). With 36 hidden units, the RBF network gave and RMSE of 7%. Results for the training set are given in Tables 5.12 and 5.13. These should be compared with the lookup table values in Table 5.9.

s_A=1		
a	b	c
d 0.02	e	f
g 0.02	h 0.63	i 0.33

s_A=2		
a	b	c
d 0.77	e	f
g 0.18	h 0.04	i

s_A=3		
a 0.34	b 0.31	c 0.11
d 0.20	e	f
g	h 0.04	i

s_A=4		
a 0.01	b	c
d 0.03	e	f 0.20
g 0.03	h 0.32	i 0.41

s_A=5		
a	b 0.12	c 0.27
d 0.55	e	f
g 0.02	h 0.04	i

s_A=6		
a 0.70	b 0.25	c
d 0.02	e	f
g	h 0.03	i

s_A=7		
a 0.01	b 0.05	c 0.01
d 0.55	e 0.07	f 0.14
g	h 0.11	i 0.06

s_A=8		
a 0.03	b 0.57	c 0.33
d 0.03	e	f
g	h 0.04	i

s_A=9		
a 0.80	b 0.11	c 0.03
d 0.03	e	f
g	h 0.03	i

Table 5.11. Predicted normalised frequencies using an RBF network with 36 centres (RMS error 7%)

s_A=1		
a	b	c
d	e	f 0.04
g 0.06	h 0.49	i 0.41

s_A=2		
a	b 0.01	c 0.02
d 0.62	e 0.03	f
g 0.33	h	i

s_A=3		
a 0.37	b 0.34	c 0.15
d 0.14	e	f
g	h	i

s_A=4		
a	b	c
d	e	f 0.25
g 0.15	h 0.17	i 0.44

s_A=5		
a	b 0.16	c 0.35
d 0.41	e 0.01	f
g 0.08	h	i

s_A=6		
a 0.71	b 0.28	c
d	e	f
g	h	i

s_A=7		
a	b 0.02	c 0.06
d 0.28	e 0.14	f 0.18
g 0.01	h 0.05	i 0.27

s_A=8		
a 0.08	b 0.52	c 0.4
d	e	f
g	h	i

s_A=9		
a 0.59	b 0.31	c 0.10
d	e	f
g	h	i

Table 5.12. Predicted normalised frequencies using an MLP (9-18-9); trained for 100,000 passes with an epoch size of 4 (RMS error 0.3%)

With small SOMs, there is no advantage in using a neural network over using a lookup table; however, when considering the case of a pair of larger SOMs, the table specifying the mapping from the one to the other could get large, and a neural network mapping could be more efficient.

5.4.7 Results on test set

The test set consisted of a set of six take-offs collected between 31 August and 6 September 1996, for one aircraft. The airframe inputs were passed through the airframe SOM and the engine inputs were passed through the engine self-organising map, and the resulting distributions of SOM prototypes, or "states" are tabulated in Table 5.13(a)-(d). The same presentation of the results is used as in Section 5.4.3. This presentation is used in order to show the structure extracted by the network. (Each airframe state maps to a limited set of engine states).

As in Tables 5.1-5.3, the order of the engine states in the 3x3 table is

1	2	3
4	5	6
7	8	9

indicated by the letters *a* to *i*.

s_A=1		
a	b	c
d	e	f
g	h	i
0.21	0.51	0.28

s_A=2		
a	b	c
d	e	f
0.04		
g	h	i
0.96		

s_A=3		
a	b	c
0.85	0.15	
d	e	f
g	h	i

s_A=4		
a	b	c
d	e	f
	0.18	
g	h	i
0.18	0.62	0.03

s_A=5		
a	b	c
d	e	f
0.46	0.54	
g	h	i

s_A=6		
a	b	c
0.51	0.49	
d	e	f
g	h	i

s_A=7		
a	b	c
d	e	f
	0.50	0.26
g	h	i
	0.21	0.04

s_A=8		
a	b	c
		0.28
d	e	f
0.72		
g	h	i

s_A=9		
a	b	c
0.49	0.43	0.08
d	e	f
g	h	i

Table 5.13(a): normalised frequencies of each engine SOM prototype: Engine 1

s_A=1		
a	b	c
0.01		
d	e	f
		0.12
g	h	i
0.04	0.25	0.58

s_A=2		
a	b	c
		0.04
d	e	f
0.04		0.24
g	h	i
0.72		

s_A=3		
a	b	c
0.69	0.17	0.14
d	e	f
g	h	i

s_A=4		
a	b	c
d	e	f
	0.10	0.04
g	h	i
0.21	0.04	0.60

s_A=5		
a	b	c
d	e	f
	1.00	
g	h	i

s_A=6		
a	b	c
0.25	0.40	0.35
d	e	f
g	h	i

s_A=7		
a	b	c
d	e	f
	0.38	0.15
g	h	i
	0.12	0.34

s_A=8		
a	b	c
		0.28
d	e	f
0.72		
g	h	i

s_A=9		
a	b	c
0.46	0.42	0.11
d	e	f
	.01	
g	h	i

Table 5.13(b): normalised frequencies of each engine SOM prototype given the airframe SOM prototype. Engine 2.

s_A=1		
a	b	c
d	e	f 0.14
g	h 0.05	i 0.80

s_A=2		
a	b	c
d	e 0.04	f 0.96
g	h	i

s_A=3		
a 0.69	b 0.11	c 0.19
d	e	f
g	h	i

s_A=4		
a	b	c
d	e	f 0.35
g	h	i 0.65

s_A=5		
a	b	c
d	e 0.03	f 0.97
g	h	i

s_A=6		
a	b 0.51	c 0.49
d	e	f
g	h	i

s_A=7		
a	b	c
d	e	f 0.52
g	h	i 0.48

s_A=8		
a	b	c
d 0.72	e 0.47	f 0.53
g	h	i

s_A=9		
a 0.11	b 0.36	c 0.48
d	e	f 0.05
g	h	i

Table 5.13(c) : normalised frequencies of each engine SOM prototype given the airframe SOM prototype. Engine 3.

s_A=1		
a 0.01	b	c
d	e	f 0.09
g 0.09	h 0.22	i 0.59

s_A=2		
a	b	c 0.04
d 0.04	e 0.28	f
g 0.68	h	i

s_A=3		
a 0.70	b 0.17	c 0.13
d	e	f
g	h	i

s_A=4		
a	b	c
d	e 0.34	f 0.09
g	h 0.09	i 0.49

s_A=5		
a	b	c
d	e 1.0	f
g	h	i

s_A=6		
a 0.47	b 0.45	c 0.07
d	e	f
g	h	i

s_A=7		
a	b	c
d	e 0.37	f 0.35
g	h	i 0.28

s_A=8		
a	b	c 0.90
d	e 0.10	f
g	h	i

s_A=9		
a 0.46	b 0.38	c 0.15
d	e	f 0.01
g	h	i

Table 5.13(d) : normalised frequencies of each engine SOM prototype given the airframe SOM prototype. Engine 4

5.4.8 A measure for context-sensitive novelty detection.

For an unseen data record r, whose engine state as determined by the Engine SOM is $s_E(r)$, let p denote the proportion of times in the training set for which that particular engine state occurred in conjunction with the airframe state $s_A(r)$ observed for r. The value p used here is defined as follows:

$$p = \Pr(\hat{s}_E(s_A(r)) = n | s_E(r) = n), \qquad \text{where } \hat{s}_E = M(s_A)$$

If we let q = 1-p then the value of q can be considered as a **context-sensitive novelty measure,** which gives an indication of how familiar the recall data is, given the network trained on the training data.

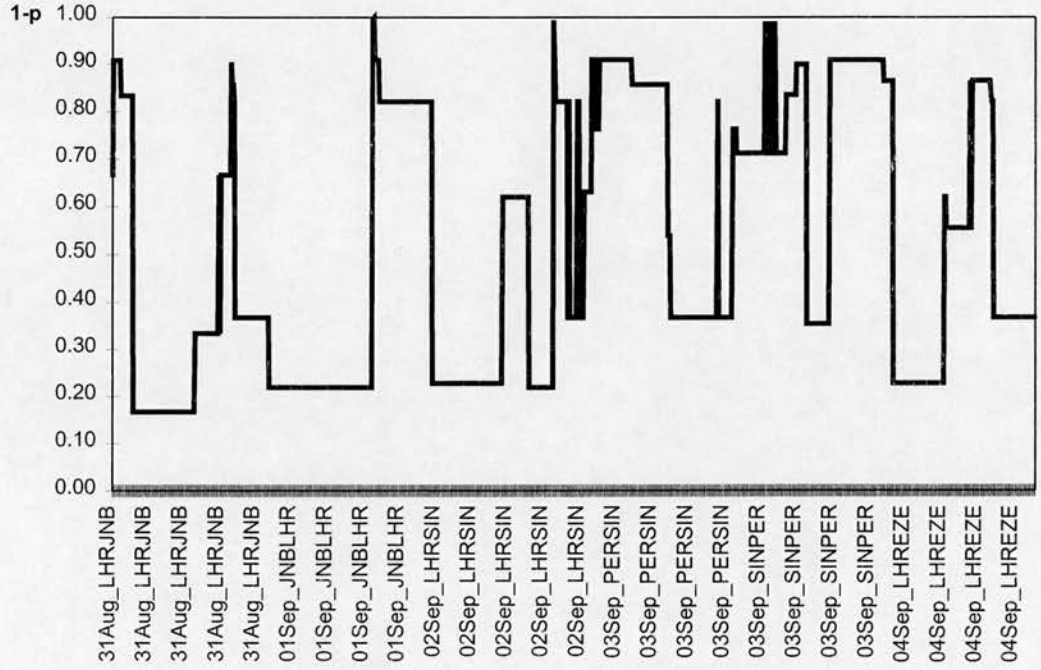


Figure 5.9 : Values of q *for an unseen test set of six take-offs.*

In Figure 5.9, the six take-off files have been concatenated together. The value plotted is 1 minus the normalised frequency of the engine state for the number-1 engine in each case from Figure 5.6(a), for the current value of the airframe state. Momentary high values of q *(i.e.* low values of p*),* indicate *transients* (see section 5.4.10). The presence of these can be

explained by inertia and by the fact that no explicit representation of the time dimension or of rates of change of the input variables has been used. A high value of q persisting for a longer period of time would be considered to be an indication of a novel situation (but not necessarily a fault).

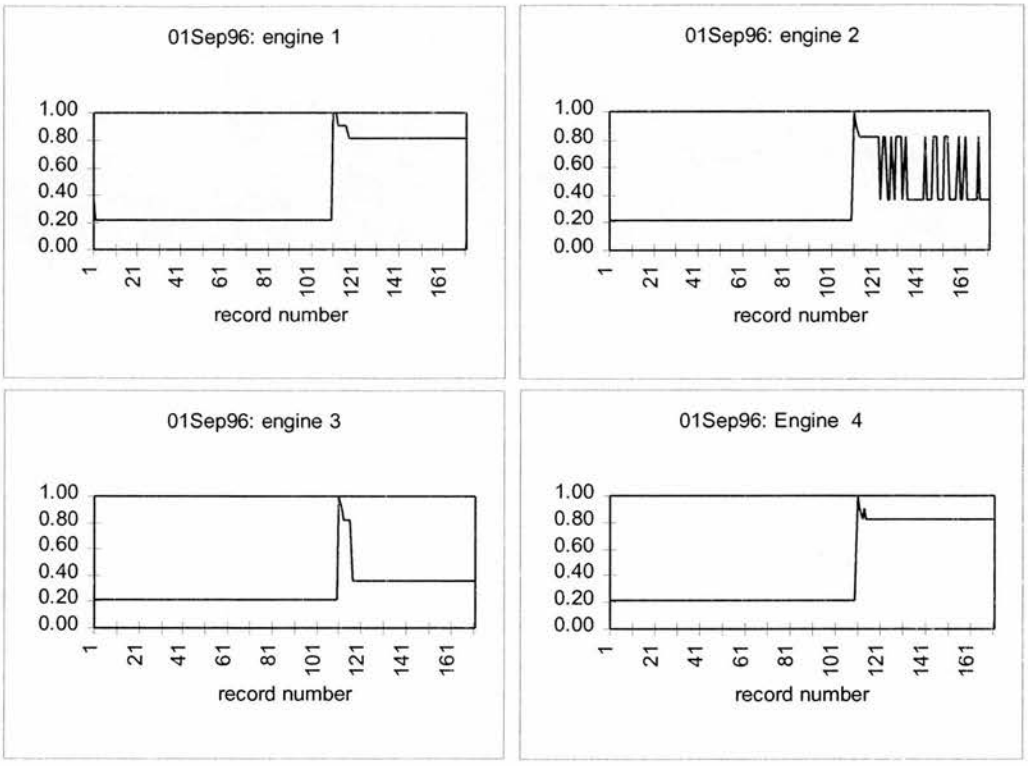


Figure 5.10: value of q for a take-off from Johannesburg

Figure 5.10 shows the value of q for a take-off from Johannesburg. A transient state is evident at around record 110. In records from about 118 onwards, the data points are at a similar distance from two states. Thus engines 1 and 4 show one of them, while engine 3 shows the other and the data for engine 2 appears to oscillate between the two. This is an artifact of the modelling method and one which it would appear to be difficult to control using this technique, without adding a clustering layer which identified states which may be treated similarly.

5.4.9 Results using distances

The results presented in Tables 5.5 and 5.6 and Figures 5.9 and 5.10 (the conditional frequencies of the engine data prototypes given the airframe data prototypes) can be considered to give the degree of novelty in the relationship between the engine and airframe parameters. This is one component of the three-level system described in section 5.1. In addition it is useful to examine the minimum distances in the SOMs, as these give a measure of the degree of novelty in the airframe and engine sets of inputs considered separately. Figures 5.11 and 5.12 show minimum SOM distances on the airframe and engine data SOMs, using the same training and test data used for Figure 5.9. The data presented here is *not* fault data, but data gathered in the course of normal flight operations. The occurrence of novelty is manifest because of the fact that a reasonably small set of data was used in training. However this set of data can be used effectively to demonstrate the principles involved. For a practically usable system it would be necessary to collect far more data in order to ensure that the events being flagged as "novel" corresponded to a high probability of finding a true fault.

In Figure 5.11, the distance between the current record and the nearest template node in the trained airframe SOM is plotted for a sequence of six takeoffs in the recall set. It is evident that one of the takeoffs has characteristics which are not represented in the training set. In fact, the exceptional data shown is from a takeoff from Johannesburg, where the airfield is situated at a high altitude (about 5000 feet) above sea level.

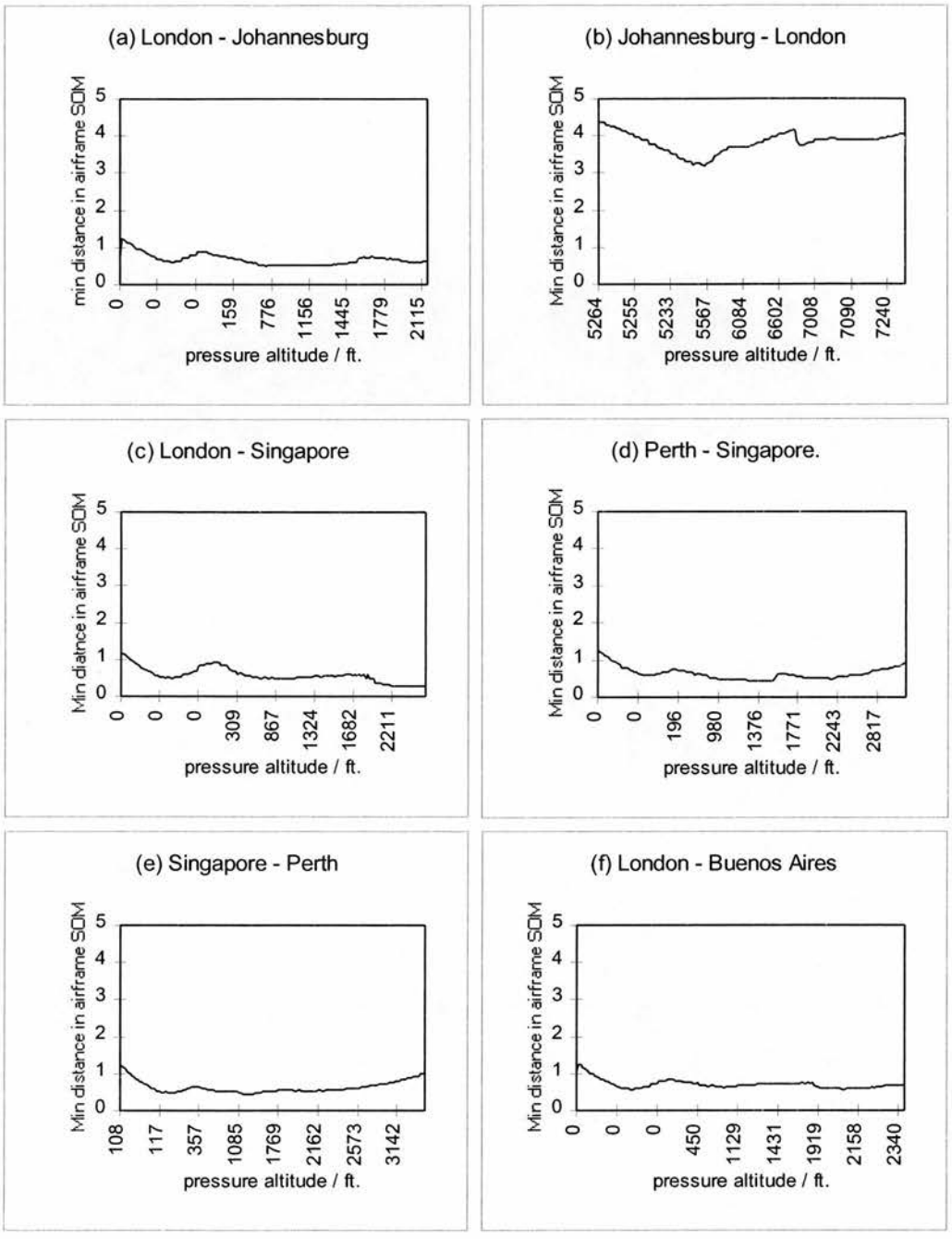


Figure 5.11: the distance between the current record and the nearest template node in the trained airframe SOM

In Figure 5.12, the distance between the current record and the nearest template node in the SOM trained for engine 1, is plotted for the same sequence of six takeoffs in the recall set. It will be noted from examination of Figure 5.12, that the *engine* behaviour for this sequence of data (the takeoff from Johannesburg) is indeed noticeably different according to the novelty detection capabilities of the network, but that the distances are not so extreme as for the airframe SOM. Looking at the conditional prototype frequencies (Figure 5.8), it is seen that the engine data does not show particular novelty *in the context of* the airframe data.

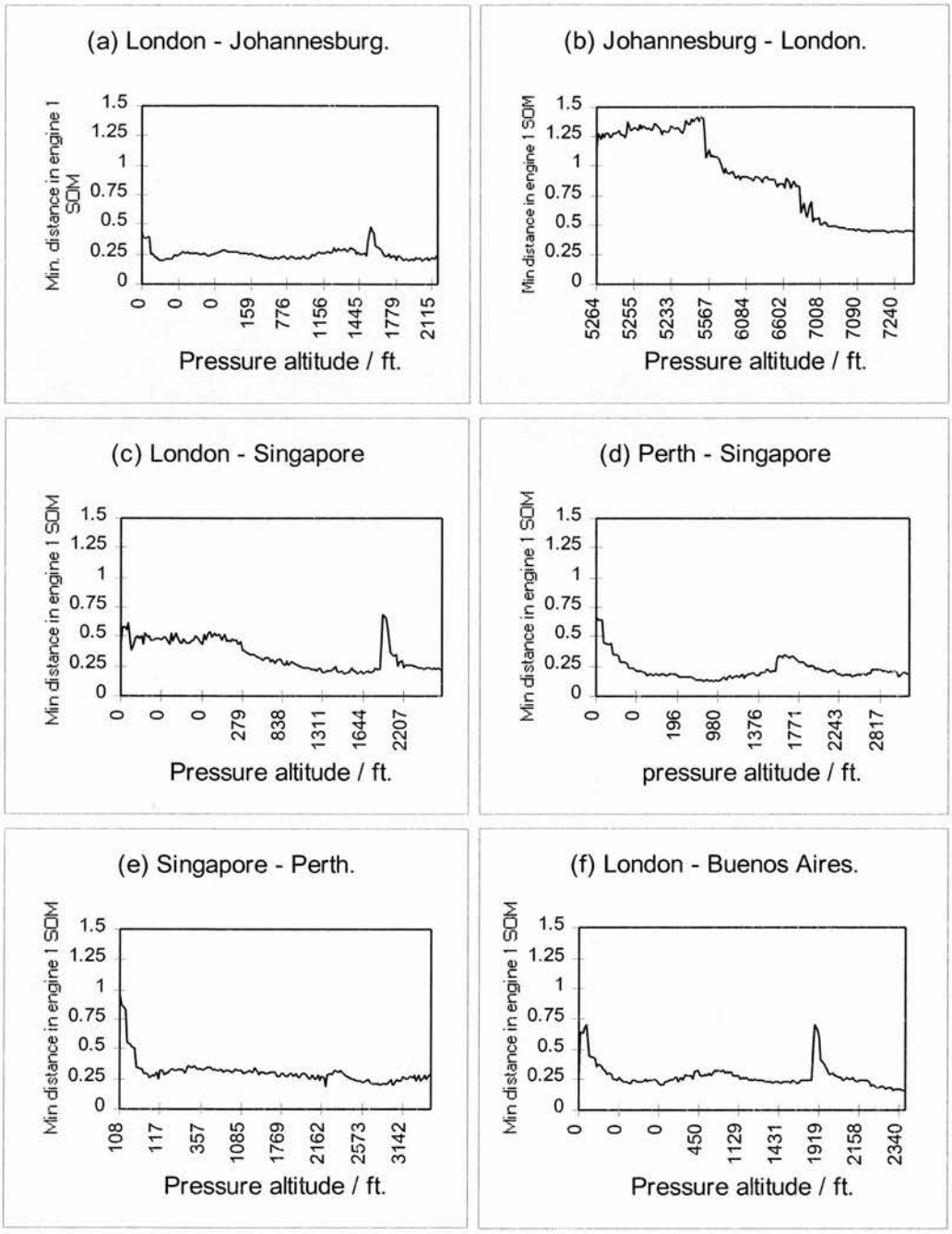


Figure 5.12: Distance between the current record and the nearest template node in the SOM trained for engine 1

5.4.10 Discussion of transients

In Figure 5.12 (c) and (f) a short-lived "peak" in the distance measure is evident, corresponding to an aircraft altitude of about 1900 feet above sea level. This is due to a change in throttle setting after the take-off flight phase, and the peak arises because of the time lag between the change in throttle setting and the change in the other engine parameters.

5.4.11 Effect of training strategy

In this section we consider the training the network on data from only one engine (out of four on an aircraft), and the meaning of the terms *generalisation* and *overfitting* in this context. (see section 5.3.4)

As will be seen from Table 5.10, the training data used for this study consists of one flight's data for each of seven aircraft (denoted (i) to (vii) in Table 5.11) and five flights' data for an eighth aircraft (denoted (viii) in Table 5.11).

Table 5.9 shows some noticeable differences in the results from passing the number-2-position engine data through the trained networks when compared to the results for the number-1-position engines in Table 5.8.

In assessing this method, one needs to demonstrate to what extent this discrepancy is due to genuine differences in performance between the two power units, and to what extent it is evidence of a kind of overfitting by the networks.

Another factor which needs to be taken into account here is that some of the states may be very close together and for practical purposes interchangeable. In other words, more than one SOM node may have come to represent the same real-world airframe or engine state. A prerequisite for the use of this method in practice is to impose some clustering of the SOM nodes into clusters representing "true" states. See section 5.5.4

5.5 Conclusions and further work

5.5.1 Interpretation of results

In this Chapter, the use of a combination of neural networks as a context-based novelty detection technique has been described, and an attempt made to answer questions about the extent of deviation of measured engine behaviour, as manifested in data contained in the flight data recorder (QAR) traces, not from an ideal, thermodynamically-motivated model, but from a model constructed from a set of examples representative of acceptable, good-condition, operating data. Although this could be achieved using linear techniques, where the interactions of the variables involved are linear, or where the non-linearities involved are explicitly known and modelled, for example using standard linearisation techniques, one of the advantages gained by using neural networks, is that the stage of explicit model identification is not necessary. The use of such a "knowledge-free" model also has its disadvantages, in the sense that extensive calibration on labelled data of known quality is required in order to assess the generalisation ability (and in particular the novelty detection reliability) of the resulting system. Also, there is a limit to the extent to which such a model can provide information about components of a system such as an engine. The identification of the types of variation in the data, as described in this Chapter, is only the first stage in seeking a generic data-driven condition monitoring method. The function of the neural network in this case is primarily one of dimensionality reduction and indirectly one of clustering. In this (rather restricted) setup, the airframe and engine trajectories are seen to be modelled adequately, and the mapping between the airframe states and the engine states is represented. The pair of SOMs and the state-pair representation captures the fact that the airframe and engine parameters are varying at different rates. This method also allows for the fact that there are natural variations in takeoff pattern to which we ultimately wish the monitoring system to be invariant. The "novelty" signals found in the data as detailed in Figures 5.9 to 5.13, are in fact well within the routine envelope of operation and in fact refer to different operating conditions. That these have been "detected" by the system is as a result of the fact that a limited amount of data has been used for training in this particular exercise. In a full-blown implementation where more complete training data had been used, these events would not be evident as novelty signals. However, this treatment of the data demonstrates the principle by which unusual events in environmental, airframe or engine data could be detected. Successful detection of outliers and unusual events is

dependent on achieving a training set which is as widely representative as possible of the total span of operating conditions and operational engine conditions. Because of the way in which the inputs have been set up in this exercise there is a certain amount of coupling between the two maps. This is by virtue of the fact that the "airframe" map contains as inputs two parameters which are strictly-speaking engine-related (Throttle Resolver Angle: TRA and Engine Pressure ratio: EPR) but which are treated for this purpose as "control settings" and therefore logically "inputs" rather than "outputs". It is possible to vary the composition of the sets of input variables to the two SOMs. An extension of the method might envisage an array of SOMs each with different selections of input variables. One has to be careful however not to increase the complexity of the representation beyond a point appropriate to the complexity of the problem itself.

5.5.2 Thoughts on the concept of "novelty detection"

When considering the task of detecting "novelty" in data it is necessary to give very careful consideration as to what exactly we mean by novelty; novelty with respect to *what?* The distinction between interpolation and extrapolation becomes a crucial and rather delicate one here. One requires the network to be reliably invariant to changes in the data which are routine or permissible and thus to minimise the false positive rate; whilst at the same time exhaustively spotting any true excursions from the desired envelope. A way to achieve this in practice, is iteratively to use the network to detect novelty in the data, and treat the novelty in one of the following ways:

(i) accept it as having a rightful place in the dataset (and retrain the network so that this is taken into account).
(ii) accept it as representing an identifiable effect or condition; label it as such and remove it from the training set, training a separate network (identical to, or different from, the first) to model it.
(iii) accept it as novelty which one requires the network to identify and flag up (leaving the network unchanged).
(iv) reject it as irrelevant, erroneous or confusing to the model; remove it from the dataset and filter it out from the recall data.

This requires human intervention to label and model the novel data, and as such overcomes some of the limitations of a unsupervised learning approach. One can envisage the iterative use of the above criteria as a way of refining a model over time to reflect with increasing

accuracy and confidence the nature of the underlying processes which are of interest. The above arguments can be applied to novelty detected within training set or on test or recall data.

5.5.3 Benefits of context-based novelty detection

The proposed method has advantages over a single SOM in this application, as there is wide variation in the operating conditions, which in turn affects the engine parameters. See Figure 5.13 for an example. Variations *between* engines, which we are seeking to examine, are comparatively small (see Figure 5.14). A single Kohonen network is most likely to capture differences in operating circumstances and the engine effects would then be masked out. The use of two Kohonen networks in parallel serves to overcome this problem, as the scale of events in the airframe data network and the engine data network can be different.

Although a "multiple regression" approach could be taken whereby an attempt is made to model the engine parameters directly as a function of the airframe and control parameters, it would be difficult to capture in a single model, the different situations which obtain in different phases of flight and ground activity.

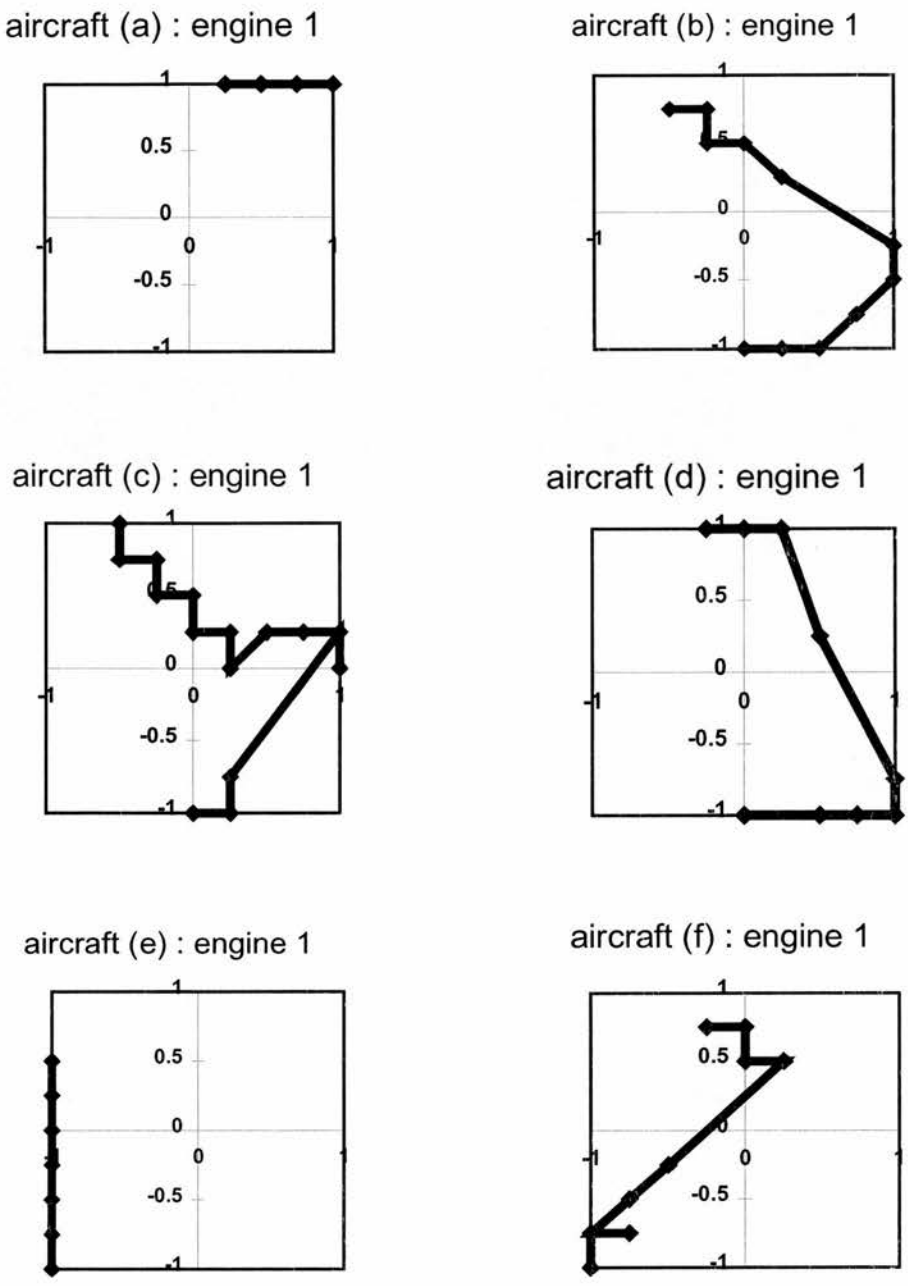


Figure 5.13: Variations between aircraft in trajectory through single SOM.

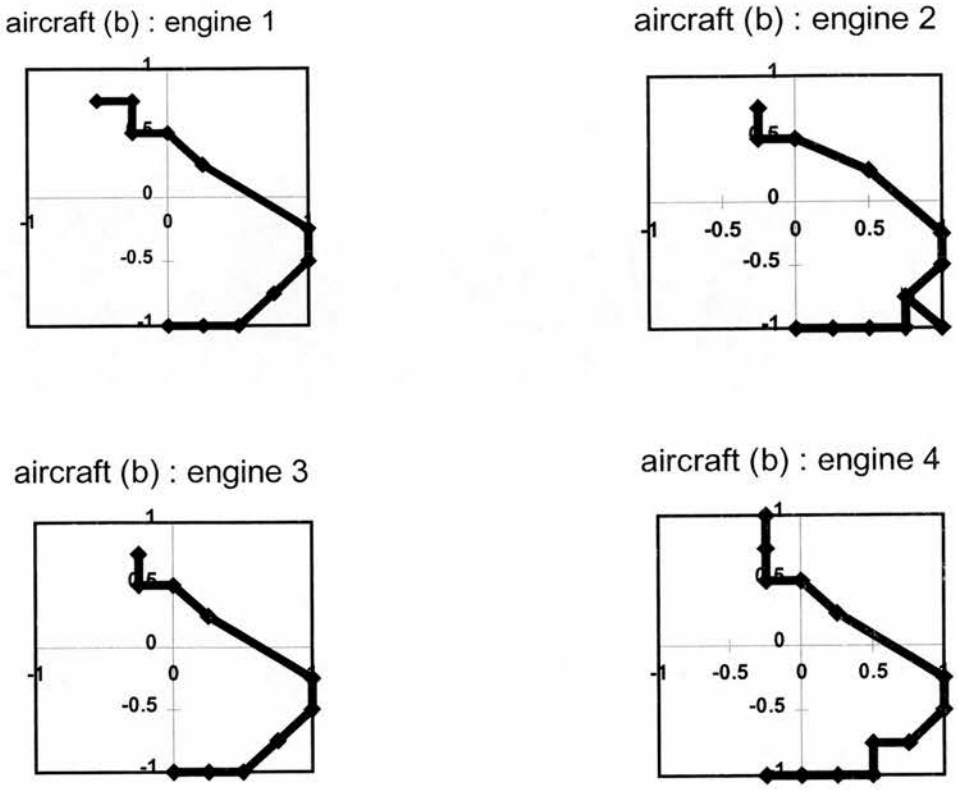


Figure 5.14: variations in trajectory in single SOM between engines on the same aircraft.

5.5.4 Weaknesses of the method

This method can at best only find instances of novelty, rather than identify faults. Without labelling, it is not known which of the "states" identified by the templates in the SOM are "good" and which, if any, are "bad". It is not known, without thorough experimentation, what "mapping" the SOMs have found, i.e. whether there are any discontinuities, etc. If a neural network is used to map from one SOM output onto the other, there is also the possibility of this network converging to a local optimum.

5.5.5 Further Work

This section describes some ways in which the method might be extended to overcome some of the deficiencies noted in previous sections.

Clustering of states

Although Kohonen's SOM produces a grid of clusters (here associated with "states"), it is often, as in this case, useful to combine these into meta-clusters or super-clusters, in this case clusters of states within which the identity of the individual state is not important from the point of view of the application. Murtagh (1995) discusses approaches to the clustering of SOM states using contiguity constrained clustering.

Labelling of state clusters

At some stage in the analysis it is desirable if possible to give definition to the clusters formed by the states and label them. The extent of this process will depend to some degree on the size of the respective SOMs compared with the effective number of states. In the above case the size of the SOMs and the number of states seem to be well-matched, but one could have a case where many SOM nodes mapped onto a single effective state (for either the input or output parameter sets). One needs to consider the question of where the labelling information is stored in the model and how it is represented and used. The SOM grids could be supplemented by a clustering stage where the SOM templates are grouped into clusters which are labelled by the engineer. These could be represented using a lookup table. The "third mapping" referred to above would then be between clusters of nodes in the airframe grid and clusters of nodes in the engine grid.

Use of raw distances

Rather than applying the winner-take-all rule on the trained SOMs to form the inputs and outputs for the third mapping, one could proceed to map from the "raw" distance outputs of the airframe network to the "raw" distance outputs of the engine network. One would then be dealing with a mapping from a distribution of distances to a distribution of distances (see Figure 5.7). This can be visualised effectively by using a colour coding scheme to represent the distance. It is then easy to visualise contours and to gain a rapid evaluation of the extent of novelty for a particular record.

Methods using whole trajectory

An alternative method of visualisation is to plot the whole trajectory through the SOM for a take-off or a whole flight. A disadvantage of this method is that the distance in each case is not readily visualisable. However this method in combination with the raw distance method detailed above provides a more powerful visualisation scheme.

Use of Generative Topographic Map (GTM)

Given that the Kohonen network does not form a probability density model, the use of a Generative Topographic Mapping approach may give a better basis upon which to discern the reasons for discrepancies noticed in the SOMs between different engines, as the GTM framework can be used to show a measure of how much distortion in the 2-d grid is required to accommodate a particular data set.

6 Concluding remarks: Real-world use of neural networks

The studies detailed in this work have shown that artificial neural network techniques perform at least comparably with conventional statistical techniques if a suitable network is chosen. However, there is overall an additional cost involved in using a neural network if one takes into account software costs and the time required for the iterated approach which is necessary. Therefore to be of real value, the neural network approach must provide additional insights and information.

6.1 Abstracted and "real-world" problems

In this thesis we have been examining the question of how well neural network techniques (particularly the well-tried and used techniques of back-propagation, radial basis function networks, and self-organising maps) scale to dealing with real-world problems. In the development of algorithms it is necessary to test new algorithms and methods using standard sets of data, preferably ones which are of sufficient size and complexity to demonstrate an algorithm's ability to solve hard problems, while being sufficiently simple for the mathematical and computational characteristics and effects of both the domain and the algorithm to be understood, and to enable the use of scientific method and experimental design. However, now that neural networks are in regular use for nontrivial problems in an industrial and commercial setting, it is appropriate to examine the issues concerned when implementing neural networks using real world data. In defining what we mean by real world data and situations, the following observations can be made. Firstly, there is a requirement to produce a cost-justified system which will be possible to implement, and usable in the workplace; moreover it is necessary for the results to be convincing enough to stakeholders, who are often non-technical in background and outlook, to justify what is seen as an additional risk associated with what is regarded as a non-standard approach. Secondly, with real-world applications, the practitioner often has little choice over the method of gathering data, and the opportunities for systematic experimental design are often very limited. In some situations, purely mathematical or scientific criteria for judging the success, or otherwise, of an approach, are over-ridden by human, professional and "cultural" factors. With many real-world problems, neural networks are chosen because no other

method has been successful in solving the problem. Sometimes, data will be newly available, and neural network techniques will be judged against simpler, linear, "traditional" statistical techniques which have not yet been tried in the particular domain area. In other cases a problem area may have a renewed business focus, and the question arises whether methods previously used to solve it are still the best. A third complication of real-world problems is that they are more "open". With an abstraction or subset of a problem, or in a laboratory situation, the system is "closed", in other words, the researcher has control, and can set boundaries and produce definitive results within the context of those boundaries. In most "real-world" cases solving the problem for a subset of the data or an abstraction of the task is very far from providing a satisfactory solution. Traditionally the field of operational research has found its strength in transforming messy, real-world domains into mathematical formulations for which crisp, well-quantified solutions can be found. The results of this type of analysis, of course, have to be applied with the understanding in mind of what assumptions and simplifications have been made in formulating the problem, and where the limits of applicability are. Neural networks are usually applied to problems which are closer to the real world, and less abstracted in nature. This fact, combined with the nature of neural network methods themselves, being non-linear and nonparametric, leads to some difficulties in stating traditionally understood confidence limits. It also reflects a difference in approach and method. Although the neural network approach tries to be closer to the irregularities of the real world, this in itself can open the method to the criticism that because it is based on historic data, models derived using neural can be unresponsive to step changes. A fourth issue, one which is often not mentioned in writings on the application of neural networks, is the effect of a problem of a hierarchical nature, and the level of hierarchy at which a solution is sought, or at which the data are available. From a simple consideration of statistical error theory, it is apparent that at a higher level of aggregation, many random errors will cancel and averaging will reduce the standard error due to random variations. The systematic component of the error is of course not reduced in this case. For any application with a given set of hierarchical data, there will be a limit to the level at which the data can be broken down, beyond which random variations or noise will outweigh predictable variations, or signal, in the data. In airline data there are often multiple hierarchical dimensions to the data, and answers are sought at the lowest possible level of aggregation.

6.2 Real-world data and data quality issues[1]

Having discussed some of the characteristics of real-world problem domains requiring some care in the application of neural networks, in this section we discuss issues relating to real-world data sources. A neural network (MLP for example) will give an "answer" whatever data is presented to it. Outside the training set, it has no "knowledge" to what extent this answer is right or makes sense. The processes of validation lie outside the domain of pure neural network modelling but still within the domain of delivering real world computational neural network systems. It is often the case that by using a neural network the analyst gains further insight into the structure and behaviour of the data (for example, by using a network to perform regression or clustering analysis of the data). In most cases this analysis could be done using other appropriate statistical data exploration methods. An iterative approach to neural network development, which is a necessity from the fact that neural networks are a data driven technique, brings benefits in that at each stage of the iteration there is a phase of sense-checking which will aid in elucidating data quality problems. It is in this way that a neural network approach can be of the most use in dealing with areas where data quality is suspect or unknown. In a bid to exploit newly collected data it is often the case that the neural network analyst is presented with very "raw" data. A neural network can be a very efficient way to determine if the data are adequate for the task prescribed, and can be used to determine the extent to which the relationships in the data are non-linear and as a guide to input selection.

[1] Many of the following points are separately discussed in detail for the specific issues raised by sensor faults in chapters 3 and 4.

6.3 User expectations and corporate environments

In neural network applications, user expectations and judgment of system performance are harsher than in the case of comparable "conventional" technologies, for example statistical pattern recognition. In a corporate environment unproven technology is likely to be closely examined as regards profitability and return on investment. Usually the problems being tackled by neural networks will be ones which have proven insoluble by other techniques. This can cause a misplaced hope that a new technique will necessarily be the answer. The reason the problems are hard may well not be mathematical complexity but rather randomness or lack of accurate information, or other procedural reasons.

6.4 Validation and verification of neural network systems

Now that experience of applying neural networks to complex, real-world domains is being amassed in a variety of industries, monitoring and control applications in particular will be under scrutiny from the point of view of dependability and ease of validation and verification.[2] Opinion is split concerning whether neural networks present an advantage or a disadvantage in this respect compared to other techniques for achieving the same ends. The "black box" reputation is hard to overcome, and the fact that inputs and outputs can be continuous make formal enumeration methods impossible or very difficult to apply. However, practicality imposes some limits on the demands which can be made, and an estimated (and statistically supported) probability level associated with aspects of reliability may be possible.

6.5 Probabilistic approaches to neural networks

A current direction of research in the application of neural networks and related methods is towards performing a more thorough characterisation of the statistical distribution of the data and the probabilistic description of the data generating processes. Such methods have a conceptual attractiveness, and are often referred to as "principled" methods. Examples are Bayesian approaches to neural networks (e.g. Mackay, 1991 *et seq.*, Wolpert, 1993b, 1994, Thodberg, 1993), Gaussian mixture model approaches, and generative topographic maps (Bishop, 1996). The hope is that such methods will give deeper insight into the phenomena

[2] See for example, [Proceedings of ERA conference on Dependable Neural Network Systems, 1996]

underlying the data, as well as overcoming the difficulties arising from the fact that simpler neural network methods produce merely a point estimate rather than a distribution as an output. They are, however, very computationally intensive, given that high-dimensional integrations are being performed numerically, and may in real-world cases often be intractable. There are still some theoretical and practical limitations to these approaches and most of them have not been extensively tried on real-world data. In instances where they have, improvements are often only marginal and it is questionable whether the improvement justifies the additional computational expense.

6.6 Conclusions

This thesis has explored two example application areas where solutions to real-world problems were sought using neural networks. In the first of these, the no-show forecasting problem, it was found that overall, the performance of neural network models was comparable with that of moving average and lookup table methods, as well as the tree-based CHAID method (which is similar in nature to many of the "rule induction" or "machine learning" methods). It was found that there was some variation between networks and between instances of the same network initialised with different random weights. Variation between different flight numbers, classes and days of week was also significant. Radial basis function networks showed a marginal improvement over multilayer perceptron networks. This can be understood in terms of radial basis function networks forming a more local model of the data, without the long-range effects which arise from the use of sigmoid activation functions in the multilayer perceptron framework.

In the engine condition monitoring task, reported in the second half of the thesis, the focus was on finding the best way of using neural networks to aid the development of condition monitoring techniques and systems, using flight data recorder and condition monitoring system data. The task was to perform simultaneous monitoring of a large number of measured parameters and to detect any departure from normality. In addition, the remote detection of fault modes from data measured within the jet engine formed an important sub-goal in this work. This task is related to the no-show task covered in the first part of the thesis, in that the objective is to train a network so that it can assess posterior class probabilities from training data consisting of a number of attributes (though in this case they are real numbers rather than binary values).

The condition monitoring problem, however can be viewed in terms of regression or classification. When couched as a regression problem, novelty detection can be achieved by analysis of the residuals. The difficulty at the heart of novelty detection, however, is to characterise all the legitimate variation in the parameters and at the same time readily detect any departure from a normal envelope of operation. This presents a considerable challenge to a learning system, in a stricter way than the usual requirement for "generalisation to an unseen data set". The results from Chapter Four show that both regression and classification neural networks can be used to provide complementary approaches to the problem of detecting a fault in an inaccessible part of the engine, and of using gathered data to perform off-line monitoring with respect to particular fault modes. Note however that even in combination the techniques do not always correctly identify fault situations and so can be used at best as secondary methods, to provide corroboration of evidence from other sources, or an indication of which engines to examine more thoroughly. These methods do not provide a replacement for the extensive visual examination of components or other health monitoring and condition monitoring methods based on physical analysis of residues from oil, etc., nor for the routine, off-line condition monitoring from flight data which is carried out using conventional software.

Chapter Five introduced a technique for performing novelty detection within a context, by using a combination of two self-organising maps, and characterising the mapping between the states and trajectories in the respective SOMs, using a lookup table or a feedforward network. This structure is an attempt to refine the ability of the network to discriminate between routine variation and a novel situation which may represent a fault mode. With the continuous data gathered in take-off, the principle was demonstrated by using the method to detect, not faults, but take-offs from airfields with unusual characteristics. The results of the work reported here indicate that to build a fault detection system with the appropriate robustness, it is necessary to use substantially more data in order to cover the space of variation of operating conditions. The task of selecting the training and test sets for this type of application, is a challenging area in itself. However the work reported in Chapter Five has demonstrated that it is in principle possible to detect unusual circumstances in this way.

Taken as a whole, the two problems examined in this thesis as application areas for artificial neural networks show a number of the characteristics of problems in a commercial or industrial context that distinguish them from the majority of benchmark data used in evaluating algorithms from a research perspective. As data-driven methods, neural networks did, in both the cases examined, form acceptable solutions which compared well to other applicable methods, and in being able to model nonlinear relationships, possessed an additional element of generality.

References

Abu-Mostafa, Y.S. (1990), "Learning from hints in neural networks", *Journal of Complexity* **6,** pp 192-198.

Abu-Mostafa, Y.S., (1995), "Financial applications of learning from hints" in: Tesauro, G., Touretzky, D.S., & Leen, T.K. (eds.) *Advances in Neural Information Processing Systems* **7**, MIT Press.

Adriaans, P., & Zantinge, D., (1996) *Data Mining* (Addison Wesley).

Atkinson, R.M, & Woollons, D.J. (1996), "A neural network approach to fault diagnosis in electro-hydraulic systems", in: *Condition Monitoring and Diagnostic Engineering Management: Proceedings of COMADEM 96*, pp125-134, 1996.

Austin, S-J., (1993) "Requirements for, and Sources of, Travel Data". British Airways. Unpublished report.

Aznar Fernandez-Montesinos, M (1994), "Intelligent Aircraft Engine Condition Monitoring", MSc Thesis, Delft University of Technology.

Aznar Fernandez-Montesinos, M., Janssens, P. and Vingerhoeds, R.A.(1994), "Enhancing Aircraft Engine Condition Monitoring", in *IFAC Workshop, "Safety, Reliability and Applications of Intelligent Control Techniques"*, Hong Kong, Dec 1994.

Azzam, H. and Hazell, J. (1996), "Appraisal of advanced diagnostic strategies using a suite of helicopter fault simulations", in *Condition Monitoring and Diagnostic Engineering Management (Proceedings of COMADEM 96)* Sheffield Academic Press, 1996, pp21-31.

Berthold, M.R., and Diamond, J., (1995) "Boosting the performance of RBF networks with Dynamic Decay Adjustment" in: Tesauro, G., Touretzky, D.S., & Leen, T.K. (eds.) *Advances in Neural Information Processing Systems* **7**, MIT Press.

Bishop, C.M. (1994), "Novelty Detection and neural network validation", *IEE Proceedings: Vision, Image and Signal Processing* 141(4), 217-222

Bishop, C.M. (1995), *Neural Networks for Pattern Recognition*, Oxford.

Bishop, C.M., Svensen, M. and Williams, C.K.I., (1996) "GTM: The Generative Topographic Mapping." Technical Report NCRG 96/015, Aston University.

Box, G.E.P & G.M. Jenkins, *Time Series Analysis, Forecasting and Control*, Holden-Day, 1970.

Bridle, J.S., (1990) "Probabilistic Interpretation of Feedforward Classification Network Outputs with Relationships to Statistical Pattern Recognition", in: Fogelman-Soulie, F., & Herault, J (eds.) *Neuro-computing: Algorithms, Architectures and Applications* NATO ASI Series, Springer-Verlag.

Broomhead, D.S., and Lowe, D., (1988) "Multivariable functional interpolation and adaptive networks" *Complex Systems* **2**, 321-355.

Bryson, A.E (Jr) and Yu Chi Ho (1969) *Applied Optimal Control,* Blaisdel.

Buntine, W., and Weigend, A.S.,(1991) "Bayesian Back-Propagation", *Complex Systems* **5,** 603-643.

Chappell G.J and J.G. Taylor (1993) The Temporal Kohonen Map, *Neural Networks* **6,** pp 441-445.

Cottrell, M., Fort, J.C., & Pages, G.,(1994) "Two or three things that we know about the Kohonen algorithm" Manuscript, University of Paris.
[in *Neuroprose* archive; ftp://archive.cis.ohio-state.edu/pub/neuroprose]

Cowley, P.H, (1995) [*Personal communication.*] Rolls-Royce Applied Science Laboratory, Derby, UK.

Cox, T.F., and Cox, M.A.A., (1994), *Multidimensional Scaling*, Chapman & Hall.

Craven, M.W., and Shavlik, J.W., (1996) Extracting Tree-Structured Representations of Trained Networks, in: Touretzky, D.S., Mozer, M.C., & Hasselmo, M.E., *Advances in Neural Information Processing Systems* **8.** MIT Press.

Cumming, S.N. (1993), "Neural Networks for Monitoring of Engine Condition Data", Neural Computing and Applications, **1**, 1, pp 96-102.

Daubechies, I., (1992), *Ten Lectures on Wavelets*, Society for Industrial Mathematics.

de Sa, V.R., "Learning Classification with Unlabelled Data" in: Cowan et al., *Advances in Neural Information Processing Systems* **6**, pp 112-119.

Dempster, A.P, Laird, N.M., and Rubin, D.B., (1977) "Maximum Likelihood from Incomplete Data via the EM algorithm", *Journal of the Royal Statistical Society* B , vol 39, pp 1-38.

Dodd, N. (1990) "Intensive Care Ward Monitoring and Default Training", *Proceedings of INNC, Paris, 1990*, Vol 1 . Kluwer Academic.

Durbin, R and Willshaw, D.J., (1987) "An analogue approach to the travelling salesman problem using an elastic net method", *Nature* **326**, 689-691.

Erwin, E., Obermayer, K., & Schulten, K., (1992) "Self-organizing maps: ordering, convergence properties and energy functions." *Biological Cybernetics* **67,** 47-55.

Fahlman, S.C., and Lebiere, C.,(1990) "The Cascade-Correlation Learning Architecture", in Touretzky, D.S (ed) *Advances in Neural Information Processing Systems* **2**. Morgan Kaufmann.

Girosi, F., Jones, M. & Poggio, T. (1994), "Priors, Stabilizers and Basis Functions: from regularization to radial, tensor and additive splines" MIT AI Memo no 1430, March 1994.

Goodhill, G.J. and T. J. Sejnowski (1996). "Quantifying neighbourhood preservation in topographic mappings" *Proceedings of the 3rd Joint Symposium on Neural Computation,* University of California, San Diego and California Institute of Technology, **6,** 61-82, Pasadena, CA.

Goodhill, G.J., Simmen, M. and D.J.Willshaw (1995), "An evaluation of the use of multidimensional scaling for understanding brain connectivity" *Phil. Trans. Roy. Soc. B,* **348**, 265-280.

Hansen, L.K, Lautrup, B., *et al.* (1994) "Extremely ill-posed learning" Manuscript, CONNECT, Niels Bohr Institute, Copenhagen.

Harrison, P.J and Stevens, C.F. (1976), "*Bayesian Forecasting*", Royal Statistical Scoiety 1976 (JRSS Nr 3 1976 p205 - 228).

Harrison, R.F., and Marshall, S.J., "The Multi-layer Perceptron as an Aid to the Early Diagnosis of Myocardial Infarction". Research Report no 395, University of Sheffield.

Hawkins, D. & G.V.Kass (1982), "Automatic Interaction Detection" in Hawkins, D. *Topics in Advanced Multivariate Analysis* (CUP 1982).

Hecht-Nielsen, R. (1990), *Neurocomputing.* Addison-Wesley

Hertz, J., Krogh, A., & Palmer, R.G., (1991*) Introduction to the Theory of Neural Computation*, Addison-Wesley.

Hornik, K., Stinchcombe, M., and White, H.(1988), "Multilayer feedforward networks are universal approximators", Technical Report, Dept of Economics, University of California at San Diego.

Horswell, R.L., and Looney, S.W., "A comparison of tests .. based on measures of multivariate skewness and kurtosis", *Journal of Statistical Computing Simulations*, **42,** 21-38.

Hutchison & Stephens (1987), The Airline Marketing Tactician, Proc 1st ICNN 1987. IEEE.

Infield, R., (1989) "PUMA Users' guide". Unpublished document. British Airways PLC.

Jepson, B., Collins, A., and Evans, A.(1993), "A procedure to Determine the Confidence Limits for Artificial Neural Network Predictions" (University of Portsmouth). Oral presentation at NCAF Conference, Heidelberg, April 1993.

Jolliffe, I.T. (1986), *Principal Components Analysis*, New York: Springer

Jordan, M.I., and Jacobs, R.A.(1994), "Hierarchical Mixtures of Experts and the EM algorithm" *Neural Computation* **6**, 181-214 (preprint - MIT AI Memo no 1440, Aug 1993).

Kalman, R.E (1963) "New methods in Wiener filtering theory" in: *Proceedings of the First Symposium on Engineering Applications of Random Function Theory and Probability* (J.L. Bogdanoff & F. Kozin (eds.)), New York: Wiley.

Kendall, M.G., and Stuart, A.,(1967) *The Advanced Theory of Statistics*, vol 2. 2nd edition. Griffin.

Kerry, P., Page, M. and Ward, M.J., *COMPASS Analysis Guide for RB211-524G-EMMU Engine*. Rolls-Royce, 1988-1991.

Keymeulen, D., and M. De Gerlache, (1994), "Comparison Training for a Rescheduling Problem in Neural Networks" In Cowan, J., Tesauro,G. and J. Alspector (eds.) *Advances in Neural Information Processing Systems 6,* San Mateo, CA, Morgan Kaufmann.

King, R.D., Feng, C., & Sutherland, A., (1995) "STATLOG: Comparison of Classification Algorithms on Large Real-World Problems" *Applied Artificial Intelligence*, **9** (3), Taylor & Francis.

Kohonen, T. (1982) "Self-organised formation of topologically correct feature maps". *Biological Cybernetics*, **43**, 59-69.

Kohonen, T. (1984,1988) "*Self-organization and associative memory*", Springer, Berlin.

Kohonen, T. (1995) *Self-organizing Maps*. Springer.

Kramer, M.A.(1991) "Nonlinear Principal Component Analysis Using Autoassociative Neural Networks", *AIChe Journal* **37** (2) , 233-243.

Lisboa, P.J.G. (ed.) , *Neural Networks: Current Applications*. Chapman & Hall 1992.

Lowe, D. (1993a), "A Brief Review of the Theory of Radial Basis Function Networks", Manuscript, Aston University.

Lowe, D. (1993b), "Novel topographic nonlinear feature extraction..." Proc. IEE Neural Network Conference 1993, pp95-99.

Lowe, D. & Tipping, M. (1996), "Feed-forward neural networks and topographic mappings for Exploratory Data Analysis", *Neural Computing and Applications*, 4, 83-95.

Luttrell, S.P, (1989a) "Self-Organisation: a derivation from first principles of a class of learning algorithms" *Proc. 3rd IEEE Int. Joint Conf. on Neural Networks, Washington DC, 2,* pp495-498.

Luttrell, S.P, (1989b) "Hierarchical Self-Organising Networks", *Proc. 1st IEE Conf. on Artificial Neural Networks, London*, pp.2-6.

Luttrell, S.P., (1990) Derivation of a class of training algorithms. *IEEE Trans. Neural Networks* **1**, 229-232.

Luttrell, S.P., (1994) A Bayesian Analysis of Self-Organising Maps. *Neural Computation*, **6**, 767-794.

McCullagh, P. and Nelder, J.A, (1989) *Generalised Linear Models*, London. Chapman and Hall.

McGrath,P., (1995) "Forecasting airline passenger noshow rates using radial basis function networks", MSc Thesis, Brunel University, Uxbridge, UK.

Mackay, D.J.C, (1991)"Bayesian Methods for Adaptive Models", PhD thesis, California Institute of Technology.

Mackay, D.J.C, (1992a) "Bayesian interpolation" *Neural Computation* **4(3)**, 415-447.

Mackay, D.J.C, (1992b) "A Practical Bayesian framework for backpropagation networks" *Neural Computation* **4(3)**, 448-472.

Mackay, D.J.C, (1992c) "The Evidence Framework applied to classification networks" *Neural Computation* **4(5)**, 698-714.

Mackay, D.J.C, (1994) "Bayesian nonlinear modelling for the prediction competition", in: *ASHRAE Transactions*, V 100, Pt 2.

Mackay, D.J.C, (1995) "Hyperparameters: optimize or integrate out?", in*: Maximum Entropy and Bayesian methods*, Santa Barbara 1993, ed. G.Heidbreder, Dordrecht. Kluwer .

Mackay, D.J.C, (1995) "Probable networks and plausible predictions - a review of practical Bayesian methods for supervised neural networks" Manuscript, Cavendish Laboratory, University of Cambridge.

McLachlan, G., & Basford, K., (1988), *Mixture Models: Inference and Application to Clustering*, Marcel Dekker.

Makridakis, S., and Hibon, M. (1979), Accuracy of Forecasting: An Empirical Investigation". *J. Royal Statistical Soc. A* **142**, Part 2, pp 97-145.

Mezard, M, and Nadal, J.P.,(1989) "Learning in feedforward layered networks: The tiling algorithm", *Journal of Physics 'A'*, **22**, 2191-2203.

Michie, D., Spiegelhalter, D.J., and Taylor, C.C. (eds.) (1996*) Machine Learning, Neural and Statistical Classification*. Ellis Horwood.

Mitchison, G. (1995) "A type of duality between self-organising maps and minimal wiring", *Neural Computation*, **7**, 25-35.

Morrison, N.W.A., (1990), "Using neural networks to forecast airline passenger numbers" MSc Thesis, Brunel University, Uxbridge, UK.

Moody, M., (1992) "The effective number of parameters" (*in Advances in Neural Information Processing Systems* **4**, p847 *et seq.*).

Moody, J., and Darken, C.(1989) "Fast Learning in Networks of Locally-tuned Processing units", *Neural Computation* **1**, 281-294.

Murray, A., & J. Penman, (1996) "Wavelets as an Alternative to the FFT in Condition Monitoring Schemes using ANNs", in Rao *et al., Condition Monitoring and Diagnostic Engineering Management (Proceedings of COMADEM 96)* Sheffield Academic Press, pp 177-186.

Nairac, A., Corbett-Clark, T., Ripley, R., Townsend, N & Tarassenko, L. (1997,a) "Choosing an Appropriate Model for Novelty Detection", *Proc 5th IEE Conference on Artificial Neural Networks, Cambridge*, 117-122.

Nairac, A., Townsend, N., Carr, R., King, S., Cowley, P. & Tarassenko, L., (1997,b) "A System for the Analysis of Jet Engine Vibration Data" *Journal of Computer Aided Engineering*. In press.

Neal, R.M.(1991), "Bayesian Mixture Modelling by Monte Carlo Simulation" Technical Report CRG-TR-91-2, Dept of Computer Science, University of Toronto.

Neal, R.M. (1995), "Bayesian Learning for Neural Networks", PhD thesis, Dept of Computer Science, University of Toronto.

Neuneier, R., Hergert, F., Finnoff, W., Ormoneit, D. (1994), "Estimation of Conditional Densities: A Comparison of Neural Network Approaches" *Proc. ICANN 94*, Sorrento, Italy.

NeuralWare (1993) "Neural Computing" & "NeuralWorks Reference Guide". Pittsburgh, PA.

Oja, E. (1989), "Neural Networks, Principal Components and Subspaces" *International Journal of Neural Systems*, **1**, (1), pp 61-68.

Omohundro, S., "Efficient algorithms with neural network behaviour" *Complex Systems* **1**, 273-347.

Orr, M.J.L.,(1996) "Introduction to Radial Basis Function Networks". Technical report, Centre for Cognitive Science, University of Edinburgh.

Patel, V.C & V. Kadirkamanathan (1996), "Adaptive self-learning fault detection system for gas turbine engines" in Rao et al, *Condition Monitoring and Diagnostic Engineering Management (Proceedings of COMADEM 96)* Sheffield Academic Press, pp 937-946

Perrone, M.P., and Cooper, L. (1993), "When networks disagree: Ensemble Methods for Hybrid Neural Networks" in *Neural Networks for Speech and Image Processing*, Mammone, R.J. (ed.), Chapman Hall.

Pilkington, J. (1995) [Personal communication.] Scientific Computers Ltd, Crawley, UK.

Platt, J.C,, (1991a) "Learning by combining memorization and gradient descent" in Lippman et al (ed.), *Advances in Neural Information Processing Systems*, Vol. 3, Morgan Kaufman 1991.

Platt, J.C., (1991b) "A Resource Allocating Network for Function Interpolation" *Neural Computation* **3**, 213-225 .

Plutowski, M., Cottrell, G., & White, H. (1994), "Learning Mackey-Glass from 25 examples, plus or minus 2", in: Cowan, J.D., Tesauro, G. and Alspector, J. (eds) *Advances in Neural Information Processing Systems* **6**, pp 1135-1142 (published in Volume 6 under "Addenda to NIPS5")

Poggio, T., and Girosi, F., (1990) "Networks for approximation and learning." *Proceedings of the IEEE* **78 (9)**, 1481-1497.

Prechelt, L. (1994) "A Study of Experimental Evaluations of Neural Network Learning Algorithms: Current Research Practice" Technical Report 1994-19, Inst. fuer Programmstrukturen und Datenorganisation, Universitaet Karlsruhe

Quinlan, J., (1993) *C4.5: Programs for machine learning.* Morgan Kaufmann.

Richard , M.D, & Lippmann, R.P. (1991) "Neural network classifiers estimate Bayesian *a posteriori* probabilities" *Neural Computation* **3(4),** 461-483.

Ripley, B.D. (1993) "Statistical Aspects of Neural Networks" in Barndorff-Nielsen, O.E., Jensen, J.L., & Kendall, W.S. (eds.) *Networks and Chaos - Statistical and Probabilistic Aspects, pp40-123.* Chapman & Hall.

Ripley, B.D.(1996), *"Pattern Recognition and Neural Networks"*, Cambridge University Press, 1996

Robbins, H., & Monro, S., 1951, "A Stochastic Approximation Method", *Annals of Mathematical Statistics* **22**, 400-407.

Rock, D., Malkoff, D. and R.Stewart. (1993), "AI and Aircraft Health Monitoring", *AI Expert*, February 1993.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J., (1986) "Learning internal representations by error propagation", in Rumelhart, D.E, McClelland, J.L., *et al* (eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Volume 1, MIT Press.

Sammon, J.W (Jr) (1969) "A non-linear mapping for data structure analysis". *IEEE Transactions on Computers* **18**, 401-409

Satchwell, C. (1993) "Neural-Statistical Concepts in Data Analysis: - Prediction" Technical Report, Neural Statistics, PO box 31, Southampton.

Simmen, M. (1992) "Neural Network Optimization" PhD thesis, University of Edinburgh.

Skitt, P.J.C & Witcomb, R.C.,(1990) "The analysis of the acoustic emission of jet aircraft engines using artificial neural networks" *Condition Monitoring and Diagnostic Technology* **1**, (Jul 1990) pp7-12.

Statistical Innovations Inc. (1992) "The SI-CHAID SAS Procedure" *The SI-CHAID Segmentation System, Users' manual for SAS release 6.07.*

Tarassenko, L. *et al* (1996) "Novelty Detection", *NCAF Symposium,* Edinburgh, 17-18 Sep 1996.

Thodberg, H.H.(1993) "Ace of Bayes: application of neural networks with pruning" Manuscript 1132 E, Danish Meat Research Institute.

Thodberg, H.H (1994) "Bayesian Backpropagation in Action: Pruning, Committees, Error Bars, and an Application to Spectroscopy" in: Cowan, J.D., Tesauro, G. and Alspector, J. (eds) *Advances in Neural Information Processing Systems* **6.** Morgan Kaufmann.

Thodberg, H.H (1995)., "A review of Bayesian neural networks with an application to near infrared spectroscopy. Revised version of Manuscript 1132 E (part 1). 6 Feb 1995. Danish Meat Research Institute.

Tibshirani, R. and Hinton, G. (1994) " 'Coaching' Variables for Regressioin and Classification" Manuscript, University of Toronto.

Titterington, D.M., Smith, A.F.M., &Makov, U.E., (1985), *Statistical Analysis of Finite Mixture Distributions* New york: John Wiley.

Tsung, F., and Cottrell, G., (1995) "Phase Space Learning" in: Tesauro, G., Tourezky, D., Leen, T., (eds.) *Advances in Neural Information Processing Systems* **7**. MIT Press.

Wahba, G., Wang, Y., *et al.* (1994) "Structured Machine learning For 'Soft' Classification with Smoothing Spline ANOVA and Stacked Tuning, Testing and Evaluation." *in* Cowan, J., Tesauro, G., and Alspector, J., *Advances in Neural Information Processing Systems* **6.** Morgan Kaufmann.

Waterhouse, S.R., and Robinson, A.J.(1994), "Classification using hierarchical mixtures of experts" in *Proc. IEEE Workshop on Neural Networks for Signal Processing* **IV**, pp177-186.

Watrous, R, Towell, G. and M.S.Glassman (1993), "Synthesize, Optimize, Analyze, Repeat (SOAR): Application of Neural Network Tools to ECG Patient Monitoring", *Proc. 1993 Symposium on Nonlinear Theory and its Applications*, Honolulu. pp565-570.

Werbos, P.J. (1974) Beyond Regression: new tools for prediction and analysis in the behavioural sciences. PhD thesis, Harvard University.

White, H., (1990) "Connectionist nonparametric regression: multilayer feedforward networks can learn arbitrary mappings" *Neural Networks* **3** (5), 535-549.

Witcomb, R.C, Skitt,P.J.C and M.J.Down (1990), "Automating the Collection and Organisation of Reference Spectra for use in Vibration Monitoring", Technical Report, Smiths Industries Aerospace and Defence Systems, Cheltenham UK.

Witcomb, R.C., Skitt, P.J.C. and Hewitt, P.D. (1989), "The adaptive acoustic monitoring of aircraft engines" COMADEM 1989, p194-198.

Wolpert, D.H.,(1992), "On the connection between in-sample testing and generalization error". *Complex Systems* **6**, 47-94.

Wolpert, D.H.,(1992/3), "On Overfitting Avoidance as Bias", Technical Reports SFI-TR 92-03-5001. and SFI-TR-93-016. Santa Fe Institute, NM.

Wolpert, D.H., (1993a), "Stacked Generalization", Technical Report LA-UR-90-3460, Santa Fe Institute, NM.

Wolpert, D.H., (1993b), "On the Use of Evidence in Neural Networks", in: *Advances in Neural Information Processing Systems* **5.** Giles, *et al.* (eds.) Morgan Kaufmann.

Wolpert, D.H, (1994) "Reconciling Bayesian and non-Bayesian analysis" Manuscript, Santa Fe Institute, NM.