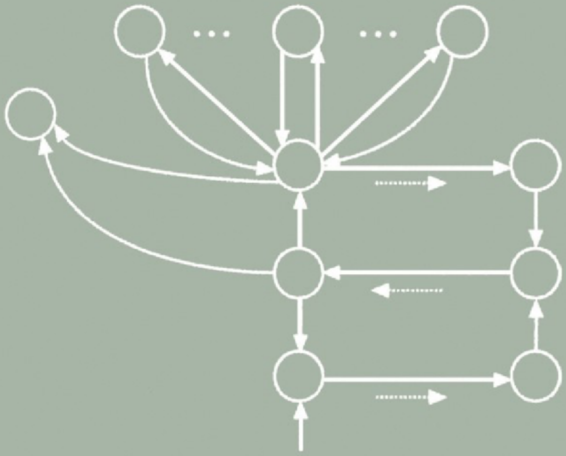
Xue Z. Wang

DATA MINING AND KNOWLEDGE DISCOVERY FOR PROCESS MONITORING AND CONTROL



Advances in Industrial Control

Springer

Advances in Industrial Control

Springer-Verlag London Ltd.

Xue Z. Wang, BEng, MSc, PhD
Department of Chemical Engineering
University of Leeds
Leeds LS2 9JT
UK

British Library Cataloguing in Publication Data
Wang, Xue Z.
Data mining and knowledge discovery for process monitoring and control. - (Advances in industrial control)
1.Process control - Data processing 2.Data mining
I.Title II.McGreavy, C..
629.8'9
ISBN 978-1-4471-1137-5

Library of Congress Cataloging-in-Publication Data
Wang, X. Z. (Xue Zhang), 1963-
Data mining and knowledge discovery for process monitoring and Control / Xue Z. Wang.
p. cm. -- (Advances in industrial control)
Includes bibliographical references and index.
ISBN 978-1-4471-1137-5 ISBN 978-1-4471-0421-6 (eBook)
DOI 10.1007/978-1-4471-0421-6
1. Process control-- Data processing. 2. Data mining. 3. Knowledge acquisition (Expert systems) I. Title. II. Series.
TS156.8 .W37 1999
670.42'7563--dc21

Advances in Industrial Control series ISSN 1430-9491
ISBN 978-1-4471-1137-5

9 8 7 6 5 4 3 2 1

springer.com

Xue Z. Wang

Data Mining and Knowledge Discovery for Process Monitoring and Control

With 121 Figures

Springer

Xue Z. Wang, BEng, MSc, PhD
Department of Chemical Engineering, University of Leeds, Leeds. LS2 9JT.

ISBN 978-1-4471-1137-5

British Library Cataloguing in Publication Data
Wang, Xue Z.
Data mining and knowledge discovery for process monitoring and control. - (Advances in industrial control)
1.Process control - Data processing 2.Data mining
I.Title II.McGreavy, C..
629.8'9
ISBN 978-1-4471-1137-5

Library of Congress Cataloging-in-Publication Data
Wang, X. Z. (Xue Zhang), 1963-
Data mining and knowledge discovery for process monitoring and Control / Xue Z. Wang.
p. cm. -- (Advances in industrial control)
Includes bibliographical references and index.
ISBN 978-1-4471-1137-5 ISBN 978-1-4471-0421-6 (eBook)
DOI 10.1007/978-1-4471-0421-6
1. Process control-- Data processing. 2. Data mining. 3. Knowledg acquisition (Expert systems) I. Title. II. Series.
TS156.8 .W37 1999
670.42'7563--dc21

Typesetting: Camera ready by author

69/3830-543210 Printed on acid-free paper SPIN 10703341

Advances in Industrial Control

This book is dedicated to
my wife Yanmin for her love
and to
Professor Colin McGreavy, an
influential figure in process control
who died on June 23rd 1999

SERIES EDITORS' FOREWORD

The series *Advances in Industrial Control* aims to report and encourage technology transfer in control engineering. The rapid development of control technology has an impact in all areas of the control discipline. New theory, new controllers, actuators, sensors, new industrial processes, computer methods, new applications, new philosophies..., new challenges. Much of this development work resides in industrial reports, feasibility study papers and the reports of advanced collaborative projects. The series offers an opportunity for researchers to present an extended exposition of such new work in all aspects of industrial control for wider and rapid dissemination.

How is information technology having an impact on the control, monitoring and operation of large-scale industrial processes? This monograph by Xue Wang of Leeds University supplies an in-depth answer to this question. The text has the starting point that, traditionally, control engineers have concentrated on system dynamics, measurement selection, control structure determination and the choice of the controller algorithm to be used. This ignores the large quantity of data generated by modern plant sensing devices as a source of information for improved control and enhanced process monitoring. The difficulty in many cases is the sheer quantity of data arising and the problem of interrogating, analysing and interpreting this data. Solutions, some old... multivariate analysis, and some new... neural networks and fuzzy sets, are presented by the author along with illustrative examples usually based on real industrial processes and data. This very instructive monograph will be of interest to the practising industrial process engineer for its insights and to the academic control community for its industrial perspective.

M.J. Grimble and M.A. Johnson
Industrial Control Centre
Glasgow, Scotland, UK

PREFACE

Being able to collect and display to operators a large amount of information is regarded as one of the most important advances provided in distributed control (DCS) over earlier analogue and direct digital control systems. The data are used by plant operators and supervisors to develop an understanding of plant operations through interpretation and analysis. It is this understanding which can then be used to identify problems in current operations and find better operational regions which result in improved products or in operating efficiency.

It has long been recognised that the information collected by DCS systems tends to overwhelm operators and so makes it difficult to take quick and correct decisions, especially in critical occasions. For example, olefin plants typically have more than 5000 measurements to be monitored, with up to 600 trend diagrams. Clearly there is a need to develop methodologies and tools to automate data interpretation and analysis, and not simply rely on providing the operators large volumes of multivariate data. The role of the acquisition system should be to provide the operators with information, knowledge, assessment of states of the plant and guidance in how to make adjustments. Operators are more concerned with the current status of the process and possible future behaviour rather than the current values of individual variables.

Process monitoring tends to be conducted at two levels. Apart from immediate safe operation of the plant, there is also the need to deal with the long term performance which has been the responsibility of supervisors and engineers. The databases created by automatic data logging provide potentially useful sources of insight for engineers and supervisors to identify causes of poor performance and opportunities for improvement. Up to now such data sources have not been adequately exploited.

The above roles of plant operators and supervisors imply that they are an integral part of the overall control system. The current approach to designing control systems has not adequately addressed this point. It is done mainly in terms of identifying the process dynamics, selecting measurements, defining control structures and selecting algorithms.

This book introduces development in automatic analysis and interpretation of process operational data both in real-time and over the operational history, and describes new concepts and methodologies for developing intelligent, state space based systems for process monitoring, control and diagnosis. It is known that processes can have multiple steady and also abnormal states. State space based monitoring and diagnosis can project multivariate real-time measurements onto a point in the operational state plane and monitor the trajectory of the point which can identify previously unknown states and the influence of individual variables. It is now possible to exploit data mining and knowledge discovery technologies to the analysis, representation, and feature extraction of real-time and historical operational data to give deeper insight into the systems behaviour. The emphasis is on addressing challenges facing interpretation of process plant operational data, including the multivariate dependencies which determine process dynamics, noise and uncertainty, diversity of data types, changing conditions, unknown but feasible conditions, undetected sensor failures and uncalibrated and misplaced sensors, without being overwhelmed by the volume of data.

To cover the above themes, it is necessary to cover the following topics,

- new ways of approaching process monitoring, control and diagnosis
- specification of a framework for developing intelligent, state space based monitoring systems
- introduction to data mining and knowledge discovery
- data pre-processing for feature extraction, dimension reduction, noise removal and concept formation
- multivariate statistical analysis for process monitoring and control
- supervised and unsupervised methods for operational state identification
- variable causal relationship discovery
- software sensor design

The methodologies and concepts are illustrated by considering illustrative examples and industrial case studies.

Xue Z. Wang

Leeds, England, 1999

ACKNOWLEDGEMENTS

The author wishes to acknowledge the invaluable contribution of the late Professor Colin McGreavy in the preparation and editing of this text.

The author would like to thank his present and former colleagues and students at the University of Leeds, Drs F.Z. Chen, B.H. Chen, S.H. Yang, Y.C. Huang, Mr. R.F. Li, Ms B. Yuan, Dr. S.A. Yang, Mr. R. Garcia-Flores, Mr. D. Sampson, and Mr. A. Watson who have provided much help with this book. He acknowledges gratefully Dr. M.L. Lu at Aigis Systems USA who has given useful input continuously. Thanks are also due to Dr. G.G. Goltz for his friendship and help in many ways.

Particular thanks go to Professors M.A. Johnson and M. J. Grimble at the Industrial Control Centre, University of Strathclyde for their encouragement. Professor M.A. Johnson has read the manuscript and given very useful input. The author would like to extend his thanks to Mr. O. Jackson, Ms E. Fuoco and Mr. N. Pinfield at Springer-Verlag London for their successful cooperation and help.

The author is grateful to the University of Leeds for providing the supporting facilities in the preparation of the book. The book has also benefited from an EPSRC grant (GR/L/61774).

CONTENTS

CHAPTER 1 INTRODUCTION

Over the last twenty years, it has become increasingly obvious that the performance of process control systems depends not only on the control algorithms but also on how these integrate into the operational policy in terms of safety, environmental protection, equipment protection, as well as general monitoring to identify poor performance and detect faults. This calls for creating a virtual environment which is able to provide a comprehensive assessment of performance and can identify the factors which determine it. The important agents which are responsible for bringing these together are the plant operators and supervisors. This implies that it is important to consider them as part of the overall solutions, and if they are part of the system they must be provided with the means of carrying out the role effectively. While attention has been given to improving the interface of control systems for operators and supervisors through the design of information display and alarm systems, most of this is concerned with awareness, with little concern as to processing functionality in assessing the large volume of multivariate data more effectively.

This book addresses these issues by seeking to make use of emerging data mining and knowledge discovery technology (KDD) to develop approaches for designing state space based process monitoring and control systems so as to integrate plant operators and supervisors into the operational strategy. Modern computer control and automatic data logging systems create large volumes of data, which contain valuable information about normal and abnormal operations, significant disturbances and changes in operational and control strategies. The data unquestionably provides a useful source for supervisors and engineers to monitor the performance of the plant and identify opportunities for improvement and causes of poor performance. The volume of data is generally so large and data structure too complex for it to be used for characterisation of behaviour by manual analysis. Data mining and KDD offers the capability of extracting knowledge from such data which can be used for developing state space based process monitoring systems.

The first chapter reviews the current approaches to process monitoring with particular reference to distributed control systems (DCS) displays, monitoring charts for statistical quality control, and the use of a concept of the operating window. This leads naturally to the concept of state space based process monitoring and control which is well suited for defining operators and supervisors requirements. Based on this, it is possible to develop a conceptual architecture for system design.

1.1 Current Approaches to Process Monitoring, Diagnosis and Control

To meet the goals of process control requires monitoring and control of process variables such as temperature, pressure, flow, composition and the levels in vessels. In modern distributed control systems, to assist in process monitoring and fault diagnosis, the measurements are displayed as groups of variables, dynamic trends and alarms. To do this effectively requires careful consideration as to the design of the displays.

A first requirement is to collect and display as much information as possible which is relevant. In fact, being able to collect and provide access to a large amount of information of measurement of variables is regarded as one of the most important advances of DCS when compared with earlier systems [1]. In the case of a typical olefin plant, for example, over 5000 measurements need to be monitored [2].

Secondly, the display should be arranged in a way that makes it easy for operators to assimilate the measured values, obtain associated information about key variables provided by independent sensors and diagnose incipient problems, preferably before causing major upsets. This requires that the interface display to operators should be properly designed. The display to operators in a modern DCS system has a hierarchical structure as shown in Figure 1.1 [1]. This arrangement allows engineers and operators having varied responsibilities to access information at different levels in a convenient way. For example, supervisors might be mainly interested in the general performance of the plant in terms of operating capacity, energy efficiency, overall mass balances and normal and abnormal status of plant areas or units. On the other hand, operators are more concerned with one section of a plant and monitoring the associated variables.

Other features of DCS displays make it possible for operators to quickly assimilate data using colours to indicate different operational states. For example, green is used for normal and red for abnormal. Care is also needed in the location of displays, grouping of variables and grouping and sequencing of alarms.

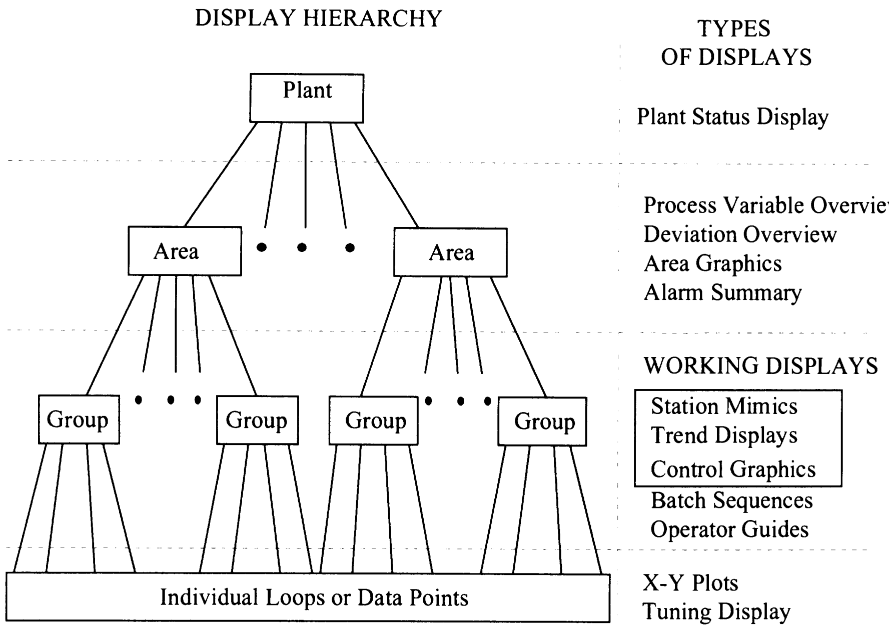


Figure 1.1 Typical DCS display hierarchy.

This emphasis again draws attention to the fact that supervisors and operators are part of the overall control system. In fact, not only are they responsible for many feedback control tasks which are not automated, such as switching feedbacks, but also they must undertake supervision of the general strategy and seek to develop an understanding of the process performance both in the short- and long- term. This understanding can be used to identify [3] :

- problems in the current operation.
- deteriorating performance of instruments, energy usage, equipment, or catalysts.
- better operating regions leading to improved product or operating efficiency.

Such tasks require operators to be able to not only access the data at the right time but more importantly to assimilate and assess the data quickly and correctly, especially when abnormal conditions arise. This is a very challenging task because the volume of data is generally very large: large-scale plants have as many as 20, 000 variables which are measured continuously [4, 5, 6]. Moreover, the data are multivariate and are interrelated so there is a need to make evaluations simultaneously. Humans are not able to simultaneously analyse problems involving more than three variables very effectively and this becomes more difficult when the data are corrupted by noise and uncertainty. The need to provide computer assistance in assimilating data has now become a major concern and it is important that automatic data analysis systems should be developed and integrated with DCS control.

1.2 Monitoring Charts for Statistical Quality Control

Automatic process control compensates for the effects of disturbances and maintains the controlled variable at the desired value. However, it does not eliminate the course of poor operation. Since the sources of disturbances have not been eliminated, leaving the process susceptible to future disturbances from the same source. Statistical process control (SPC) has the goal of detecting and eliminating disturbance. SPC works in conjunction with automatic control and monitors the performance of a process over time in order to verify that the process meets the quality requirements [22]. SPC charts such as Shewhart [7, 8, 9] , cumulative sum (CUSUM) [10] or exponentially weighted moving average (EWMA) charts [11, 12] are used to monitor the key product variables in order to detect the occurrence of any event having a special or assignable cause. In the case of assignable causes, long-term improvements can be made in process operations which enhance product quality.

Figure 1.2 is an example of a Shewhart chart. It shows the variation in a single variable against the statistical mean and the upper and lower limits defining an acceptable quality band. The mean and the upper and lower limits are obtained by statistical analysis of historical data. If the measured value falls outside either of the limits, a diagnostic analysis is carried out to determine the assignable cause so that appropriate corrective action can be taken.

Most SPC methods record only a small number of variables, usually the final product quality variables which are examined sequentially. Such an approach is now not acceptable for modern processes. Modern computer based monitoring collects massive amounts of data continuously. Variables such as temperatures, pressures, flowrates etc. are typically measured every seconds although only a few of the underlying events drive the process at any time: in fact, all measurements are simply aspects of the same underlying events. Examining one variable at a time, is of limited value. There is clearly the need to develop methods to examine a number of variables simultaneously.

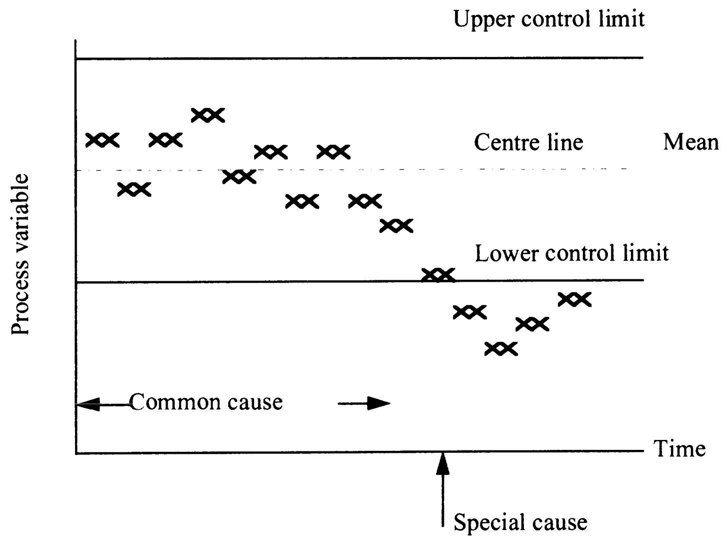


Figure 1.2 An example of Shewhart chart.

1.3 The Operating Window

The use of an operating window enables several variables to be considered together and is useful in monitoring an allowable region. It covers a region of possible steady-states for a process limited by constraints such as safety, product quality and equipment performance [8]. An example of an operating window is shown in Figure 1.3, where the region bounded by the solid curve represents the allowable steady-states. The dashed locus (or trajectory) shows that during a disturbance the operating point can move outside of the steady-state region. Operating windows have not been adopted in DCS display design as widely as they ought to have and there is still the problem of dimension limitation. Systems aiming to replace or help operators in process status assessment require capabilities for dealing with much higher dimensionality.

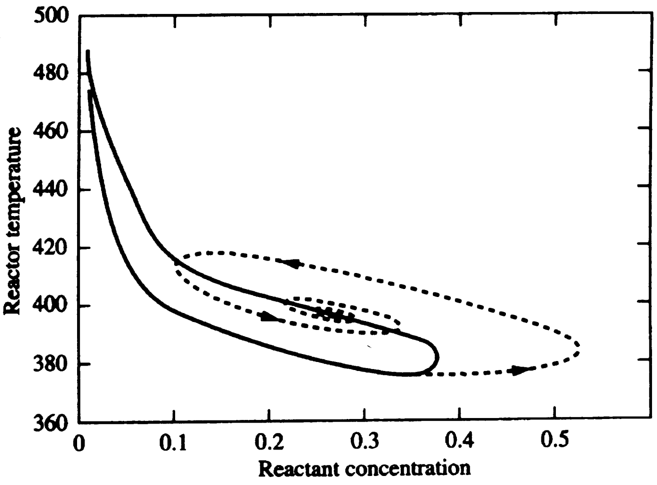


Figure 1.3 An example of operating window and trajectory.

1.4 State Space Based Process Monitoring and Control

In practice, operators are usually more concerned with the current operational status and evolving patterns of behaviour, rather than the instant values of specific variables. The judgement on the operational states requires the simultaneous assimilation of the multivariate process variables. In doing this, operators are actually projecting the process to a single point of a state space. It is clear that the above described monitoring charts and operating windows are still far less powerful than what is required. The concept of state space based monitoring is helpful in clarifying the ideas behind this book. Figure 1.4 illustrates how a process can be operated in various normal and abnormal operational states or modes that have different characteristics in terms of operability and controllability, safety and flow

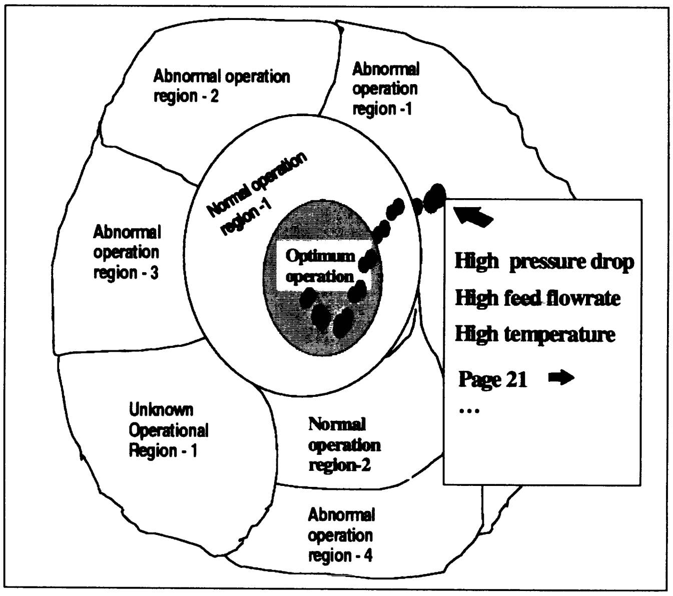

Abnormal operation region - 2
Abnormal operation region -1
Normal operation region -1
Optimum operation
Abnormal operation region - 3
High pressure drop
High feed flowrate
High temperature
Page 21
...
Unknown Operational Region - 1
Normal operation region-2
Abnormal operation region - 4

Figure 1.4 Illustration of state space based operational plane.

patterns among other things. The accumulation of know-how derived from previous experience and from computer simulation makes it possible to gain better insight into the operational behaviour of equipment. A state space based monitoring system can therefore be identified to have the following functions.

It should be able to automatically assimilate the real-time measurements, project the process to a specific state and indicate the path of the operating point in the state space in real-time. It is now known that a process or a unit operation can operate at abnormal and multiple steady states. The best known example is the exothermic continuous stirred tank reactor (CSTR) with a cooling water jacket [8, 13]. Multiple steady-state behaviour is also possible in distillation columns [14], reactive distillation processes [15], and refinery fluid catalytic cracking processes [16]. Changes of product specifications and feedstock properties, which are very common today [17], as well as large disturbances may also drive a process to a different operating state. These operational states are not obvious without careful analysis. Abnormal operations can take various forms and are more difficult to predict. Nevertheless, with the gradual accumulation of knowledge, more insight is being generated about operational behaviour patterns. It is also important to be able to identify various unfamiliar operational states, whether normal or abnormal.

Moreover it should be possible to identify the most important variables which are responsible for changes in operational states and so provide guidance for operators in adjusting the process. While the schematic in Figure 1.4 is yet another facet of system characteristics, it will not replace the traditional DCS display. It provides a different perspective of system behaviour.

1.5 Characteristics of Process Operational Data

The major challenge in developing the kind of state space based system described above arises from the characteristics of operational data, which are summarised as follows:

- Large volume. A DCS automatic data logging system continuously stores data. The large volume makes manual probing almost impossible. Large volumes of data also demand large computer memory and high speed.
- High dimensionality. The behaviour of a process is usually defined by a large number of correlated variables. As a result it is difficult to visualise the behaviour without dimension reduction.
- Process uncertainty and noise. Uncertainty and noise emphases the need for good data pre-processing techniques.
- Dynamics. In operational status identification, it is very important to take account of the dynamic trends. In other words, the values of variables are dynamic trends. Many data mining and knowledge discovery tools, such as the well-known inductive machine learning system C5.0 [17, 18], are mainly designed to handle categorical values such as a colour being red or green. They are not effective in dealing with continuous-valued variables. These tools are not able to handle variables that take values as dynamic trends.
- Difference in the sampling time of variables. On-line measurements and laboratory analyses have variable sampling periods.
- Incomplete data. Some important data may not be recorded.
- Small and stale data. Sometimes, data analysis is used to identify abnormal operations. The data corresponding to abnormal operations might be buried in a huge database. Some tools are not effective in identifying small patterns in a large database.
- Complex interactions between process variables. Many techniques require that attributes be independent. However, many process variables are interrelated.

- Redundant measurements. Sometimes several sensors are used to measure the same variable, which gives rise to redundant measurements.

Current methods only address some of these issues, certainly not all and the following observations can be made:

(1). Data pre-processing is critical for various reasons including noise removal, data reconciliation, dimension reduction and concept formation.

(2). Effective integration of the tools is needed. It means combining various tools for data preparation for other tools or for validation.

(3). Validation of discoveries from the data and presentation of the result is essential. Many times, because of lack of knowledge about the data, interpretation becomes a major issue.

(4). Windowing and sampling from a large database for analysis. This is necessary particularly for analysis of historical operational data.

1.6 System Requirement and Architecture

To develop the kind of state space based monitoring and control environment described above calls for an integrated data mining and KDD system. The system should have the following functions: (1) identification of operational states; (2) projection of the operation to a single point of the operational plane; and (3) giving explanations on the major variables that are responsible for the projection and providing guidance on adjustment. A conceptual system architecture and associated components are shown in Figure 1.5. It is important for a system to be able to provide some basic functions and be flexible enough to be tailored to meet special purposes [21]. The basic functions include:

- Pattern discovery. Grouping data records into clusters and then analysing the similarities and dissimilarities of data between clusters is an important starting point for analysis. An obvious example is identifying abnormal conditions.
- Trend and deviation analysis. Various technologies for trend and deviation analysis are available including statistics and calculation of mean and standard deviation.
- Link and dependency analysis. The link and dependency between performance metrics is important in understanding process behaviour and improving performance. Some existing data mining methods such as the inductive learning

approach C5.0 [18, 19, 20], as well as many graphical tools, are not able to be applied because of the real-valued dynamic trends and interactions between variables.

- Summarising. This provides a compact description of a subset of data, such as the mean and standard deviation of all fields. More sophisticated techniques use summary rules, multivariate visualisation techniques, and functional relationships between variables.
- Sequence analysis. Analysing models of sequential patterns (e.g., in data with time dependence, such as time series analysis) aims at generating the sequence or extracting report deviations and trends over time. A typical example is in batch process operations.
- Regression. This is required for predictive model development, as in the case of software sensor models.

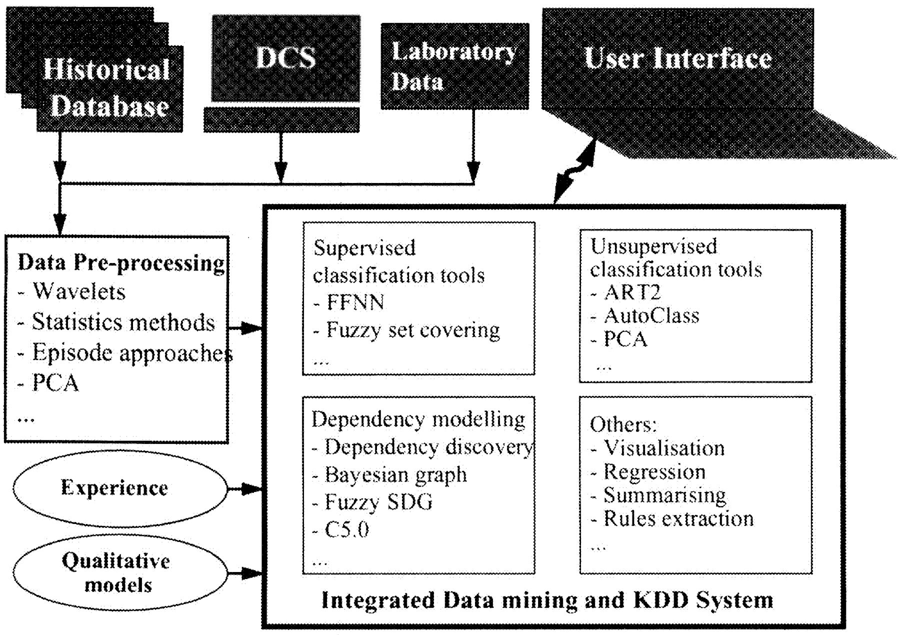


Figure 1.5 The conceptual system architecture.

1.7 Outline of the Book

The rest of the book is organised as follows. In Chapter 2 the methodologies and tools for data mining and KDD are briefly reviewed. Chapter 3 focuses on data pre-processing for the purpose of feature extraction, dimension reduction, noise removal and concept formation. The emphasis is on processing of dynamic trend signals which are considered as the most important information in operational state identification. Three approaches are introduced, namely principal component analysis, wavelet analysis, and episode representation.

Multivariate statistical analysis methods are introduced in Chapter 4 for analysis of process operational data and developing multivariate statistical control charts. These include linear and nonlinear principal component analysis (PCA), partial least squares (PLS), and multiblock PCA. Examples are used to illustrate the approaches and an industrial case study is described which uses PCA to discover knowledge from data for operational strategy development and product design.

Supervised machine learning approaches are discussed in Chapter 5 for identification of process operational states. While the focus is put on feedforward neural networks (FFNNs), other methods are also introduced and compared with FFNNs, including fuzzy FFNNs, single layer percetron, fuzzy set covering method and fuzzy signed digraphs.

Supervised machine learning requires data with known classification as training data and therefore learns from known to predict unknown, while unsupervised approaches do not require training data therefore are able to learn from unknown. Chapter 6 is devoted to unsupervised approaches for identification of operational states. An integrated framework (ARTnet) combining unsupervised neural network ART2 with wavelet feature extraction is developed, which uses wavelet as the substitute of the data pre-processing part of ART2. It is shown that ARTnet is superior over ART2 in avoiding the adverse effect of noise and is faster and more robust. A Bayesian automatic classification approach (AutoClass) is also described. The advantage of the approach is that it does not require the users to give any input: the system determines the classification scheme automatically. A refinery fluid catalytic cracking process is used to illustrate the approaches.

Most approaches for operational state identification, whether supervised or unsupervised, are based on calculating a *distance* or *similarity* measure. They give the classification but not causal explanations. *Conceptual clustering* which is introduced in Chapter 7, on the other hand is able to create a conceptual clustering language as well as giving a prediction of operational states. An inductive learning approach is introduced in this chapter for conceptually clustering operational states.

The methods introduced in Chapter 8 are able to identify cause-effect links between variables and between variables and operational states, as well as automatic discovery of operational rules from process operational data. Fuzzy sets, rough sets, neural networks and fuzzy neural networks are introduced for automatic generation of rules from data while the emphasis is on fuzzy neural networks.

Software sensor design for real-time monitoring of hard-to-measure variables and inferential control is examined in Chapter 9. The approach is based mainly on the use of neural networks. The main concerns are with selection of training and test data, selection of input variables as well as validation of data and models.

Chapter 10 closes the book with a brief summary of important issues and suggestions for future work.

CHAPTER 2

DATA MINING AND KNOWLEDGE DISCOVERY - AN OVERVIEW

2.1 Definition and Development

The emerging of data mining and knowledge discovery in databases (KDD) as a new technology is due to the fast development and wide application of information and database technologies. With the increasing use of databases the need to be able to digest large volumes of data being generated is now critical. It is estimated that only 5%-10% of commercial databases have ever been analysed [23]. As Massey and Newing [24] indicated that database technology has been successful in recording and managing data but failed in the sense of moving from data processing to making it a key strategic weapon for enhancing business competition. The large volume and high dimensionality of databases leads to the breakdown of traditional human analysis. Data mining and KDD is aimed at developing methodologies and tools to automate the data analysis process and create useful information and knowledge from data to help in decision making (Figure 2.1). A widely accepted definition is given by Fayyad et al. [25] in which KDD is defined as the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. The definition regards KDD as a complicated process comprising a number of steps and data mining is one step in the process.

The goal of data mining and KDD is very broad and can describe a multitude of fields of study related to data analysis. It is known that statistics has been preoccupied with this goal for over a century. Other fields related to data analysis

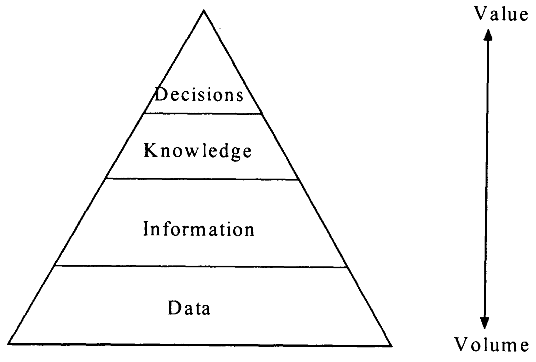


Figure 2.1 From volume to value.

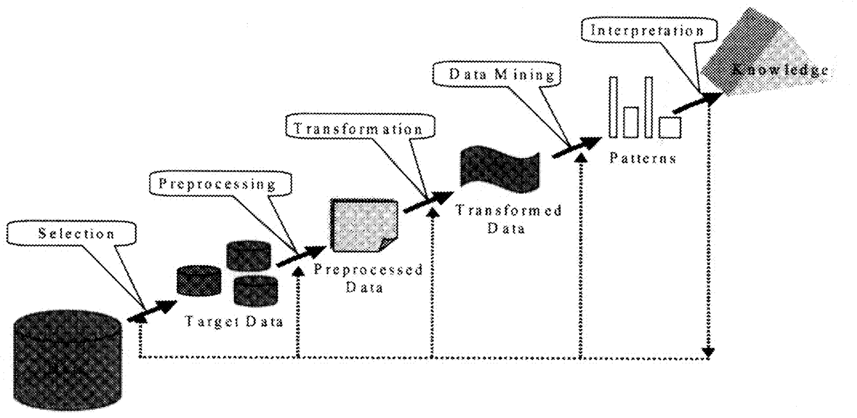


Figure 2.2 An overview of the steps comprising the KDD process.

include statistics [26, 27], data warehousing [28, 29, 30], pattern recognition, artificial intelligence and computer visualisation. Data mining and KDD draws upon methods, algorithms and technologies from these diverse fields, and the unifying goal is extracting knowledge from data .

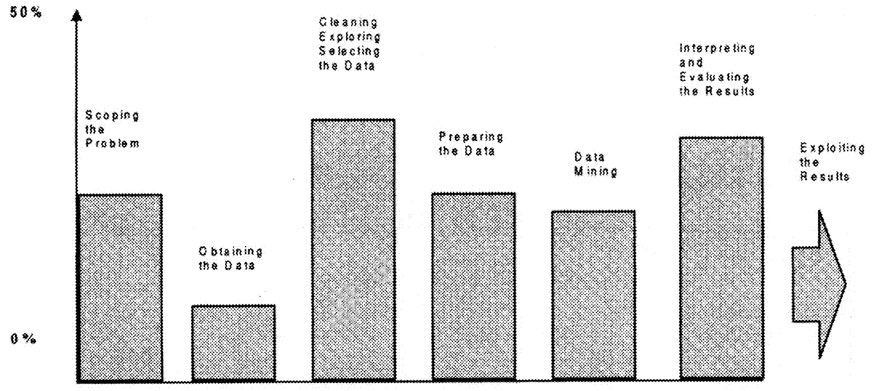


Figure 2.3 Relative effort of a data mining project.

Over the last ten years data mining and KDD has been developing at a dramatic speed. In Information Week's 1996 survey of the 500 leading information technology user organisations in the US, data mining came second only to the Internet and intranets as having greatest potential for innovation in information technology. The rapid progress is reflected not only by the establishment of research groups on data mining and KDD in many international companies, but also by the investment from banking, telecommunication and marketing sectors.

Figure 2.2 provides an overview of the activities involved in KDD and Figure 2.3 shows the typical distribution of effort.

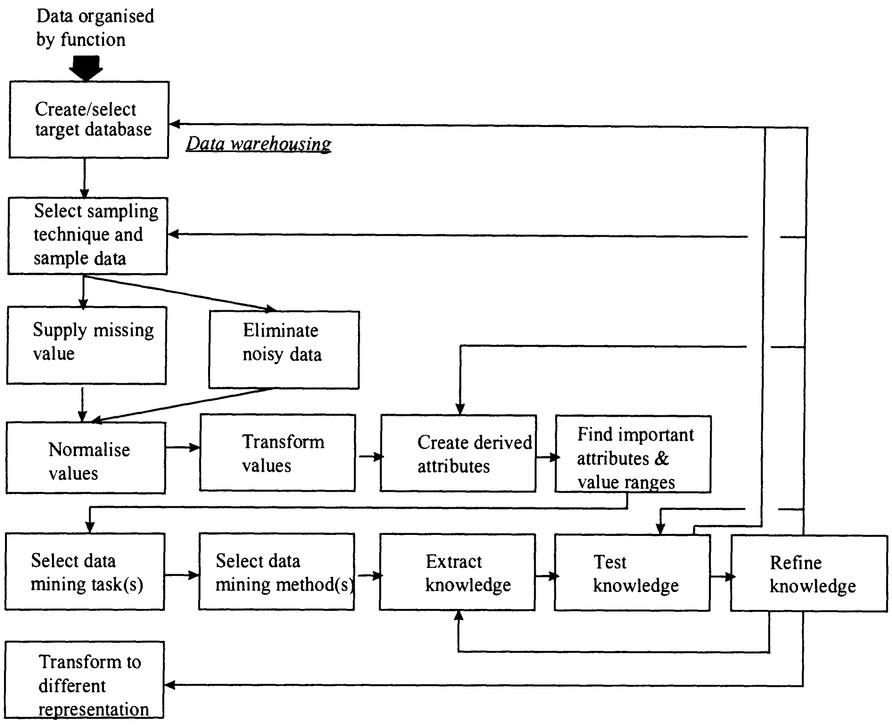
2.2 The KDD Process

Data mining and KDD is a very complex process, typically involving the following procedures (Figure 2.4) [23, 34]:

(1). Developing an understanding of the application domain, the relevant prior knowledge and the goals of the end-user.

(2). Creating a target data set: selecting a data set, or focusing on a subset of variables or data samples, on which discovery is to be performed.

(3). Data pre-processing and cleaning: basic operations such as the removal of noise or outliners if appropriate, collecting the necessary information to model and account for noise, deciding on strategies for handing missing data fields, and accounting for time sequence information and known changes.



Data organised by function
Create/select target database
Data warehousing
Select sampling technique and sample data
Supply missing value
Eliminate noisy data
Normalise values
Transform values
Create derived attributes
Find important attributes & value ranges
Select data mining task(s)
Select data mining method(s)
Extract knowledge
Test knowledge
Refine knowledge
Transform to different representation

Figure 2.4 The KDD process.

(4). Data reduction and projection: finding useful features to represent the data depending on the goal of the task. Using dimensionality reduction or transformation methods to reduce the effective number of variables under consideration or to find invariant representations for the data.

(5). Choosing the data-mining task: deciding whether the goal of the KDD process is logical, summarising, classification, regression, prediction, and clustering etc.

(6). Choosing the data analysis algorithm(s): selecting method(s) to be used for searching for patterns in the data. This includes deciding which models and parameters may be appropriate (e.g., models for categorical data are different from models on vectors over the real domain) and matching a particular data mining method with the overall criteria of KDD process.

(7). Data mining: searching for patterns of interest in a particular representational form or a set of such representations, including classification rules or trees, regression, clustering, sequence modelling, dependency, and so forth. The user can significantly aid the data mining method by correctly performing the preceeding steps.

(8). Interpreting mined patterns, and possible return to any of the previous steps.

(9). Consolidating the discovered knowledge: incorporating this discovery knowledge into the performance system, taking actions based on the knowledge, and reporting it to interested parties. It also includes checking or testing for potential conflicts with previously believed (or extracted) knowledge.

Data mining and KDD is potentially valuable in virtually any industrial and business sectors where database and information technology is used. The following are some reported applications:

- Fraud detection: identify fraudulent transactions
- Loan approval: establish the credit worthiness of a customer requesting a loan
- Investment analysis: predict a portfolio's return on investment [32]
- Portfolio trading: trade a portfolio of financial instruments by maximising returns and minimising risks.
- Marketing and sales data analysis: identify potential customers; establish the effectiveness of a sales campaign [32].
- Manufacturing process analysis: identify the causes of manufacturing problems.
- Experiment result analysis: summarise experiment results and predictive models.
- Scientific data analysis [35].
- Intelligent agents and WWW navigation.

2.3 Data Mining Techniques

Data mining methods and tools can be categorised in different ways [25, 33, 28, 37]. According to functions and application purposes, data mining methods can be classified as clustering, classification, summarisation, dependency modelling, link analysis and sequence analysis. Some methods are traditional and established and some are relatively new. In the following a very brief review of the techniques is given.

2.3.1 Clustering

Given a number of data patterns (sometimes called instances, cases, observations, samples, objects, or individuals) as shown in Table 2.1, each of which is described by a set of attributes, clustering (also called unsupervised machine learning) aims to devise a classification scheme for grouping the objects into a number of classes such that instances within a class are similar, in some respect, but distinct from those from other classes. This involves determining the number as well as the descriptions of classes. The grouping often depends on calculating a similarity or distance measure. Grouping multivariate data into clusters according to similarity or dissimilarity measures is the goal of some applications. It is also a useful step to look at the data before further analysis is carried out. The methods can be further categorised according to requirement on prior knowledge of the data. Some methods require the number of classes to be an input though the descriptions of the classes and assignments of individual data cases are allowed to be unknown. For example, the Kohonen neural network [36, 38] is designed for this purpose. In some other methods, neither the number nor descriptions of classes are required to be known. The task is to determine the number and descriptions of classes as well as the assignments of data patterns. For example, the Bayesian automatic classification system - AutoClass [39, 40, 41, 42] and the adaptive resonance theory (ART2) [43] are designed for this purpose.

Table 2.1 An example of data structure.

Instances	**Attributes**					
	1	2	……	j	……	*m*
x_1	x_{11}	x_{12}		x_{1j}		x_{1m}
x_2	x_{21}	x_{22}		x_{2j}		x_{2m}
.						
x_i	x_{i1}	x_{i2}		x_{ij}		x_{im}
.						
x_n	x_{n1}	x_{n2}		x_{nj}		x_{nm}

As a branch of statistics, clustering analysis has been studied extensively for many years, mainly focused on distance-based clustering analysis, such as using the Euclidean distance. There are many text books on this topic [44, 45, 46]. A notable progress in clustering has been in unsupervised neural networks, including the self-organising Kohonen neural network [36, 38] and the adaptive resonance theory (ART) [47, 48, 43, 49, 50, 51]. There have been many reports on the application in operational state identification and fault diagnosis in process industries.

2.3.2 Classification

For a given number of data patterns such as those shown in Table 2.1, if the number and descriptions of classes as well as the assignments of individual data patterns are known, the task is to assign unknown data patterns to the established classes, the task belongs to *classification.* The most widely currently used classification approach is based on feedforward neural networks (FFNNs). Classification is also called *supervised machine learning* because it always requires data patterns with known class assignments to train a model which is then used for predicting the class assignment of new data patterns.

2.3.3 Conceptual Clustering and Classification

Most clustering and classification approaches depend on numerically calculating a similarity or distance measure and because of this they are often called similarity based methods. The knowledge used for classification assignment is often an algorithm which is opaque and essentially a black box. *Conceptual clustering* and *classification* on the other hand develops a qualitative language for describing the knowledge used for clustering and is basically in the form of production rules or decision trees which are explicit and transparent. The inductive system C5.0 (previously C4.5) is a typical approach [18, 19, 20], which is able to automatically generate decision trees and production rules from databases. Decision trees and rules have a simple representation form, making the inferred model relatively easy to comprehend by the user. However, the restriction to a particular tree or rule representation can significantly restrict the representation power. In addition available approaches were developed mainly for problem domains that variables only take categorical values, such as colour being green and red. They are not effective in dealing with variables that take numerical values. Discritisation of numerical variables to categorical descriptions is a useful approach. However more powerful discretisition techniques are required.

2.3.4 Dependency Modelling

Dependency modelling describes dependencies among variables. Dependency models exist at two levels: structural and quantitative. The structural level of the model specifies (often in graphical form, [52, 53]) which variables are locally dependent; the quantitative level specifies the strengths of the dependencies using some numerical scale. Examples of tools for dependency modelling include probabilistic (or Bayesian) graphs [54, 55, 56, 57, 58] and fuzzy digraph graphs [59, 60].

Table 2.2 A database example.

Variable values			
case	*x*1	*x*2	*x*3
1	1	0	0
2	1	1	1
3	0	0	1
4	1	1	1
5	0	0	0
6	0	1	1
7	1	1	1
8	0	0	0
9	1	1	1
10	0	0	0

Table 2.3 The probabilistic table associated with the probabilistic network structure of Figure 2.5(a).

$P(x_1=1)=0.6$	$P(x_1=0)=0.4$
$P(x_2=1 \mid x_1=1)=0.8$	$P(x_2=0 \mid x_1=1)=0.2$
$P(x_2=1 \mid x_1=0)=0.3$	$P(x_2=0 \mid x_1=0)=0.7$
$P(x_3=1 \mid x_2=1)=0.9$	$P(x_3=0 \mid x_2=1)=0.1$
$P(x_3=1 \mid x_2=0)=0.15$	$P(x_3=0 \mid x_2=0)=0.85$

Take probabilistic networks as an example. Table 2.2 shows a collection of 10 data patterns, each is described by three attributes. The task of dependency modelling using probabilistic networks is two-fold - learning the network structure and compiling a conditional probabilistic table. For the data collection of Table 2.2

it is not possible to know the most probable dependencies directly. Figure 2.5 (a) and (b) illustrate two such possible dependencies. Theoretically, for a given database there is a unique structure which has the highest joint probability and can be found by some algorithm such as those developed by Cooper and Herskovits [54] and Bouckaert [55, 56]. When a structure is identified, the next step is to find such a probabilistic table as shown in Table 2.3.

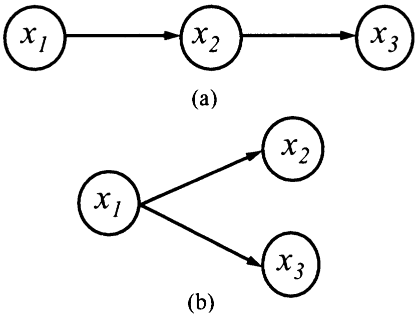


Figure 2.5 An example of two probabilistic networks.

Probabilistic graphical models are very powerful representation schemes which allow for fairly efficient inference and for probabilistic reasoning. However, few methods are available for infering the structure from data, and they are limited to very small databases. Therefore normally there is the need to find structure by interviewing domain experts. For a given data structure there are some successful reports on learning conditional probabilities from data.

Other dependency modelling approaches include statistical analysis (e.g., correlation coefficients, principal component and factor analysis) and sensitivity analysis using neural networks.

2.3.5 Summarisation

Summarisation provides a compact description for a subset of data. Simple examples would be the mean and standard deviations. More sophisticated functions involve summary rules, multivariate visualisation techniques, and functional relationships between variables.

Table 2.4 An example transaction database *D*.

Data cases	Items
100	A C D
200	B C E
300	A B C E
400	B E

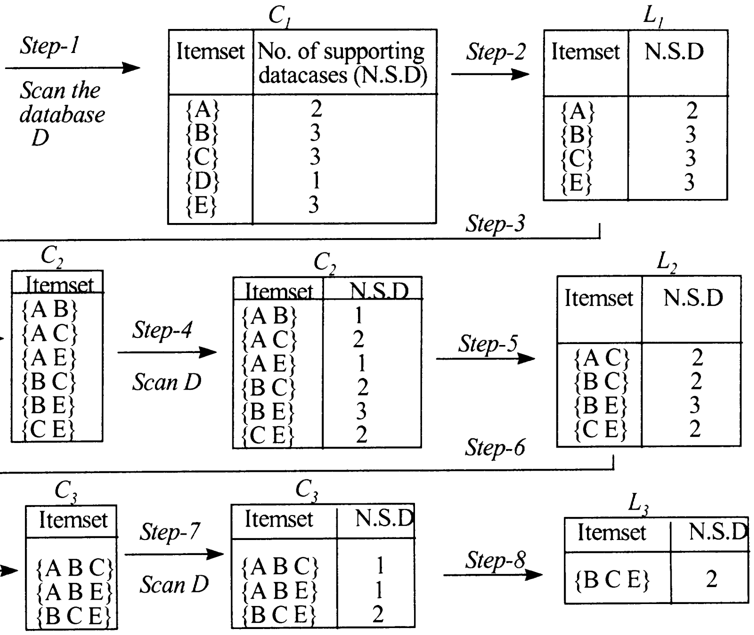


Figure 2.6 Generation of candidate itemsets and large itemsets.

A notable technique for summarisation is mining association rules [61, 62]. Given a relational database, mining association rules finds all associations of the form.

IF { set of values } THEN { set of values }

A rule is valid given two parameters T_c and T_s, such that, the rule holds with certainty > T_c and the rule is supported by at least T_s cases. Some commercial systems have been developed using this approach [61, 62].

A very simple example can be used to illustrate the approach. Table 2.4 shows an example of the transaction of part of a database. The purpose for mining the database is to summarise the number of occurrences of each candidate set of items (itemset), and then determine large sets of items based on a predetermined minimum support. Thus for a minimum support of 2 data cases, the mining process is shown in Figure 2.6.

Step-1 involves scanning the database *D* and summarising the supports for each item set. In *Step-2*, the itemset $\{C_1\}$ whose number of supporting cases is less than 2 is removed, and C_1 becomes L_1. In step-2, binary itemsets are formed and go through similar procedures. The above procedure repeats until L_3.

The approach of mining association tules has been quite successful [23]. However it also has some limitations. For example, it assumes all data is categorical and there are no good algorithms available for the numeric fields. It also assumes that the set of associations satisfying thresholds is sparse.

2.3.6 Regression

Linear and non-linear regression is one of the commonest approaches for correlating data. Statistical regression methods often require the user to specify the function over which the data is to be fitted. In order to specify the function, it is necessary to know the forms of the equations governing the correlation for the data. The advantage of such methods is that from the equation it is possible to gain some qualitative knowledge about the input - output relationships. However, if prior knowledge is not available, it is necessary to find out the most probable function by trial-and-error which may require very time consuming effort. Feedforward neural networks (FFNNs) do not need functions to be fixed in order to learn and have shown very remarkable results in representing non-linear functions. However the resulting function using a FFNN is not easy to understand and is virtually a black box without any explanations.

2.3.7 Case-based Learning

Case-based learning is based on acquiring knowledge represented by cases and it employs reasoning by analogy [63, 64]. Case-based learning focuses on the indexing and retrieval of relevant precedents. Typically the solution sequence is a parameterised frame or schema where the structure is more or less fixed, rather than expressed in terms of an arbitrary sequence of problem solving operators. Case-based reasoning is particularly useful for making use of data which has complex internal structures. Different from other data mining techniques, it does not require to have a large number of historical data patterns. There are only a few reports of the application of case-based reasoning in process industries such as case-based learning for historical equipment failure databases [65, 66] and equipment design [67].

2.3.8 Mining Time-series Data

Many industrial and business areas deal with time-series or dynamic data. It is apparent that all statistical and real-time control data in process monitoring and control are essentially time-series. Figure 2.7 shows the dynamic trend signals of a variable under two different operational conditions. It is very easy for humans to visually capture features of each trend and identify their difference. However for computers to perform the same task is difficult. Most KDD techniques cannot account for the time series of data. The techniques to deal with time series data are to carry out pre-processing of the data to use minimum data points to capture the features and remove noise. These techniques include filters, e.g., Kalman filters, Fourier and wavelet transforms, statistical approaches, neural networks as well as various qualitative signal interpretation methods. Chapter 3 will introduce some of these techniques for pre-processing dynamic data.

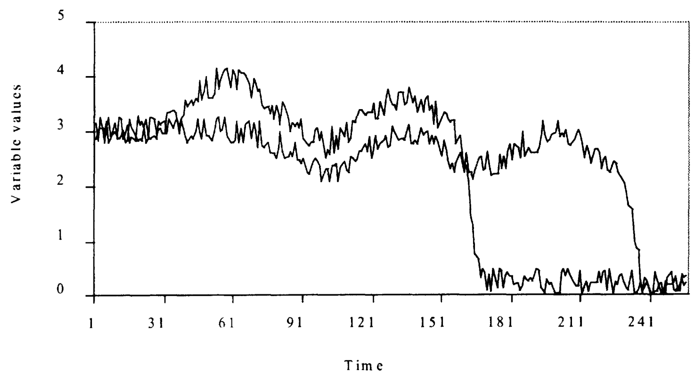


Figure 2.7 The dynamic trend signals of a variable under two different operational conditions.

2.3.9 Method Selection

There is no well-developed techniques for selecting data mining and KDD methods. It is still an art [23]. Apart from the general considerations such as cost, and support, there are some technical dimensions to the method selection. These include [23]:

(1) univariate vs. multi-variate data. Most approaches assume independence of variables or simply consider a single variable at a time.

(2) numerical vs. categorical or mixed data. Some methods are only suitable for numerical data, others only for categorical data. There are only a few cases which allow mixed data.

(3) explanation requirements or comprehensibility. Some tools give results which are implicit to users (black box), while others can give causal and explicit representations.

(4) fuzzy or precise patterns. There are methods such as decision trees which only work with clear cut definitions.

(5) sample independence assumptions. Most methods assume independence of data patterns. If there are dependency on the data patterns, it is necessary to remove or explore.

(6) availability of prior knowledge. Some tools require prior knowledge which might be not available. On the other hand, some others do not allow input of prior knowledge, causing a waste of prior knowledge.

It is important to be aware of the complexity of data which tends to contain noise and erroneous components and has missing values. Other challanges come from lack of understanding of the domain problem and assumptions associated with individual techniques. Therfore, data mining is rarely done in one step. It often requires using a number of approaches to use some tools to prepare data for other methods, or for validating purposes. As a result, multifunctional and integrated systems are required.

2.4 Feature Selection with Data Mining

Data pre-processing may be more time consuming and presents more challenges than data mining. Process data often contains noise and erroneous components and has missing values. There is also the possibility that redundant or irrelevant variables are recorded, while important features are missing. Data pre-processing includes provision for correcting inaccuracies, removing anomalies and eliminating duplicate records, and filling holes in the data and checking entries for consistency. It also requires making the necessary transformation of the original to put it in the format suitable for data mining tools.

The other important requirement with KDD process is feature selection. KDD is a complicated task and often depends on the proper selection of features. Feature selection is the process of choosing features which are necessary and sufficient to represent the data. There are several issues influencing feature selection, such as . masking variables, the number of variables employed in the analysis and relevancy of the variables [68].

Masking variables hide or disguise patterns in data. Numerous studies have shown that inclusion of irrelevant variables can hide real clustering of the data so only those variables which help discriminate the clustering should be included in the analysis [68, 69, 70].

The number of variables used in data mining is also an important consideration. There is generally a tendency to use more variables. However, increased dimensionality has an adverse effect because, for a fixed number of data patterns, increased dimensionality makes the multidimensional data space sparse.

However failing to include relevant variables also causes failure in identifying the clusters. A practical difficulty in mining some industrial data is to know if all important variables have been included in the data records.

Prior knowledge should be used if it is available. Otherwise, mathematical approaches need to be employed. Feature extraction shares many approaches with data mining. For example, principal component analysis (PCA), which is a useful tool in data mining, is also very useful for reducing the dimension (PCA and its applications are introduced in Chapters 3 and 4). However, PCA is only suitable for dealing with real-valued attributes. Mining of association rules mining is also an effective approach in identifying the links between variables which take only categoric values [68]. Sensitivity studies using feedforward neural networks (FFNNs) are also an effective way of identifying important and less important variables (sensitivity studies using FFNNs are introduced in Chapter 9). Gnanadeskikan et al. [69] reviewed a number of clustering techniques which identify discriminating variables in data.

2.5 Final Remarks and Additional Resources

This Chapter has provided an overview of data mining and KDD. Data mining and KDD makes use of various technologies including statistics, neural networks, machine learning, artificial intelligence, pattern recognition and databases. The unifying goal is to extract useful information and knowledge from massive data. It is a complex and interative process, starting with data access, continuing with data cleaning and pre-processing as well as data mining and knowledge discovery, finally culminating with intepretation and validation of results. It is important to be aware of the complexity of industrial data and that there are always some assumptions related to specific KDD techniques. Integration of various methods is necessary, so that some tools can be used in preparing data for other methods and results obtained using different methods can be compared.

Table 2.5 Useful data mining and KDD sites.

- The KDD Foundation - a starting point for exploring Internet resources in knowledge discovery and data mining.
  http://www.kdd.org/
- Decision Theory & Adaptive Systems Group, Microsoft Research
  http://www.research.microsoft.com/~fayyad
- BT's Data Mining Group homepage
  http://www.labs.bt.com/projects/mining/index.htm
- Birmingham University Data Mine
  http://www.cs.bham.ac.uk/~anp/TheDataMine.html
- The University of Ulster Data Mine
  http://iserve1.infj.ulst.ac.uk:8080/
- The Corporate KDD Bookmark
  http://www.cs.su.oz.au/~thierry/ckdd.html
- The Machine Learning Database Repository at the University of California Irvine
  http://www.ics.uci.edu/AI/ML/Machine-Learning.html
- ML Net Site - Machine learning publications, data and software
  http://www.gmd.de/

Despite the rapid growth, the success achieved and a huge predicted market, KDD is still considered to be in its infancy. There are still many challenges to overcome. An obvious issue in process monitoring and control is how to deal with dynamics associated with the data. Another issue is that most data mining tools assume that the variables are independent of each other, but process variables are often connected. The challenges posed by operational data have already been summarised in Section 1.5.

Apart from the need to develop more reliable data mining and KDD tools, there is also the need to gain more experience in applying them to industrial and business problems.

For introduction and review of data mining and KDD, readers are referred to Fayyad et al. [71], Wu [72], Chen et al. [28], Simoudis et al. [73], Wu et al. [74], and Pyle [75]. There are also some useful web sites which are summarised in Table 2.5 and provide gateways to other resources.

CHAPTER 3
DATA PRE-PROCESSING FOR FEATURE EXTRACTION, DIMENSION REDUCTION AND CONCEPT FORMATION

This chapter describes data pre-processing for feature extraction, dimension reduction, noise removal and concept formation from monitored process measurements. The discussion is concerned with capturing the features in dynamic trend signals. A dynamic trend representation is the visualisation of the changing trajectory of a variable over time and consists of many sample values. However, in order to make effective use of trends in a computer based system, it is necessary to compress the data to fewer values. One of the earliest examples of dealing with such trends is based on a real time expert system G2 [76] which uses qualitative expressions such as temperature increase or decrease as descriptors. Later, various other approaches were developed including episodes [77, 78, 79], neural networks [80] and more recently wavelets [81, 82] and principal component analysis [83]. This chapter introduces principal component analysis, wavelet analysis as well as episode representations.

3.1 Data Pre-processing

Data pre-processing is used to

(1) filter out the noise components otherwise this may result in wrong conclusions being reached from the data.

(2) extract features, reduce the dimensionality of the original signal and retain as much relevant information as possible. The main reasons for feature extraction are,

first of all to minimise the dependencies between attributes and secondly to reduce dimensionality.

(3) Deal with the problem of variable sampling periods for data, such as on-line real time signals and laboratory analytical data.

(4) Develop concept formation because some data mining and KDD tools have been developed only for dealing with discrete-valued attributes and are not effective in dealing with continuous-valued variables. It is not possible to use variables represented by a trend without preprocessing the data.

It is worth noting that data pre-processing has many features in common with data mining, such as principal component analysis, supervised and unsupervised classification using statistical and neural network algorithms.

3.2 Use of Principal Component Analysis

The method of principal component analysis (PCA) was originally developed in the 1900's [84, 85], and has now re-emerged as an important technique in data analysis. The central idea is to reduce the dimensionality of a data set consisting of a large number of interrelated variables, while retaining as much as possible of the variation present in the data set. Multiple regression and discrimination analysis use variable selection procedures to reduce the dimension but result in the loss of one or more important dimensions. The PCA approach uses all of the original variables to obtain a smaller set of new variables (principal components - PCs) that they can be used to approximate the original variables. The greater the degree of correlation between the original variables, the fewer the number of new variables required. PCs are uncorrelated and are ordered so that the first few retain most of the variation present in the original set.

3.2.1 Basic Concepts: Mean, Variance, Covariance

It is convenient at this point to gave a brief summary of the basic points. In the univariate case the mean and variance are used to summarise a data set. The mean or mean value of a discrete distribution is denoted by μ and is defined by

$$\mu = \sum_i x_i f(x_i) \tag{3.1}$$

where $f(x_i)$ is the probability function of the random variable X considered. The mean is also known as the mathematical expectation of X and is sometimes denoted by $E(X)$.

The variance of a distribution is denoted by σ^2 and is defined by

$$\sigma^2 = \sum_i (x_i - \mu)^2 f(x_i) = E(x_i - \mu)^2 \tag{3.2}$$

In fact it is an index reflecting the deviation of x_i from the mean μ. In other words, the variance σ^2 describes the linear dependency of all x_i . The bigger the variance, the less dependent; the smaller the variance and hence the greater the linear dependency between x_i .

To summarise multivariate data sets, it is necessary to find the mean and variance of each of the p variables, together with a measure of the way each pair of variables is related. For the latter, the covariance or correlation of each pair of variables is used.

The population mean vector is given by $\mu' = [\mu_1, \mu_2, \ldots, \mu_p]$, where

$$\mu_i = E(x_i) \tag{3.3}$$

An estimate of μ based on n, p-dimensional observations is $\bar{\mathbf{x}}' = [\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_p]$, where $\bar{x}_i$ is the sample mean of variable x_i.

The vector representing the population variance is $\sigma' = E(x_i^2) - \mu_i^2$

An estimate of σ based on n, p-dimensional observations is $s' = [s_1^2, s_2^2, \ldots s_p^2]$, where s_i^2 is the sample variance of variable x_i.

The covariance of two variables x_i and x_j is defined by

$$\text{Cov}(x_i, x_j) = E(x_i, x_j) - \mu_i \mu_j \tag{3.4}$$

With p variables, $x_1, x_2, \ldots x_p$, there are p variances and $\frac{1}{2}p(p-1)$ covariances. In general these quantities are arranged in the $p \times p$ symmetric matrix, called covariance matrix, Σ,

$$\Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \cdot & \cdot & \cdot & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & \cdot & \cdot & \cdot & \sigma_{2p} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \sigma_{p1} & \sigma_{p2} & \cdot & \cdot & \cdot & \sigma_{pp} \end{pmatrix} \tag{3.5}$$

where $\sigma_{ij} = \sigma_{ji}$. The simple version of Σ , usually denoted, **S**, is generally estimated as

$$\mathbf{S} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})' \tag{3.6}$$

The covariance is often difficult to interpret because it depends on the units in which the two variables are measured; consequently it is conveniently standardised by dividing by the product of the standard deviations of the two variables to give a quantity called the correlation coefficient, ρ_{ij}, defined by

$$\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}} \tag{3.7}$$

The correlation coefficient lies between -1 and 1 and gives a measure of the linear relationship between variables x_i and x_j

3.2.2 Principal Component Analysis

Given a data matrix **X** representing n observations of each of p variables, $x_1, x_2, \ldots x_p$, the purpose of principal component analysis is to determine a new variable y_1, that can be used to account for the variation in the p variables, $x_1, x_2, \ldots x_p$. The first principal component is given by a linear combination of the p variables as

$$y_1 = w_{11}x_1 + w_{12}x_2 + \ldots + w_{1p}x_p \tag{3.8}$$

where the sample variance is greatest for all of the coefficients (also called weights), $w_{11}, w_{12}, \ldots w_{1p}$, conveniently written as a vector $\boldsymbol{w}_1$. The $w_{11}, w_{12}, \ldots w_{1p}$ have to satisfy the constraint that the sum-of-squares of the coefficients, i.e., $\boldsymbol{w}'_1\boldsymbol{w}_1$, should be unity.

The second principal component, y_2, is given by the linear combination of the p variables in the form:

$$y_2 = w_{21}x_1 + w_{22}x_2 + \ldots + w_{2p}x_p \tag{3.9}$$

or $\quad \boldsymbol{y}_2 = \boldsymbol{w}'_2\mathbf{x}$

which has the greatest variance subject to the two conditions,

$$\boldsymbol{w}'_2\boldsymbol{w}_2 = 1 \tag{3.10}$$

and

$$\boldsymbol{w}'_2\boldsymbol{w}_1 = 0 \text{ (so that } y_1 \text{ and } y_2 \text{ are uncorrelated)}$$

Similarly the jth principal component is a linear combination

$$y_j = \boldsymbol{w}'_j\mathbf{x} \tag{3.11}$$

which has greatest variance subject to

$$\boldsymbol{w}'_j\boldsymbol{w}_j = 1$$

$$\boldsymbol{w}'_j\boldsymbol{w}_i = 0 \qquad (i < j)$$

To find the coefficients defining the first principal component, the elements of $\boldsymbol{w}_1$ should be chosen so as to maximise the variance of y_1 subject to the constraint, $\boldsymbol{w}'_1 \boldsymbol{w}_1 = 1$. The variance of y_1 is then given by

$$\mathrm{Var}(y_1) = \mathrm{Var}(\boldsymbol{w}_1{}'\mathbf{x}) = \boldsymbol{w}'_1 \mathbf{S}\, \boldsymbol{w}_1 \tag{3.12}$$

where **S** is the variance-covariance matrix of the original variables (See Section 3.2.1). The solution of $\boldsymbol{w}_1 = (w_{11}, w_{12}, \ldots w_{1p})$ to maximise the variance y_1 is the eigenvector of **S** corresponding to the largest eigenvalue. The eigenvalues of **S** are roots of the equation,

$$||\mathrm{S} - \lambda \mathrm{I}| = 0 \tag{3.13}$$

If the eigenvalues are $\lambda_1, \lambda_2, \ldots \lambda_p$, then they can be arranged from the largest to the smallest. The first few eigenvectors are the principal components that can capture most of the variance of the original data while the remaining PCs mainly represent noise in the data.

PCA is scale dependent, and so the data must be scaled in some meaningful way before PCA analysis. The most usual way of scaling is to scale each variable to unit variance.

3.2.3 Data Pre-processing Using PCA

3.2.3.1 Pre-processing Dynamic Transients for Compression and Noise Removal

In computer control systems such as DCS, nearly all important process variables are recorded as dynamic trends. Dynamic trends can be more important than the actual real time values in evaluating the current operational status of the process and in anticipating possible future developments. Appendix C describes a data set of one hundred cases corresponding to various operational modes such as faults, disturbances and normal operation of a refinery reactive distillation process for manufacture of methyl tertiary butyl ether (MTBE), a lead-free gasoline additive. This can be used to illustrate the dimension compression capability of PCA. For each data case, twenty one variables are recorded as dynamic responses after a disturbance or fault occurs. Each trend consists of 256 sample points. Figure 3.1 shows the trends of a variable for two different cases. The eigenvalues of the first 20 principal components are summarised in Figure 3.2. It is apparent that the eigenvalues of the first few principal components can be used as a concise representation of the original dynamic trend, and so are used to replace the original responses for use in pattern recognition.

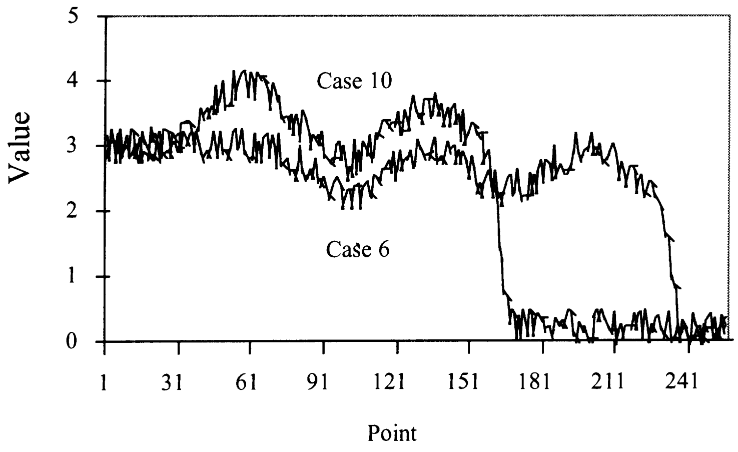


Figure 3.1 The dynamic trends of a variable for two data cases.

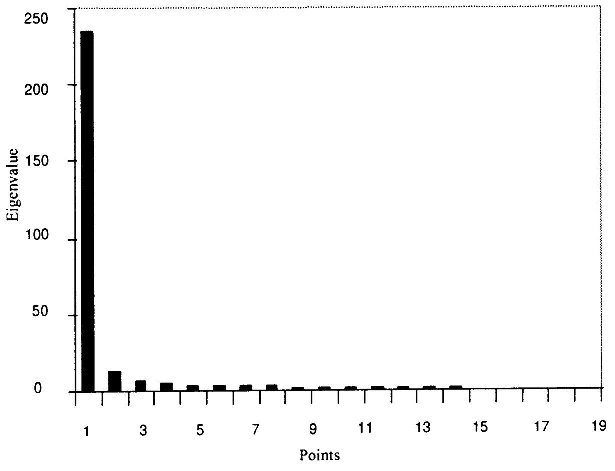


Figure 3.2 The first 20 eigenvalues of a variable.

3.2.3.2 Pre-processing of Dynamic Transient Signals for Concept Formation

Since the first two principal components can capture the main feature of a dynamic trend, this can be displayed graphically by plotting the eigenvalues on a two-dimensional plane. Figure 3.3 shows such a plot of the eigenvalues of the first two principal components of a variable F_0. A point in the two dimensional plane represents the feature of the variable response trend for one data case. Data points in region B have response trends which are similar and unlike those in region D.

The fact that a two-dimensional plot is able to capture the features can be seen from Figures 3.4 and 3.5. Figure 3.4 shows the dynamic responses of the variable T_MTBE for seven data cases. After being processed using PCA (actually the seven data cases are processed using PCA together with another 93 data cases, but here only the seven are shown for illustrative purpose), the results are shown on the two-dimensional PCA plane in Figure 3.5. It is clear that the dynamic trends of data cases 1 and 2 are more alike than with the others in Figure 3.4 and they are grouped closer in Figure 3.5. Similar observations can be made for data cases 40 and 80, as well as 14 and 15.

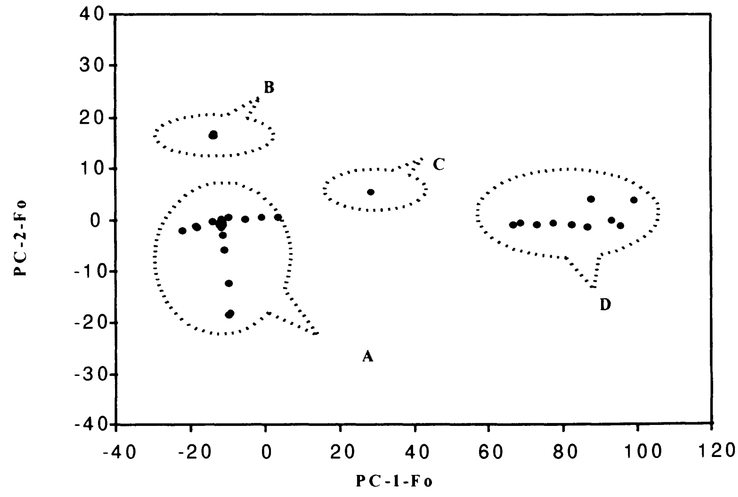


Figure 3.3 The PCA two dimensional plane of the variable Fo.

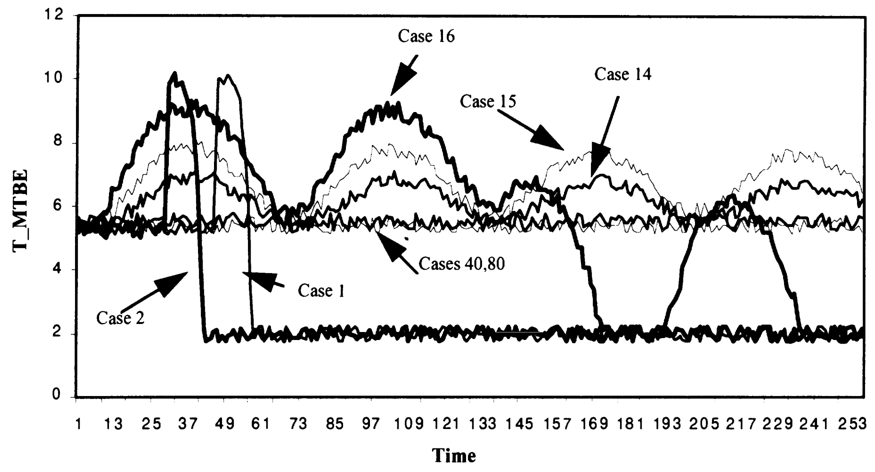


Figure 3.4 The dynamic trends of the temperature T_MTBE for the case study described in Appendix C.

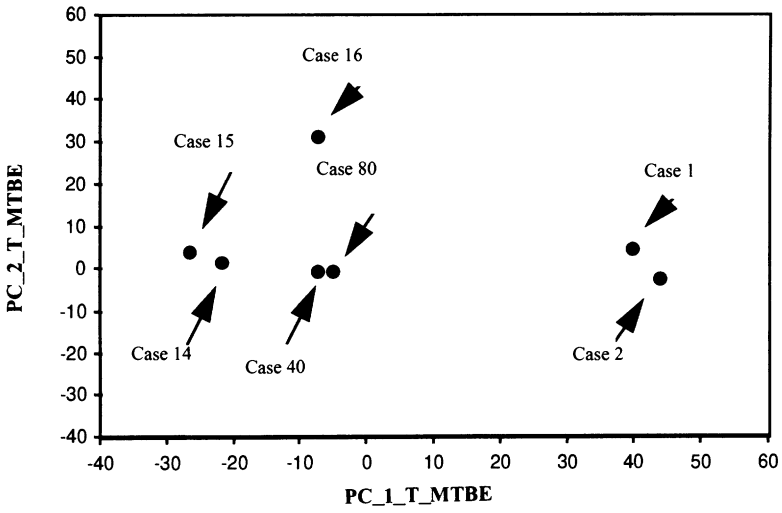


Figure 3.5 The projection of the dynamic trends of Figure 3.4 on the two-dimensional PCA plane.

The plot of the dynamic trends of a variable on a two dimensional plane, as depicted in Figure 3.3 is referred to as concept formation. Concept formation transforms a complicated trend to a concept, e.g., "the variable F_0 is in region D". The transformed concept of the trend of a variable can be used to develop knowledge based systems. A simple example is the following production rule for a case of a continuous stirred tank reactor where the data refer to historical operating data:

```
IF     Fo is in region D of Figure 3.3
AND    TR is in region C of Figure 3.6
THEN   The operation will be in region
       ABN-1 of Figure 3.7
```

A detailed discussion on using the concept formation method to develop conceptual clustering systems will be discussed later in Chapter 7.

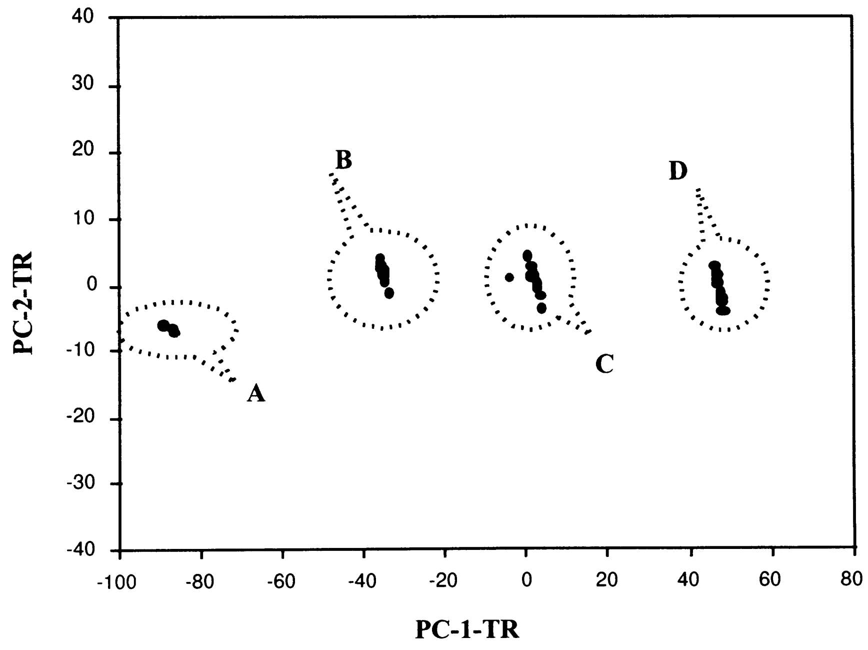


Figure 3.6 The PCA two dimensional plane of the variable TR.

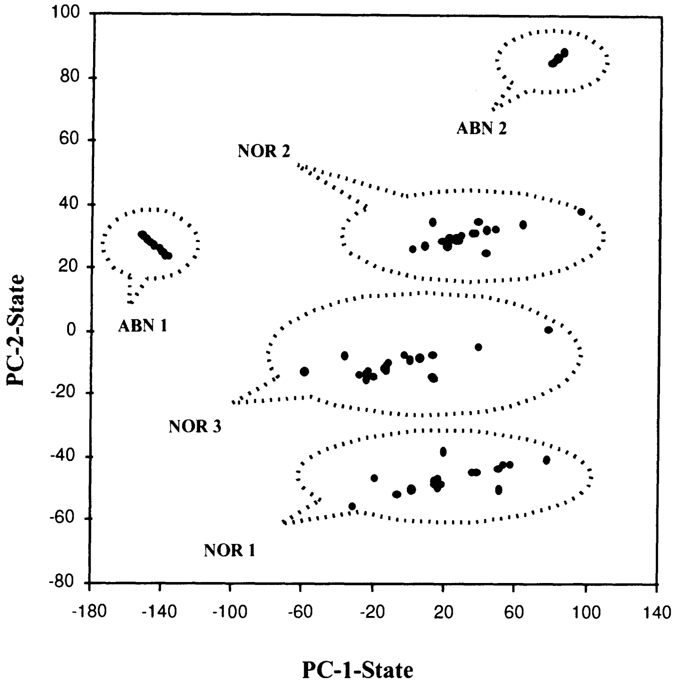


Figure 3.7 The PCA plane of operational states of a CSTR reactor.

3.2.3.3 Dependency Removal and Clearance of Redundancy Variables

Studies have found that presence of redundancy and irrelevant variables may deteriorate pattern recognition or hide the real patterns in the data [68] and so some data mining and KDD tools require the inputs to be independent. Sometimes it is not possible to directly identify the dependencies between variables. PCA can be used to pre-process the data and the first few principal components are then be used by other data mining and KDD tools.

3.3 Wavelet Analysis

Recently, wavelet analysis has emerged as a promising new approach for signal and image analysis and has been extended to process monitoring and control. In this section, it is convenient to start with the well-established approach based on Fourier transform for signal processing and how it relates to wavelet transforms. Wavelet transforms are then introduced, followed by its application for feature extraction.

3.3.1 Signal Processing Using Fourier Transform

Fourier transforms are well known as a useful technique for frequency analysis of a signal which breaks down a signal into constituent sinusoids of different frequencies. The transform is defined as

$$F(\omega) = \int_{-\infty}^{+\infty} f(t)\, e^{-i\omega t} dt \tag{3.14}$$

and the inverse is

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(\omega)\, e^{i\omega t} d\omega \tag{3.15}$$

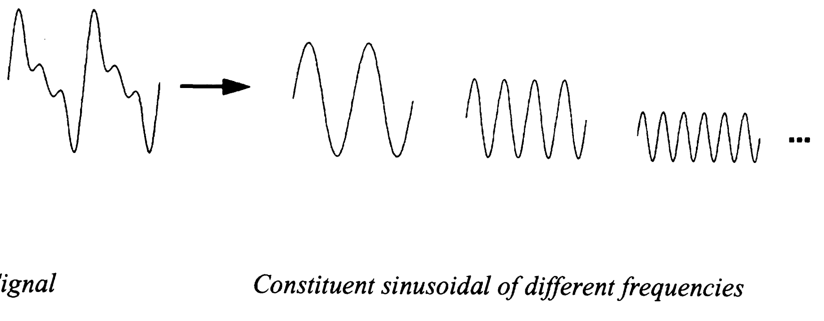
Here the Fourier transform can be viewed as the decomposition of a function f(t) into a sum of frequency components. This transform uses sine and cosine as its building blocks or basis functions to map a time function into the frequency domain. Figure 3.8(a) illustrates the Fourier transform process. However, the Fourier transform does not show how the frequencies vary with time in $f(t)$ so that looking at the transform of a signal, it is impossible to tell when a particular event took place. If a signal does not change with time, i.e., it is stationary, this is not important. However, most signals of interest contain numerous non-stationary or transitory characteristics: drift trends, abrupt changes related to beginning and ending of events. These characteristics are often the most important part of the signal, and Fourier analysis is not suited in detecting them.

The short-time Fourier transform (STFT), or windowed Fourier transform is able to overcome the limitation, and is now considered as the standard method for studying time-varying signals. The idea of STFT is to use a window which slides over the signal in time, and then to compute the Fourier transform within each window. The general definition of STFT is as follows

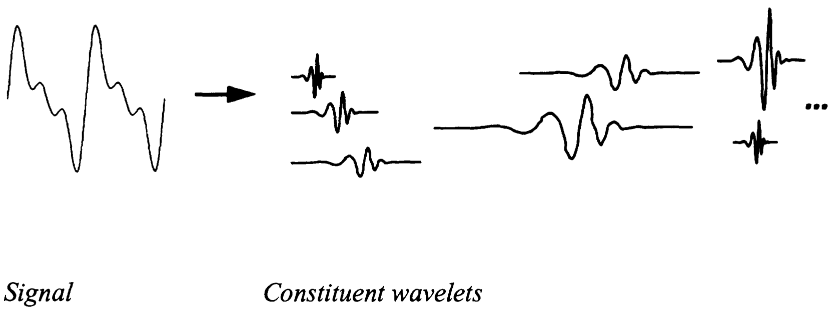
$$F(t, \omega) = \int_{-\infty}^{+\infty} f(\tau) g(\tau - t)\, e^{-j\omega\tau} d\tau \tag{3.16}$$

where $f(t)$ is the original function in time domain and $g(t)$ is the window function. The simplest window function is

$$g(t) = \begin{cases} 1 & t' \leq t \leq t \\ 0 & otherwise \end{cases} \tag{3.17}$$



(a) Fourier transform



(b) Wavelet transform

Figure 3.8 Comparison of Fourier and wavelet transforms.

In time-frequency analysis of a non-stationary signal, there are two conflicting requirements. The window width must be long enough to give the desired frequency resolution but must also be short enough not to lose track of time dependent events. While it is possible to design window shapes to optimise, or trade-off time and frequency resolution, there is a fundamental limitation on what can be done, for a given fixed window width. Figure 3.9 depicts a windowed SFTF [87].

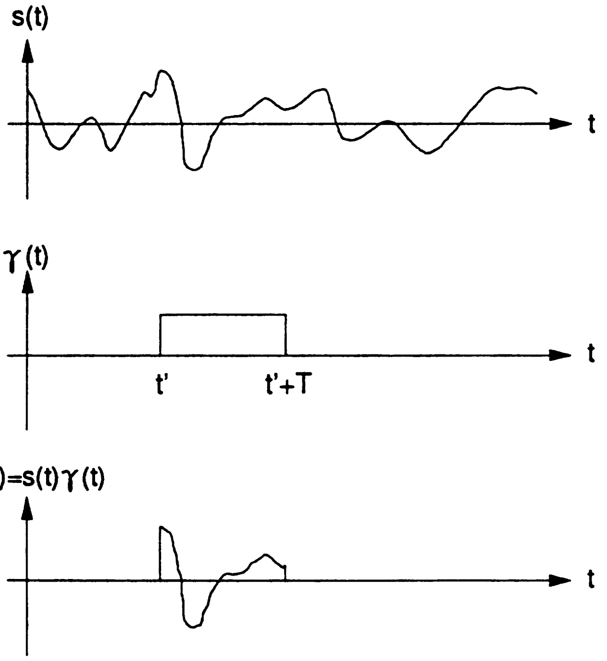


Figure 3.9 The windowing operation.

3.3.2 Signal Transformation Using Wavelets

Wavelet transformation is designed to address the problem of non-stationary signals [99, 100, 101]. It involves representing a time function in terms of simple, fixed building blocks, termed wavelets. These building blocks are actually a family of functions which are derived from a single generating function called the mother wavelet by translation and dilation operations (Figure 3.8(b)). Dilation, also known as scaling, compresses or stretches the mother wavelet, and translation shifts it along the time axis.

The mother wavelet satisfies

$$\int_{-\infty}^{+\infty} \psi(t)dt = 0 \tag{3.18}$$

and the translation and scaling operations on $\psi(t)$ creates a family of functions,

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \tag{3.19}$$

The parameter a is a scaling factor and stretches (or compresses) the mother wavelet and is easily understood by considering the sinusoid function in Figure

3.10. The scaling factor works exactly the same way with wavelets, can be seen in Figure 3.11: the smaller the scale factor, the more compressed the wavelet is.

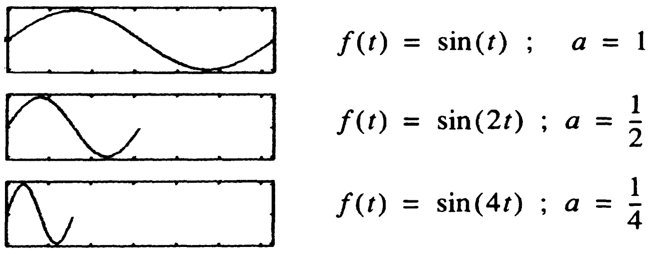


Figure 3.10 Scaling of a sinusoid function.

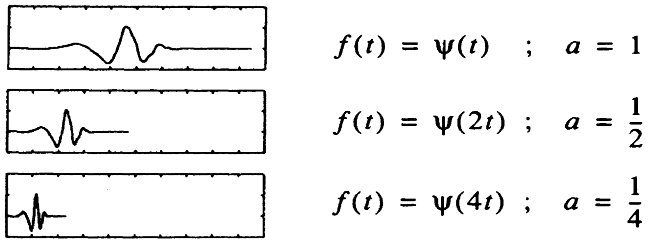


Figure 3.11 Scaling of a wavelet function.

The parameter b in Equation 3.19 is a translation along the time axis and simply shifts a wavelet and so delays or advances the time at which it is activated. Mathematically, delaying a function $f(t)$ by t_d is represented by $f(t-t_d)$. The factor $\frac{1}{\sqrt{a}}$ is used to ensure that the energy of the scaled and translated versions is the same as the mother wavelet. Figure 3.12 shows an example of a particular mother wavelet $\psi(t)$, known as the Mexican hat function [98],

$$\psi(t) = \frac{1}{\sqrt{3}}\pi^{-\frac{1}{4}}(1-t^2)\exp(-\frac{t^2}{2})$$

The stretched and compressed wavelets through scaling operation are used to capture the different frequency components of the function being analysed. The compressed version in Figure 3.12(b) is used to fit the high frequency needs, and the stretched version in Figure 3.12(c) is for low frequencies. The translation operation, on the other hand, involves shifting of the mother wavelet along the time axis to capture the time information of the function to be analysed at a different position, as shown in Figure 3.12 (d).

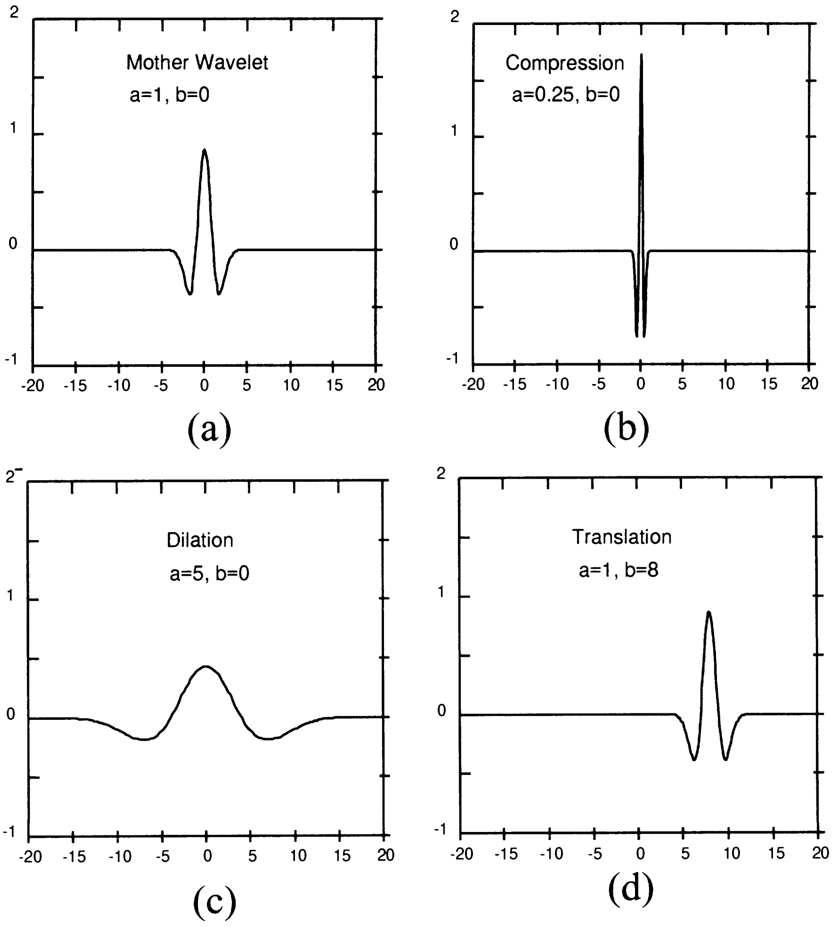


Figure 3.12 Mexican hat wavelet at different dilation and translation.

In this way, a family of scaled and translated wavelets can be created using scaling and translation parameters a and b. This allows signals occurring at different times which have different frequencies to be analysed.

In contrast to the short-time Fourier transform, which uses a single analysis window function, the wavelet transform can use short windows at high frequencies or long windows at low frequencies. Thus the wavelet transform is capable of zooming-in on short-lived high frequency phenomena, and zooming-out for sustained low frequency phenomena. This is the main advantage of the wavelet over the short-time Fourier transform.

3.3.3 Continuous Wavelet Transform

Given a mother wavelet function $\psi(t)$, the continuous wavelet transform CWT_f of a function $f(t)$ is defined by

$$CWT_f(a, b) = \langle f, \psi_{a,b} \rangle = \psi(\frac{t-b}{a})dt / \sqrt{a}, \qquad a, b \in R, \; a \neq 0 \tag{3.20}$$

where $\psi(\frac{t-b}{a})dt / \sqrt{a}$ is sometimes called the baby wavelet. Here time t and the scaling and translation parameters a and b can be changed continuously. In this case, $CWT_f(a, b)$ is called the wavelet transform coefficient. If a, b, and t change continuously, the values of $CWT_f(a, b)$ can be represented by a three dimensional diagram.

The application of a continuous wavelet transform consists of the following steps.

(1) take a wavelet and compare it to a section at the start of the original signal.

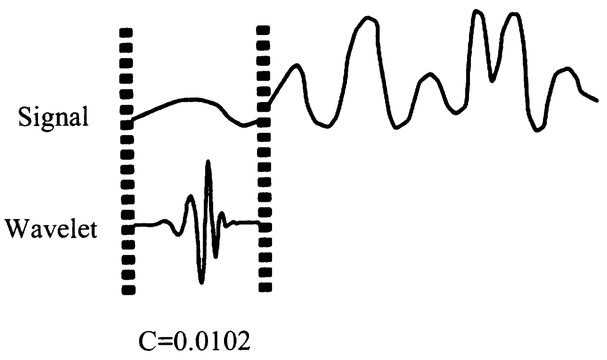


Figure 3.13 (a) Step (2) in continuous wavelet transform.

(2) Calculate the wavelet coefficient CWT_f, representing how closely the wavelet is related to this section of the signal, as shown in Figure 3.13(a). The higher CWT_f is, the greater the similarity. The result obviously depends on the shape of the wavelet chosen.

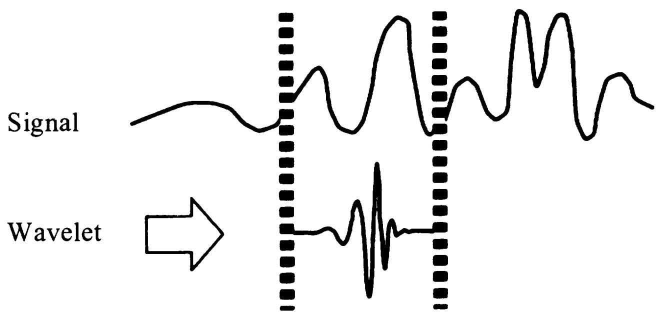


Figure 3.13(b) Step (3) in continuous wavelet transform.

(3) Shift the wavelet to the right and repeat steps 1 and 2 until all of the signal has been examined, as shown n Figure 3.13 (b).

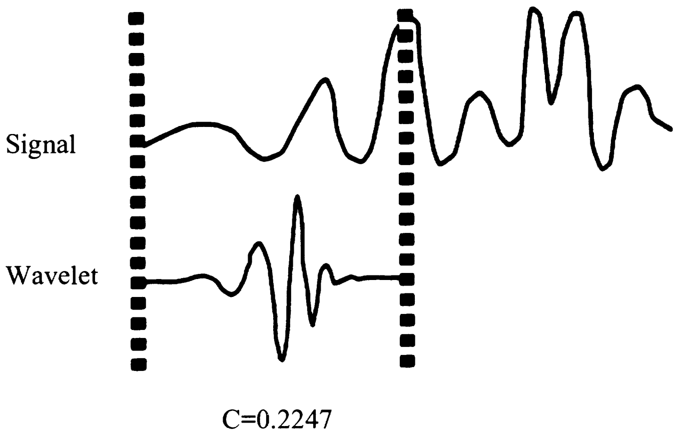


Figure 3.13(c) Step (4) in continuous wavelet transform.

(4) Scale (stretch) the wavelet and repeat steps 1 through 3 (Figure 3.13 (c)).

(5) Repeat steps 1 to 4 for all scales.

Plotting the wavelet coefficients against time and scale generates a three dimensional diagram, as shown in Figure 3.14. An alternative would be to use the two dimensional diagram in Figure 3.15, where brightness reflects the magnitude of the wavelet coefficients.

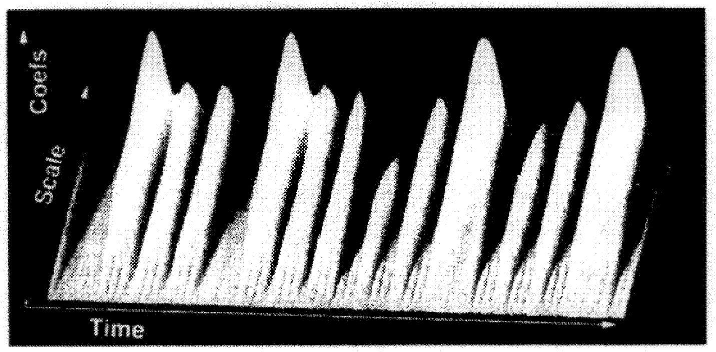


Figure 3.14 Three dimensional plot of wavelet transform coefficients.

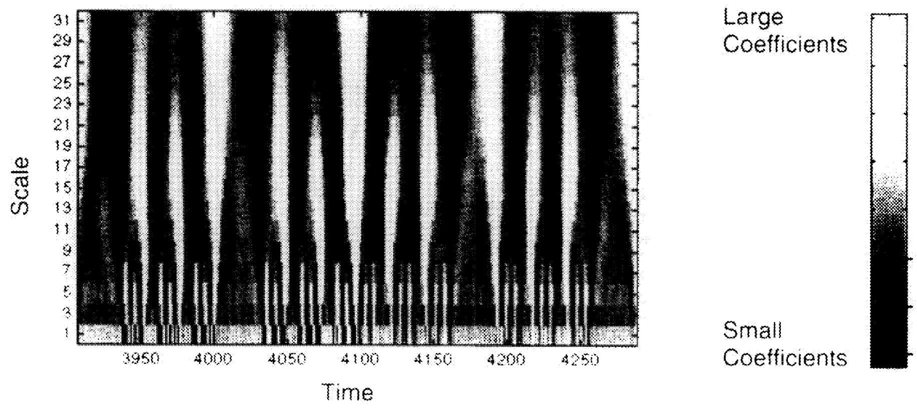


Figure 3.15 Two dimensional plot of wavelet transform coefficients.

Continuous in the context of wavelet transform implies that the scaling and translation parameters a and b change continuously. In practice, it is necessary to select a number of scales which is determined by the computational effort. A similar argument applies to the translation (shifting) parameter. In both cases, however, the data of the signal being processed by the continuous wavelet transform using a computer is discrete.

The wavelet transform coefficients which are generated can be used to reconstruct the original function.

$$f(t) = C_\psi^{-1} \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} CWT_f(a,b)\psi_{a,b}(t)\frac{dadb}{a^2} \tag{3.21}$$

where C_Ψ is called the admissibility constant defined by

$$\int_{-\infty}^{+\infty} \frac{|\psi(\omega)|^2}{\omega} d\omega = C_\psi < +\infty \tag{3.22}$$

3.3.4 Discrete Wavelet Transform

Calculating wavelet coefficients for every possible scale can represent a considerable effort and generate a vast amount of data. The discrete parameter wavelet transform uses scale and position values based on powers of two (so-called dyadic scales and positions) which makes the analysis much more efficient, while still being accurate. In practice, a discrete wavelet transform is commonly used. To do this, the scale and time parameters are discretised as follows,

$$a = a_0^m \ , \quad b = n\, b_0\, a_0^m \ , \quad m,\ n \text{ are integers} \tag{3.23}$$

The family of wavelets $\{\psi_{m,n}(t)\}$ is given by

$$\psi_{m,n}(t) = a_0^{-m/2}\ \psi(a_0^{-m}t - n b_0) \tag{3.24}$$

resulting in a discrete wavelet transform (DWT) having the form

$$DWT_f(m,n) = < f, \psi_{m,n} > = a_0^{-m/2} \int_{-\infty}^{+\infty} f(t)\psi(a_0^{-m}t - n b_0)dt \tag{3.25}$$

Mallat [89] developed an approach for implementing this using filters. For many signals, the low frequency content is the most important part. The high frequency content, on the other hand provides flavour or nuance. In wavelet analysis the low frequency content is called the *approximation* and the high frequency content is called the *detail*. The filtering process uses *lowpass* and *highpass* filters to decompose an original signal into the *approximation* and *detail* parts. It is not necessary to preserve all the outputs from the filters. Normally they are *downsampled,* keeping only the even components of the *lowpass* and *highpass* filter outputs. The decomposition can be iterated, with successive approximations being decomposed in turn, so that one signal is broken into many lower-resolution components, as illustrated in Figure 3.16. Here, S refers to the original signal, A is the approximation and D is the detail.

In the case of a discrete wavelet transform, reconstruction of the original signal is not guaranteed. Daubechies [88] developed conditions under which the $\{\psi_{m,n}\}$ form an orthonormal basis. Usually, $a_0 = 2$ and $b_0 = 1$ are used, though any values are possible. In this case, both the transform and reconstruction are complete because the family of wavelets satisfy the orthogonal condition.

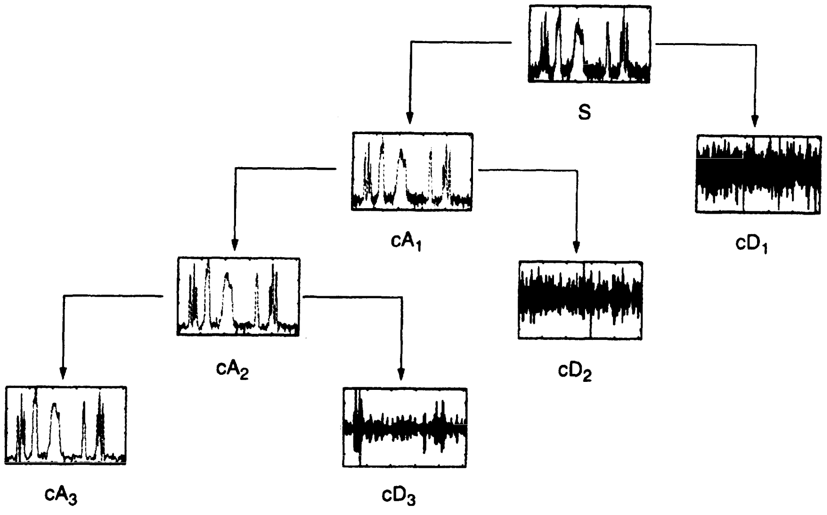


Figure 3.16 Signal multiresolution analysis using wavelets.

3.3.5 Singularity Detection Using Wavelet for Feature Extraction

Singularities often carry the most important information in signals [82]. For example, Bakshi and Stephanopoulos [90, 91] used inflexion points as the connection points of episode segments of a signal, as shown in Figure 3.17 (episode segments of a signal is discussed in Section 3.4). Singularities of a signal can be used as the compact representation of the original signal and used as inputs to pattern recognition systems.

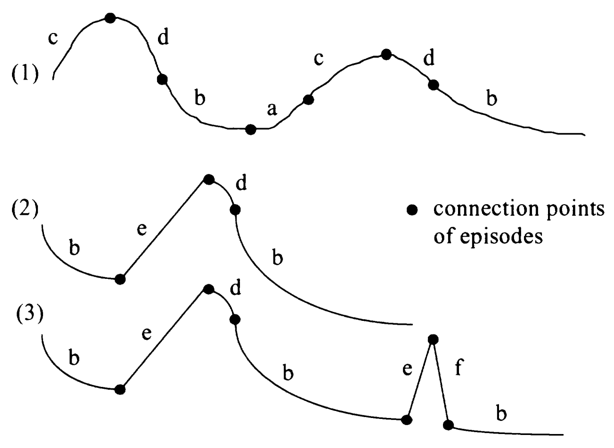


Figure 3.17 Singularities as connection points of episode segments.

Mathematically, the local singularity of a function is measured by Lipschitz exponents [92]. Mallat and Hwang [92] proved that the local maxima of the wavelet transform modulus detect the locations of irregular structures and provided numerical procedures for computing the Lipschitz exponents. Within the framework of scale-space filtering, inflexion points of $f(t)$ appear as extrema for $\partial f(t) / \partial t$ and zero crossing for $\partial^2 f(t) / \partial t^2$, so Mallat and Zhong [93] suggested using a wavelet which is the first derivative of a scaling function $\Phi(t)$,

$$\psi(t) = \frac{d\phi(t)}{dt}$$

with a cubic spine being used for the scaling function.

The wavelet modulus maxima and zero-crossing representations were developed from underlying continuous-time theory. For computer implementation, this has to be cast in the discrete - time domain. Berman and Baras [94] proved that wavelet transform extrema / zero-crossing provide stable representations of finite length discrete-time signals. Cvetkovic and Vetterli [95] have developed a more complete discrete-time framework for the representation of the wavelet transform. They designed a non-subsampled multi-resolution analysis filter bank to implement the wavelet transform for the representation. Using this filter bank, the wavelet function can be selected from a wider range than the B-spline in Mallat's method.

Non-subsampled multi-resolution analysis can then be used to detect singularities of a signal. An octave band non-subsampled filter bank with analysis filters $H_0(z)$ and $H_1(z)$ is shown in Figure 3.18. In this method, a wavelet transform is defined in terms of the bounded linear operators $W_j : l^2(Z) \rightarrow l^2(Z)$, j = 1, 2, ..J+1. The operators W_j are the convolution operators with the impulse responses of the filters:

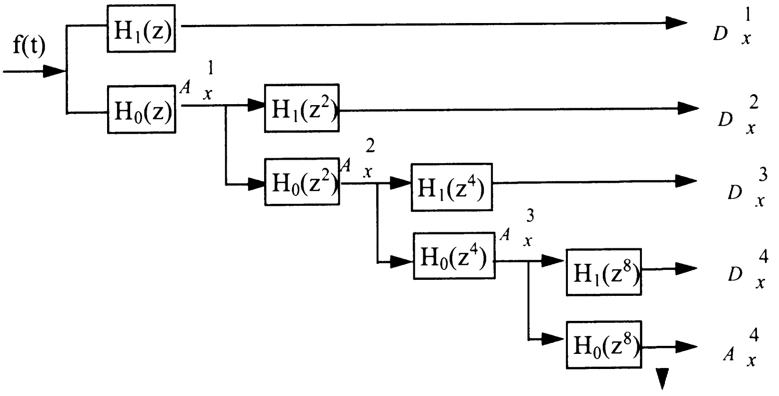
$$V_1(z) = H_1(z)$$

$$V_2(z) = H_0(z)H_1(z^2)$$

● ● ● ● ● ● ● ●

$$V_J(z) = H_0(z)\cdots H_0(z^{2^{J-2}})H_1(z^{2^{J-1}})$$

$$V_{J+1}(z) = H_0(z)\cdots H_0(z^{2^{J-2}})H_0(z^{2^{J-1}})$$



D_x^i -- detail on the ith decomposition

A_x^i -- approximation on ith decomposition

H_0, H_1 -- low-pass and high-pass filters

Figure 3.18 An octave band non-subsampled filter bank.

The multiresolution procedure depicted in Figure 3.18 can be described less rigorously. Figure 3.18 shows four steps, or four scales analysis. In the first step, the original signal is split into *approximation* A_x^1 and *detail* D_x^1. The *detail* D_x^1 is assumed to be mainly the noise components of the original signal and the *approximation* A_x^1 represents mainly the trend of the original signal. A_x^1 is further decomposed into *approximation* A_x^2 and *detail* D_x^2, A_x^2 to A_x^3 and D_x^3, and A_x^3 to A_x^4 and D_x^4. In each step we find the extrema of the *detail*. In the first few steps, the extrema are due to both the noise and the trend of the noise-free signal. As the scale increases, the noise extrema are gradually removed while the extrema of the noise-free signal remain. In this way, using multi-scale analysis and extrema determination, the extrema of the noise-free signal can be found, which represent the features of the signal.

To represent the extrema, it is convenient to use a finite impulse response (FIR) wavelet filter, which has a sequence $\{\alpha_k : k \in Z\}$ with only K non-zero terms. A typical example is the Haar wavelet, having only two non-zero coefficients. Daubechies's wavelets [88] are also FIR filters and smoother than the Haar wavelet. Daubechies' wavelets having more coefficients so are smoother and have higher vanishing moments. They also require less computation effort because they are constructed by filter convolution.

The Daubechies' scale and wavelet functions are expressed as

$$\phi(t) = \sum_k h(k)\phi(2t-k) \tag{3.26}$$

$$\psi(t) = \sum_k g(k)\phi(2t-k) \tag{3.27}$$

where $\{h(k)\}$ is the low-pass filter coefficients and $\{g(k)\}$ the band-pass filter coefficients.

Daubechies wavelets have a maximum number of vanishing moments over the support space. The vanishing moments of the wavelets also have a different number of coefficients. Using wavelets with more vanishing moments has the advantage of being able to measure the Lipschitz regularity up to a higher order, which is helpful in filtering noise, but it also increases the number of maxima lines. The number of maxima for a given scale often increases linearly with the number of moments of the wavelet. In order to minimise computational effort, it is necessary to have the minimum number of maxima to detect the significant irregular behaviour of a signal. This means choosing a wavelet with as few vanishing moments as possible but with enough moments to detect the Lipschitz exponents of the highest order components of interest.

For the cases considered here, an eight coefficient "least-asymmetric" Daubechies wavelet is used as a filter. The scale and wavelet function for this filter are illustrated in Figure 3.19.

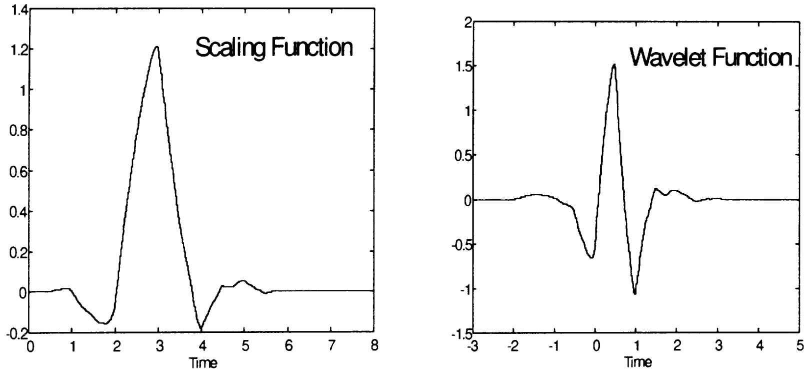
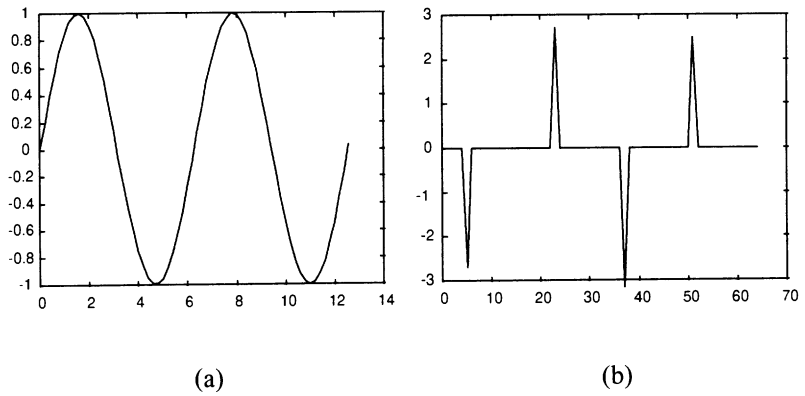


Figure 3.19 The "Least-Asymmetric" scale function and wavelet function.

A signal *f(t)=sin(t)* and its extrema from the wavelet analysis using a non-subsampled filter bank with Daubechies eight coefficients least asymmetry wavelet is illustrated in Figure 3.20. It shows that extrema of wavelet analysis correspond to the singularities of signal. The shape of corresponding extrema of the wavelet analysis can be maximum or minimum for the same signal singularity and depends on the wavelet used. In Figure 3.20(b), the wavelet is used as a filter, and the first singularity of the signal in Figure 3.20(a) corresponds to the minimum in the wavelet analysis. In Figure 3.21 it is a maximum because a different wavelet is employed. The former is used here.



(a) (b)

Figure 3.20 Signal (a) and its extrema (b) of the wavelet analysis based on Daubechies eight coefficients wavelets.

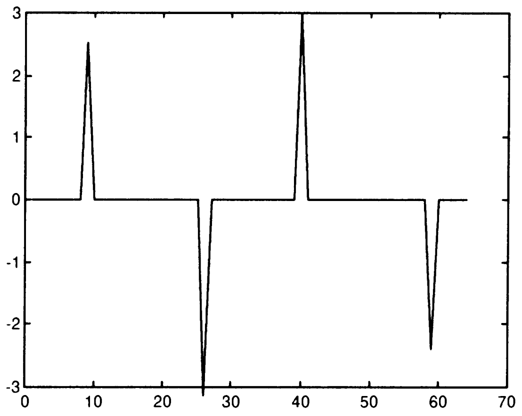


Figure 3.21 Extrema of wavelet analysis with Daubechies ten coefficients wavelet.

3.3.6 Noise Extrema Removal and Data Compression

The extrema obtained from wavelet multi-resolution analysis correspond to the singularities of the signal, which may also include those produced by noise, depending on the analysis scales. Therefore in feature extraction it is necessary to further filter out noise extrema from the wavelet transform. The most classical technique of removing noise from a signal is to filter it. Part of the noise is removed but it may also smooth the signal singularities at the same time. Mallat and Hwang [92], Mallat and Zhong [93] developed a technique for evaluating noise extrema in wavelet analysis. Some noise maxima increase on average when the scale decreases or do not propagate to larger scales. These are the modulus maxima which are mostly influenced by noise fluctuations.

Figures 3.22 and 3.23 illustrate this. In Figure 3.22, three different noise frequencies are studied. The wavelet multi-resolution analysis is shown on the left, and extrema of wavelet analysis are on the right. Clearly, the extrema will decrease and then disappear as the scale increases.

Figure 3.23 shows a noise signal which is basically the sine in Figure 3.20(a) plus white noise and the multi-resolution wavelet analysis elements. Noise components are reduced and then disappear as the scale increases. The results for scales 4 and 5 are similar to that of Figure 3.20(b) which has no noise. This shows that the extrema of the real trend are retained while noise extrema are filtered on the higher scales.

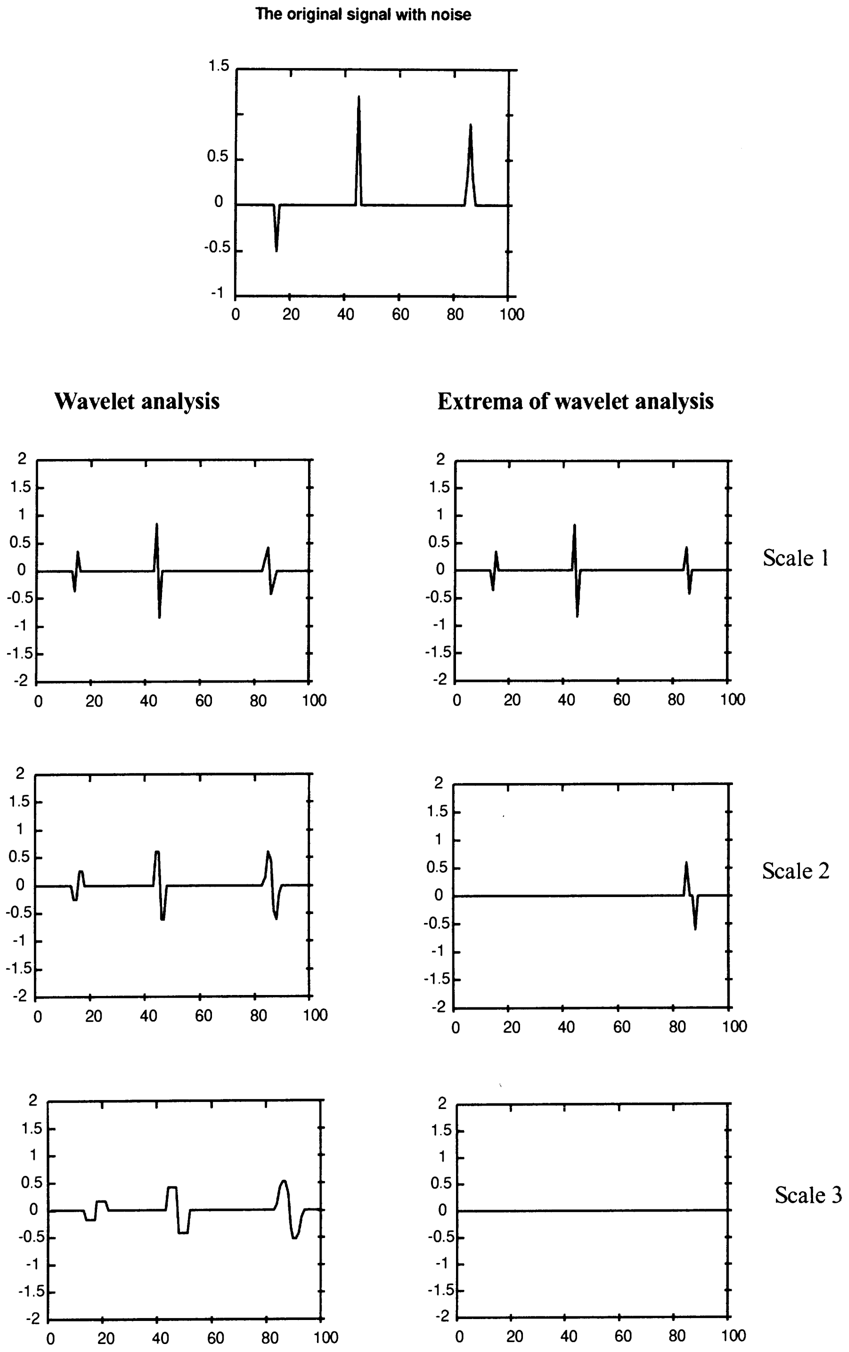


Figure 3.22 Noise signal, its wavelet transform and the extrema of wavelet transform.

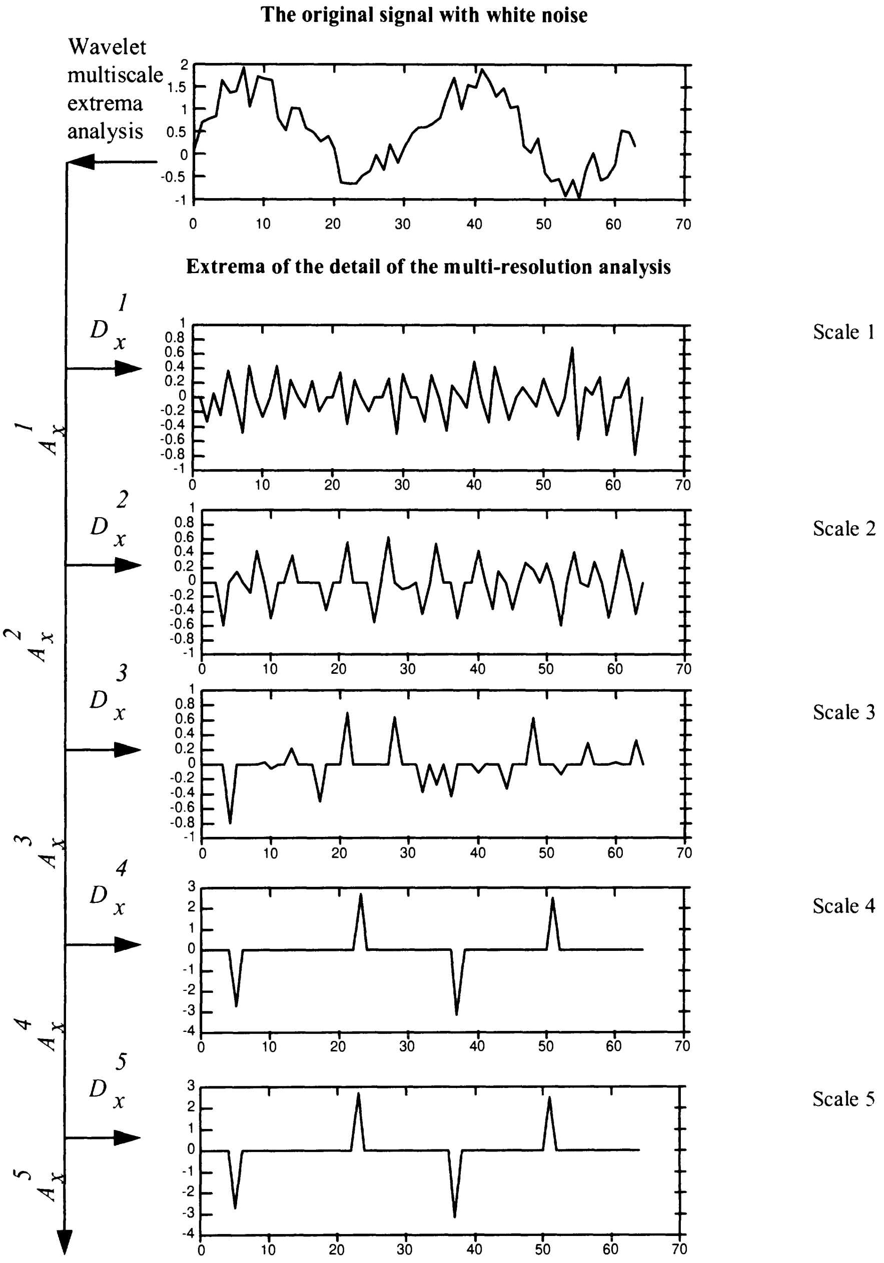


Figure 3.23 Noise signal and its multi-resolution analysis.
A_x^i - approximation of multiresolution analysis, D_x^i - detail.

3.3.7 Piece-wise Processing

Two observations can be made from the above discussion. Firstly, extrema analysis using wavelet multiresolution analysis remains steady with increase in scale. For example, in Figure 3.23 when the scale is increased from 4 to 5, the four extrema remain. Secondly, the location of extrema may slightly shift with time as the scale increases. In Figure 3.23, the extrema representation in scale 4 is a vector of dimension 70,

Scale-4 = (....x5.................x23.............x37...............x53.................)

where x5 stands for a non-zero datum.

While in scale 5, it becomes

Scale-5 = (......x7..............x22...............x38...............x54................)

The non-zero datum in the position 5 of scale-4 is shifted to the position 7 of scale-5. This inconsistency should be avoided. For instance, we should consider (2,0,...0,3) and (2,0...,3,0) to be different. This is necessary especially when considering the trends of a variable at different operational conditions.

The extrema representation results in very sparse vectors. This is true for dynamic responses at low frequencies. The method used here is termed piece-wise processing. The idea is to map a highly sparse vector to a denser vector by dimension reduction. For example, for the example of scale-4 and scale-5 discussed above, if the piece-wise sub-region is fixed as four data points, then scale-4 and scale-5 will be transformed to vectors of dimension 18.

Scale-4$'$ = (.x2...x6...x10..x13.....)
Scale-5$'$ = (.x2...x6...x10..x13.....)

It is clear that after piece-wise processing, the dimension is reduced and scale-4$'$ and scale-5$'$ are consistent. Therefore using a piece-wise processing technique, it is possible to achieve consistent feature extraction and reduction in dimension.

3.4 Episode Approach

This section describes qualitative interpretation of dynamic trends using the episode approach. It was developed earlier than PCA and wavelets and is straightforward. However it normally suffers from being week in dealing with noise.

The episode representation approach was originally developed by William [96]. Janusz and Venkatasubramanian [77] adapted it and used nine primitives to represent any plots of a function, as shown in Figure 3.24. Each primitive consists of the signs and the first and second derivatives of the function. This means, each primitive possesses information about whether the function is positive or negative, increasing, decreasing, or not changing, and the concavity. An episode is an interval described by only one primitive and the time interval the episode spans. A trend is a series of episodes that when grouped together can completely describe the qualitative states of the system. C and D in Figure 3.24 are actually not primitives because they can be regarded as the combination of A, F and B, E. Therefore they can be reduced to seven primitives as shown in Figure 3.25.

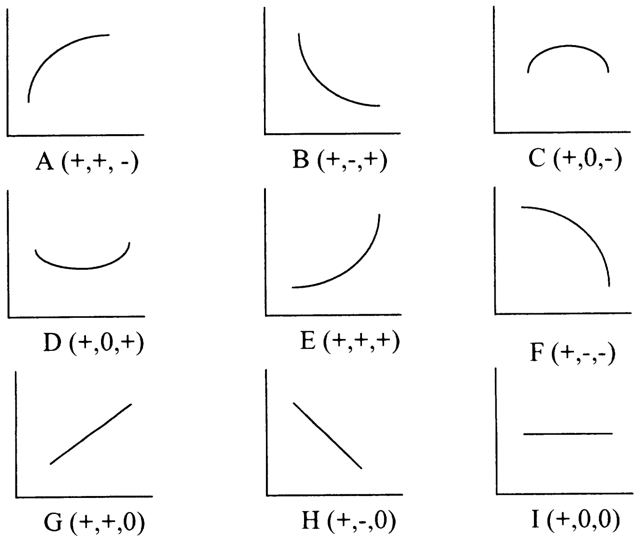


Figure 3.24 Nine primitives used in episode approach.

A combination of episodes will form a trend over an interval and is described by a primitive and the associated time. Primitives are different for first and/or second order derivatives, so the distinguishing points between episode segments are the extrema and inflexions where

$$\frac{\partial x}{\partial t} = 0 \quad \text{or} \quad \frac{\partial^2 x}{\partial t^2} = 0$$

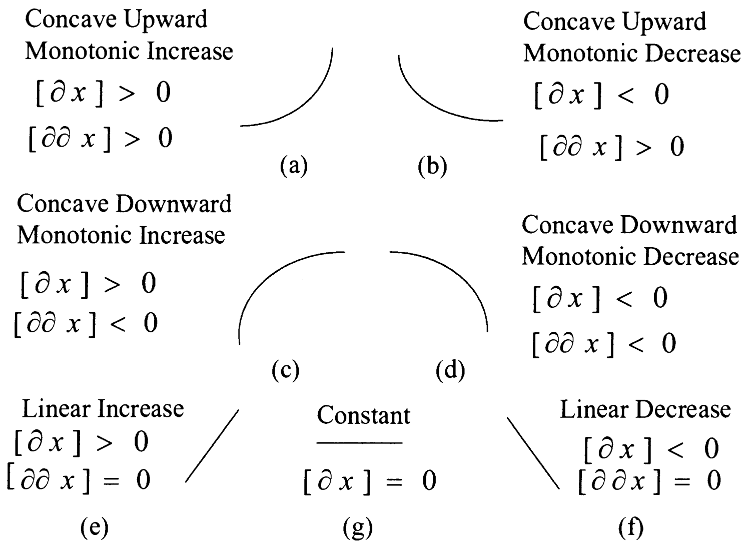


Figure 3.25 The seven primitives episodes.

This is also illustrated in Figure 3.17, where trend 1 consists of primitives c-d-b-a-c-d-b. The connection points c-d, b-a, are extrema, maximum or minimum points respectively, and inflexion points are between d and b, a and c, and d and b. Trend 2 in Figure 3.17 illustrates another case.

The task of identifying the episodes from a signal is simply to identify the inflexions and/or extrema, i.e., singularities in the signal since they correspond to distinct points of the episode segments. This means that the singularities of a signal contain the most important information about the trend. Using singularities for feature representation therefore completely defines the episodes characteristics of a signal.

However, the singularities are strongly influenced by noise and this is the major weakness of this approach. Noise components must be identified and filtered from the features, otherwise the representation will be misleading. Bakshi and Stephanopoulos [81, 90] used the wavelet approach developed by Mallat [97], Mallat and Zhong [93] and Mallat and Hwang [92] for detecting the inflection points, as described in sections 3.3.5 and 3.3.6.

3.5 Summary

In process operational state identification, dynamic transient signals are probably more important than the instant values of variables, therefore the discussion in this chapter has focused on pre-processing of transient signals. The purpose is to remove noise components, extract the features in a reduced data dimension and concept formation. The extracted features can then be used by various tools including multivariate analysis, supervised and unsupervised clustering as well as conceptual clustering which will be introduced in Chapters 4 to 7.

Signal pre-processing using wavelet multi-scale analysis is developed based on the fact that irregularities and singularities contain the most important information of trend signals. Since the extrema of wavelet transform of signals are able to capture all the irregularities and singularities of a signal when a filter bank and wavelet function are selected properly, they are regarded as the features of the trend. The advantage of being able to capture both the frequency and time features of a transient signal makes the wavelet feature extraction approach suitable not only for continuous but also batch operations. The approach described in this chapter has a number of advantages. Firstly, the extrema of wavelet multi-scale analysis can completely capture the distinguished points of a trend signal, because the original signals can be reconstructed. Secondly, the method is robust in the sense that the features captured do not change with the change of scales of analysis. Thirdly, the episode representation of a trend is primitive, there are no a priori measurements of compactness for the representation of extrema of wavelet multi-scale decomposition. In addition, a wavelet-based noise component removal procedure has been included so that noise effects can be filtered out. Wavelet analysis has also been used for model identification [98].

Principal component analysis offers an attractive alternative for pre-processing dynamic trend signals. This approach is particularly useful for concept formation.

For a specific variable, its dynamic responses under various disturbances or faults can be effectively discriminated by inspecting the location of a single data point on a PCA two dimensional plane. Concept formation is an essential step for conceptual clustering which, to be introduced in Chapter 7, is an approach that can develop a descriptive language for clustering operational states.

Episode based approaches are able to convert the signal information to qualitative descriptions however often suffer from the adverse effect of noise components. Available episode approaches have not addressed how they could deal with these effects.

CHAPTER 4
MULTIVARIATE STATISTICAL ANALYSIS FOR DATA ANALYSIS AND STATISTICAL CONTROL

There has been an increasing interest in applying multivariate statistics to analysis of historical databases of process operation and designing multivariate statistical control systems. The methods introduced in this chapter include principal component analysis (PCA), partial least squares (PLS), and multi-way and nonlinear PCA. The emphasis will be put on how these approaches can be applied to solving practical problems and addressing the advantages as well as limitations. The introduction to relevant mathematical background knowledge is less rigorous because there are already a large number of text books available.

4.1 PCA for State Identification and Monitoring

Principal component analysis (PCA) has been introduced in Chapter 3 as an approach for feature extraction and concept formation from dynamic transient signals. In this chapter, PCA is used to develop process monitoring systems through analysis of historical operational data.

4.1.1 Operational State Identification and Monitoring Using PCA

Process monitoring and diagnosis is conducted at two levels [8]: the immediate safety and operations of the plant usually monitored by plant operators, and the long-term performance analysis monitored by supervisors and engineers. Long term performance deterioration such as product quality degradation is considered as more difficult to be diagnosed than a sudden failure of equipment [102]. Trouble-shooting long term performance deterioration often requires examination of historical data which is large in volume and multivariate in nature. In a number of studies, PCA has proved to be a useful tool for this purpose. Here we use an example to illustrate the approach.

MacGregor and Kourti [103] reported an industrial case study of a continuous recovery process. The process consists of 12 separators to separate a feed of three major components A, B and C into three products, with the first product #1 comprising mainly component A. The main objectives of operating the plant are to maintain the concentration of component A in the product #1 to be at a specified level, i.e., no below 99.5 %, while achieving a certain minimum recovery, i.e., greater than 92%. However in the last three months from when the data was supplied by the company the recovery dropped significantly below 92%, as depicted by Figure 4.1.

The data to be analysed is daily averaged and covers 498 days of operation. Each data pattern has 447 process and 5 product variables. The purpose of the analysis is to find out why the recovery of product #1 has dropped and what we can do to move it back.

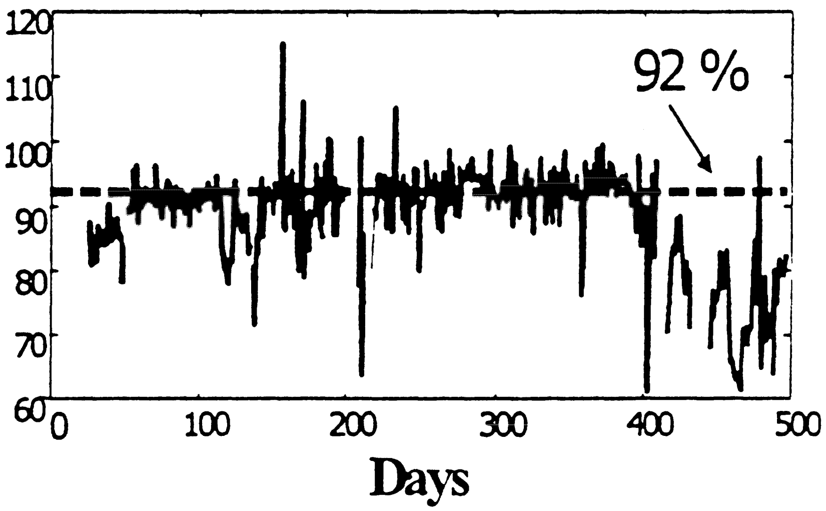


Figure 4.1 The recovery history of the product component A.

The first step involves PCA analysis of the data matrix of the 442 x 498 (process variables x number of data patterns). It was found that the first 7 PCs can explain 93% in purity variation and 93% in recovery. Projection of the first two PCs to a two dimensional plane indicated that for the last three months (data points 401 to 490), where the recovery was low, the process operation had changed to a new operational state. It clearly explains the reason for dropped recovery over the last three months, which would have been very difficult to find out without this multivariate analysis.

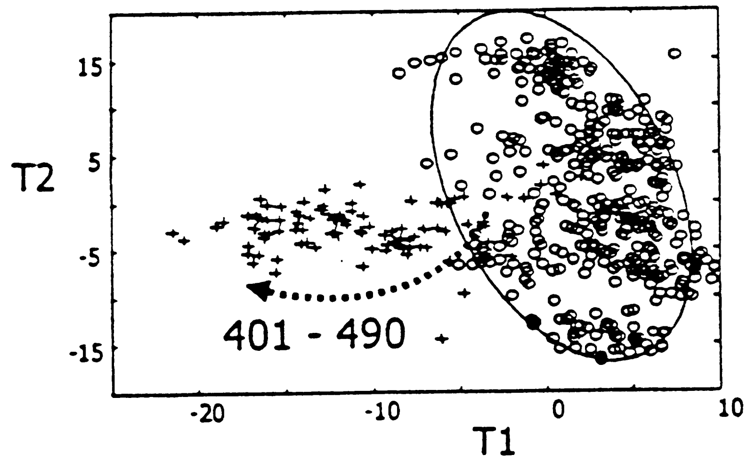


Figure 4.2 Projection of process operational history on a PC two dimensional plane.

If we regard the area in the circle of Figure 4.2 as normal operation, then this diagram can also be used for a multivariate statistical monitoring. If the operating point goes outside the region then the operation can be regarded as abnormal. Kresta et al. [104] gave other examples using the same approach to design statistical monitoring systems for a fluidised bed reactor and a binary distillation column.

4.1.2 Multivariate Quality Control Charts Based on PCA

4.1.2.1 Multivariate Shewhart Charts

Traditionally, univariate statistical process control charts, e.g., the Sherwhart chart shown in Figure 1.2 have been used in industry to separately monitor either a few process variables, or key measurements on the final product which in some way define the quality of the product. The difficulty with this approach is that these quality variables are not independent of one another nor does any of them adequately define product quality by itself. The difficulties with using independent univariate control charts can be illustrated by reference to Figure 4.3. Here only two quality variables (y_1, y_2) are considered for ease of illustration. Suppose that, when the process is in a state of statistical control where only common course variation is present, y_1, and y_2 follow a multivariate normal distribution and are correlated ($\rho_{y_1,y_2} = 0.8$) as illustrated in the joint plot of y_1 vs. y_2 in Figure 4.3.

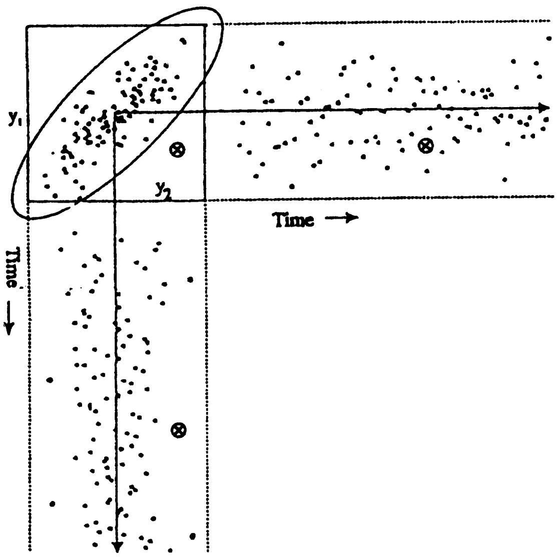


Figure 4.3 Quality control of two variables illustrating the misleading nature of univariate charts.

The ellipse represents a contour for the in-control process, and the dots represent a set of observations from this distribution. The same observations are also plotted in Figure 4.3 as univariate Shewhart charts on y_1, and y_2 vs. time with their corresponding upper and lower limits. Note that by inspection of each of the individual Shewhart charts the process appears to be clearly in a state of statistical control. The only indication of any difficulty is that a customer has complained about the performance of the product corresponding to the ⊕ in Figure 4.3. The true situation is only revealed in the multivariate y_1 vs. y_2 plot where it is seen that the product indicated by the ⊕ is clearly outside the joint confidence region, and is clearly different from the normal "in-control" population of product.

In multivariate data analysis, the ellipse in Figure 4.3 is determined by calculating the Mahalanobis distance [106]. In the discussion of bivariate samples the quantity

$$x^2 = \begin{bmatrix} y_1-\bar{y}_1 \\ y_2-\bar{y}_2 \end{bmatrix}' \begin{bmatrix} s_1^2 & s_{12} \\ s_{12} & s_2^2 \end{bmatrix}^{-1} \begin{bmatrix} y_1-\bar{y}_1 \\ y_2-\bar{y}_2 \end{bmatrix} \tag{4.1}$$

is often used to describe the locus of an ellipse in two-dimensional space with centre $(\bar{y}_1, \bar{y}_2)$. This quantity also measures the square of the Mahalanobis distance between the point (y_1, y_2) and the centre $(\bar{y}_1, \bar{y}_2)$. All points on this ellipse have the same distance m^2 from $(\bar{y}_1, \bar{y}_2)$.

This can be extended to multivariate situation and used for designing multivariate Shewhart charts, called x^2 and T^2 charts for statistical process control. This was originated by Hotelling [107] and several references discussed the charts in more detail [103, 105, 108, 109, 110].

Given a $(k \times 1)$ vector of variables $\boldsymbol{y}$ on k normally distributed variables with a covariance matrix Σ, the Mahalanobis distance from the centre equivalent to Equation 4.1 is

$$x^2 = (\boldsymbol{y}-\tau)^{\mathrm{T}} \Sigma^{-1} (\boldsymbol{y}-\tau) \tag{4.2}$$

Σ is also called in-control covariance matrix. If Σ is not known, it must be estimated from a sample of n past multivariate observations as

$$\boldsymbol{S} = (n-1)^{-1} \sum_{i=1}^{n} (y_i - \bar{y})(y_i - \bar{y})^{\mathrm{T}} \tag{4.3}$$

When new multivariate observations ($\boldsymbol{y}$) are obtained, then Hotlling's T^2 statistic is given by

$$T^2 = (\boldsymbol{y}-\tau)^{\mathrm{T}} \boldsymbol{S}^{-1} (\boldsymbol{y}-\tau) \tag{4.4}$$

T^2 can be plotted against time. An upper control limit (UCL) on this chart is given by [105],

$$T_{UCL}^2 = \frac{(n-1)(n+1)k}{n(n-k)} F_\alpha(k, n-k) \tag{4.5}$$

where $F_\alpha(k, n-k)$ is the upper 100α% critical point of the F distribution with k and n-k degrees of freedom [111].

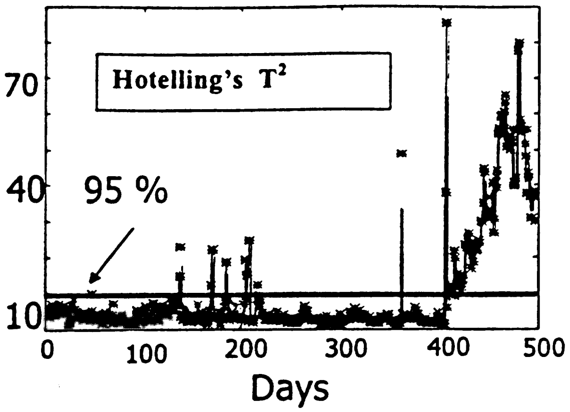


Figure 4.4 The Hotelling's T^2 chart for the recovery process indicating a deviation from normal operation around the 400th day

4.1.2.2 Multivariate Quality Control Charts Based on PCA

Since many of the process variables are autocorrelated, only a few underlying events are driving a process at any time, and all these measurements are simply different reflections of these same underlying events. Principal component analysis therefore can be applied to process the data first before the multivariate Shewhart charts are used. It means the latent variables, i.e., the first few PCs are used rather than the original variables. If the first A PCs are used, then the Hotlling's T^2 can calculated by [105],

$$T_A^2 = \sum_{i=1}^{A} \frac{t_i^2}{s_{t_i}^2} \tag{4.6}$$

where $s_{t_i}^2$ is the estimated variance of t_i. If A =2, a joint t_1 vs. t_2 plot can be used.

Note that the traditional Hottelling T^2 in Equation 4.4 is equivalent to

$$T^2 = \sum_{i=1}^{A} \frac{t_i^2}{s_{t_i}^2} + \sum_{i=A+1}^{k} \frac{t_i^2}{s_{t_i}^2} \tag{4.7}$$

For the recovery process, the T^2 chart is shown in Figure 4.4. The 95% confidence limit was determined based on good operation where the recovery is around 92%. Had the chart been on-line the deviation from normal could have been detected based on process data only, immediately when it occurred around observation 400.

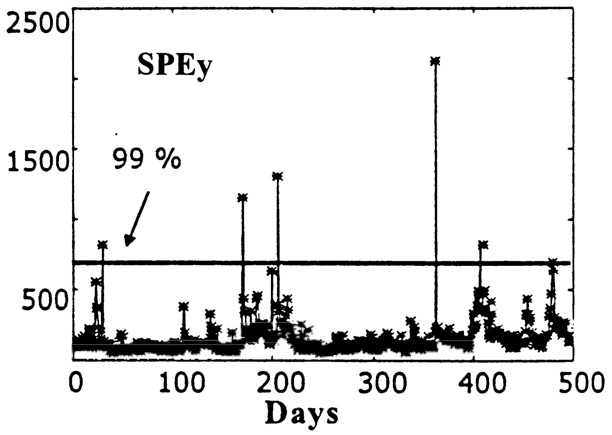


Figure 4.5 The SPEy chart for the recovery process.

However, monitoring product quality via T^2 based on the first A PCs is not sufficient. This will only detect whether or not the variation in the quality variables in the plane of the first A PCs is greater than can be explained by common cause. If a totally new type of special event occurs which was not present in the reference data used to develop the in-control PCA model, the new PCs will appear and the new observation y_{new} will move off the plane. Such new events can be detected by computing the squared prediction error (SPE$_y$) of the residual of a new observation,

$$\text{SPE}_y = \sum_{i=1}^{k} (y_{new,i} - \hat{y}_{new,i})^2 \tag{4.8}$$

where $\hat{y}_{new}$ is computed from the reference PCA model. SPE$_y$ is also refereed to as Q statistic or distance to the model. It represents the squared perpendicular distance of a new multivariate observation from the projection space. Figure 4.5 shows the SPE$_y$ chart for the recovery process. The 99% confidence limit was determined based on good operation when the recovery is around 92%.

When the process is "in-control", this value of SPE$_y$ should be small. Therefore a very effective set of multivariate statistical control charts is a T^2 chart on the A dominant orthogonal PCs ($t_1, t_2, ..., t_A$) plus a SPE$_y$ chart.

4.2 Partial Least Squares (PLS)

Given two matrices, an ($n \times m$) process data matrix **X**, and an ($n \times k$) matrix of corresponding product quality data **Y**, one would like to extract latent variables that not only explain the variation in the process data (**X**), but that variation in **X** which is most predictive of the product quality data (**Y**). PLS is a method which accomplishes this by working on the sample covariance matrix $(\mathbf{X}^T\mathbf{Y})(\mathbf{Y}^T\mathbf{X})$. In the most common version of PLS, the first PLS latent variable $t_1 = \boldsymbol{w}_1^T \boldsymbol{x}$ is that linear combination of the x variables that maximises the covariance between it and the **Y** space. The first PLS loading vector $\boldsymbol{w}_1$ is the first eigenvector of the sample covariance matrix $\mathbf{X}^T\mathbf{Y}\mathbf{Y}^T\mathbf{X}$. Once the scores $\boldsymbol{t}_1 = \mathbf{X}\boldsymbol{w}_1$ for the first component have been computed the columns of **X** are regressed on $\boldsymbol{t}_1$ to give regression vector $\boldsymbol{p}_1 = \mathbf{X}\boldsymbol{t}_1 / \boldsymbol{t}_1^T \boldsymbol{t}_1$ and the **X** matrix is deflated to give residuals $\mathbf{X}_2 = \mathbf{X} - \boldsymbol{t}_1 \boldsymbol{p}_1^T$. The second latent variable is then computed as $\mathrm{t}_2 = \boldsymbol{w}_2^T \boldsymbol{x}$ where $\boldsymbol{w}_2$ is the first eigenvector of $\mathbf{X}_2^T \mathbf{Y}\mathbf{Y}^T\mathbf{X}_2$ and so on. As in PCA, the new latent vectors or scores ($\boldsymbol{t}_1, \boldsymbol{t}_2, ...$) and the loading vector ($\boldsymbol{w}_1, \boldsymbol{w}_2, ...$) are orthogonal.

4.3 Variable Contribution Plots

Although the T^2 and SPE$_y$ charts are powerful ways for detecting deviations from normal operations, they do not indicate reasons for such deviations. This can be achieved by plotting variable contribution plots. There are two alternative ways of doing this which will be illustrated by reference to a case study described by MacGregor et al. [113], a low-density polyethylene tubular and autoclave reactor. A database of dimensionality of 55×14 (number of observations $\times$ number of process variables) was analysed using PCA and PLS. The PC two dimensional plane of PLS analysis as well as the SPE$_y$ plot are shown in Figures 4.6 and 4.7. From Figure 4.6 it can be seen that from data point 53 the operation deviates from normal operation. The deviation point is detected at point 54 on the SPE$_y$ plot. This difference is not significant and our focus here is on how to find out which variable is the main factor contributing to the deviation.

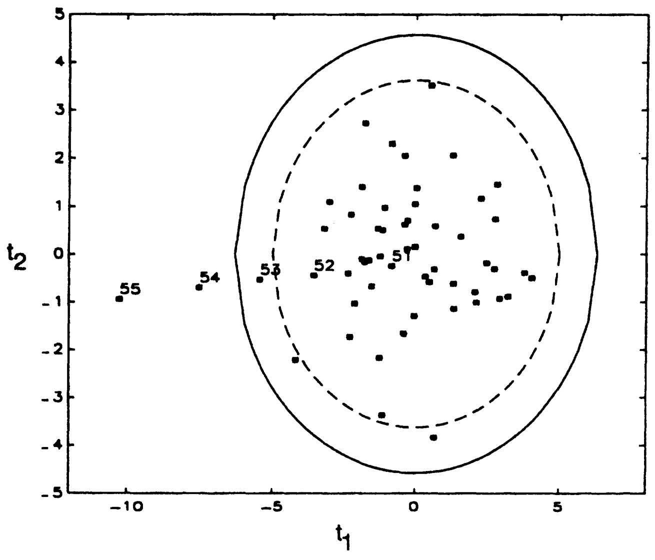


Figure 4.6 PLS t_1-t_2 plane for the low-density polyethylene reactor indicating deviation from data pint 54.

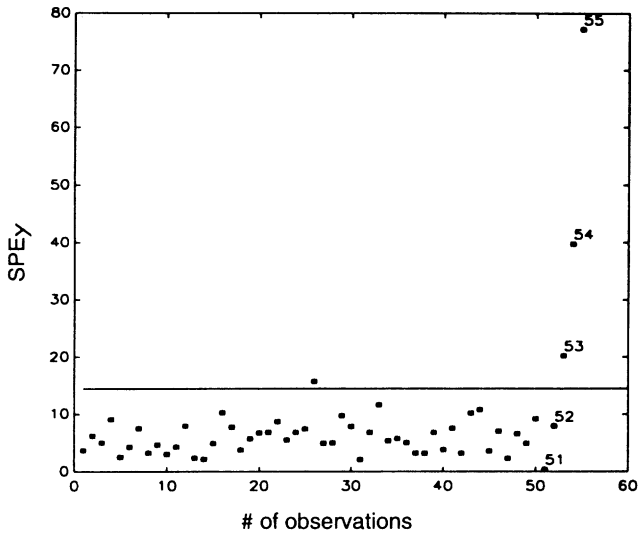


Figure 4.7 PLS SPE$_y$ chart for the low-density polyethylene reactor indicating deviation from data pint 53.

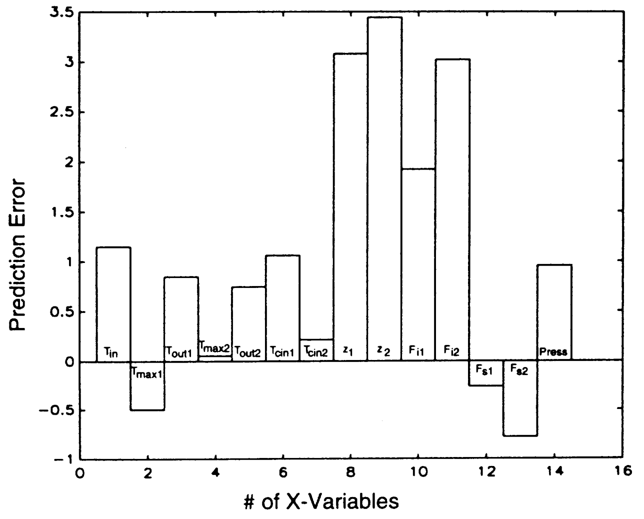


Figure 4.8 PLS prediction errors in the individual process variables contributing to SPE$_y$ at time point 54.

One way is to plot the SPE prediction error for the deviation point, say point 54, against the process variables, as shown in Figure 4.8.

To diagnose the event one can examine the contributions of the individual variables to this larger than normal value of the SPE$_y$ at point 54 that is,

$$\mathrm{SPE}_{x,54} = \sum_{j=1}^{k} (x_{54,j} - \hat{x}_{54,j})^2$$

Here the predictions $\hat{x}_{54,j}$ are made from the PLS model developed for the "in-control" operating data as,

$$\hat{x}_{54,j} = \sum_{a=1}^{A} t_{a,54} p_{aj}$$

where the new latent variable projections for the 54th observation are given by,

$$t_{a,54} = \sum_{j=1}^{k} w_{a,j} x_{54,j}$$

From Figure 4.8 it is clear that the major contributing variable is z_2.

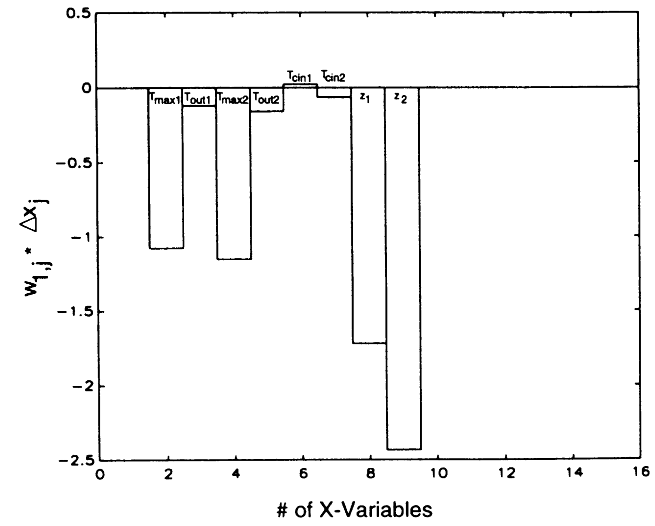


Figure 4.9 PLS variable contributions to the change in t_l from point 51 to 54.

An alternative way of diagnosing the event is note that in the latent variable plane Figure 4.6, the deviation of point 54 is mainly due to a large decrease in t_1. Therefore we can analyse the importance of each process variable to this latent variable through examining the loading vector w_l. The contribution of each variable (x_j) to this large movement in t_1 can be computed as $\{(w_1 \bullet \Delta x_j); \text{j} = 1, 2, ..., \text{k}\}$ where $\Delta x_j = (x_{54,j}-x_{51,j})$. These contributions are shown in Figure 4.9. It confirms that the main contributing variable is z_2.

4.4 Multiblock PCA and PLS

If the number of variables is not large, generally, the first two or three PCs are sufficient to capture most of the variance in the data [105, 114]. However if the number of variables is large, it is necessary to consider more PCs. Increased number of PCs makes the interpretation of results more difficult. First of all, the two or three dimensional PC plane, such as Figure 4.2 can not be used. Secondly the variable contribution analysis becomes difficult. A variation of multiway PCA and PLS [115] was proposed by MacGregor et al. [113]. The idea is to divide the process variables (**X**) into a number of blocks and then perform PCA or PLS analysis for each block as well as for the entire process. The blocks can be arranged based on functional and structural analysis of the process to be analysed and the interactions between blocks should be minimised. MacGregor et al. [113] discussed the procedures by reference to a low-density polyethylene process.

4.5 Batch Process Monitoring Using Multiway PCA

The above discussed case studies are two-way arrays: the variables and the observations. In some cases the data takes the form of three-way arrays. Data about batch process operations is such a case [116]. Batch production consists of batch runs, one after another. For a typical batch run, j=1, 2, …, J variables are measured at k=1,2,…,K time intervals throughout the batch. Suppose the data consists of i = 1, 2, …, I such batch runs, then the database will be a three-way array $\underline{X}(I{\times}J{\times}K)$, as illustrated in Figure 4.10, where different batch runs are organised along the vertical side, the measurement variables along the horizontal side, and their time evolution occupies the third dimension. Each horizontal slice through this array is a ($J{\times}K$) data matrix representing the time histories or trajectories for all the variables for

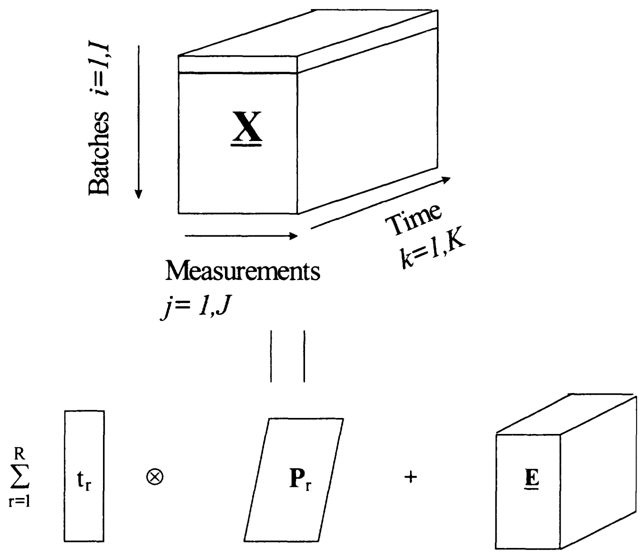


Figure 4.10 Arrangement and decomposition of a three-way array by multi-way PCA.

single batch, the *i*th batch. Each vertical slice is an (*I*×*J*) matrix representing the values of all the variables for all the batches at a common time interval *(k)*.

The multi-way PCA (MPCA) approach proposed by Wold et al. [115] is statistically and algorithmically consistent with PCA and has the same goals and benefits. The relation between MPCA and PCA is that MPCA is equivalent to performing PCA on a large two-dimensional matrix formed by unfolding the three-way array $\underline{X}$ in one of the three possible ways. For analysis of batch process operational data, Nomikos and MacGregor [116] unfolded the data in such a way as to put each of its vertical slices (*I*×*J*) slide by slide to the right, starting with the one corresponding to the first time interval. The resulting two-dimensional matrix has dimensions (*I*×*JK*). This unfolding allows us to analyse the variability among the batches in $\underline{X}$ by summarising the information in the data with respect both to variables and their time variation.

4.6 Nonlinear PCA

Though PCA has been widely studied and applied successfully to many application areas including chemistry, biology, meteorology and process engineering etc., there is criticism on this approach being a linear operation. Some researchers have pointed out that PCA can be inadequate in solving some nonlinear problems [117, 118]. Xu et al. [118] has given an example showing that when PCA is applied to a nonlinear problem, minor PCs do not always consist of noise or unimportant variance. If they are discarded, important information is lost; if they are kept, the large number of PCs make the interpretation of results difficult. As a result nonlinear PCA has attracted the interest of some researchers as summarised by Dong and McAvoy [119].

One way of introducing nonlinearity into PCA is the "generalised PCA" by Gnanadesikian [120]. The basic idea of this approach is to extend an m-D variable X to include nonlinear functions of its elements. For example, for two dimensions $X=(x_1, x_2)$, three variables can be added: $x_3=x_1^2$, $x_4=x_2^2$, and $x_5 = x_1x_2$. Then one can do the same calculations as linear PCA on this 5-D data. Another way of introducing nonlinearity into PCA is "nonlinear factor analysis" [121]. In this method l-D polynomials are used to approximate m-D data with $l<m$. A linear least squares method is used to find the coefficients of the polynomials. Dong and McAvoy [119] commented on the method that for high-dimensional data, it becomes tedious.

Kramer [122] presented a nonlinear principal component analysis method based on autoassociative neural networks. The architecture of the network is shown in Figure 4.11. It has five layers, i.e., the input, mapping, bottleneck, de-mapping and output layers, and its input is used as the desired output. The network is therefore supervised in nature and can be taught with a backpropagtion learning algorithm.

As no assumptions are needed about the nature of nonlinearity between the variables, the network can be used in situations where common transformations (e.g., logarithm, square root) can not be used. The nonlinearity is introduced into the network by sigmoidal transfer functions in the mapping and de-mapping layers.

The bottleneck layer will perform the dimension reduction, because the number of neurons in this layer is smaller than that in the input and output layers, so that the network is forced to develop a computer representation of the input data. The goal of the network is to minimise the error function

$$E = \sum_{j=1}^{n}\sum_{i=1}^{m} (X_i - X_i')_j^2 \qquad (4.9)$$

where X_i is an observation in the data set X and X_i' is an output of the network.

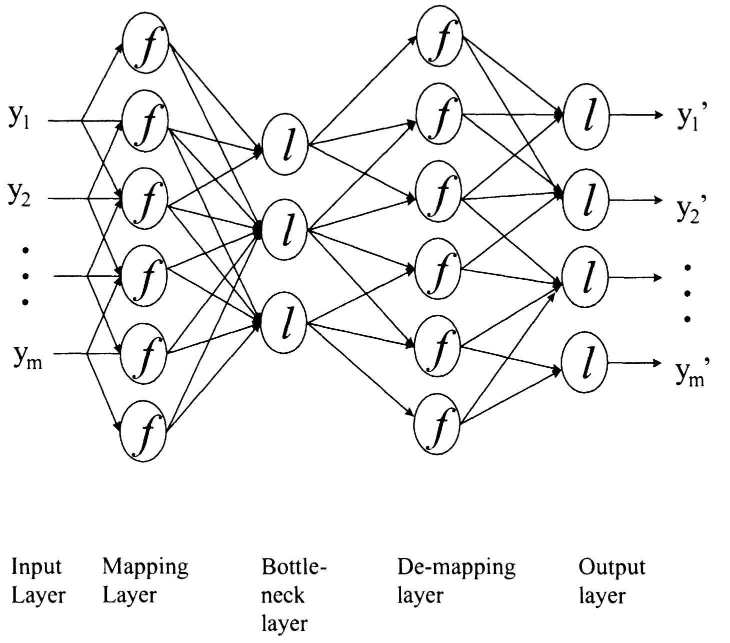


Figure 4.11 An autoassociative nonlinear PCA network with five layers. Transfer functions f are nonlinear and transfer functions l are linear.

Questions can be raised on this approach. An important point is that it is unknown if the outputs of the bottleneck layer, i.e., the principal components are linearly or non-linearly independent. It is very clear in linear PCA, the PCs are linearly independent. Dong and McAvoy raised other questions. Since there are five layers training of the network will be difficult. It is also difficult to determine the number of nodes in the mapping, bottleneck and de-mapping layers. In addition, the theoretical meaning of the outputs of the bottleneck layer, i.e., the principal components is not clear. Nevertheless, the approach is still proved to be very effective in reducing dimension as demonstrated by Kramer [122] in case studies.

4.7 Operational Strategy Development and Product Design - an Industrial Case Study

This section describes an industrial case study of analysing historical data using PCA for operational strategy development and product design [135].

4.7.1 The FCC Main Fractionator and Product Quality

The fluid catalytic cracking process (FCC) of the refinery converts a mixture of heavy oils into more valuable products. The relevant section of the process is shown in Figure 4.12, where the oil gas mixture leaving the reactor goes into the main fractionator to be separated into various products. The individual side draw products are further processed by down stream units before being sent to blending units.

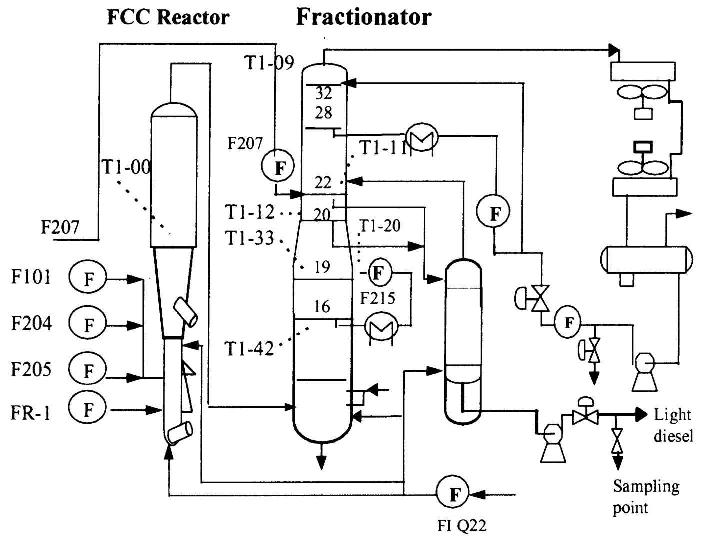


Figure 4.12 The main fractionator of the FCC process.

One of the product is light diesel whose quality is typically characterised by the temperature of condensation. Traditionally the temperature of condensation has been monitored by off-line laboratory analysis, which caused time delays because the interval between two samples is between four to six hours. As a result a software sensor has been developed using 303 data patterns spanning over nearly a year for predicting the condensation point using fourteen process variables which are measured on-line (a detail discussion of the software development is given in Chapter 9). The fourteen variables are listed in Table 4.1.

An interesting problem with the process is that it is required to produce three product grades according to seasons and market demand, namely -10#, 0# and 5# defined by the ranges of condensation temperature. Because there are more than one process variable the operators use their experience through trial-and-error to adjust process variables to move the operation from producing one product grade to another. There is a clear need to minimise the time of change over because off-specification product may be produced during transition.

Table 4.1 The fourteen variables used as input to the FFNN model.

T1-11	- the temperature on tray 22 where the light diesel is withdrawn
T1-12	- the temperature on tray 20 where the light diesel is withdrawn
T1-33	- the temperature on tray 19
T1-42	- the temperature on tray 16, i.e., the initial temperature of the pumparound
T1-20	- the return temperature of the pumparound
F215	- the flowrate of the pumparound
T1-09	- column top temperature
T1-00	- reaction temperature
F205	- fresh feed flowrate to the reactor
F204	- flowrate of the recycle oil
F101	- steam flowrate
FR-1	- steam flowrate
FIQ22	- flowrate of the over-heated steam
F207	- flowrate of the rich-absorbent oil

4.7.2 Knowledge Discovery using PCA

The difficulty of the problem comes from the fact that there are fourteen process variables to consider. Application of PCA to the database of the size 303×14 (number of data patterns×number of process variables) found that the first seven variables account for about 93% of the variance (Table 4.2). The PC1 and PC2 two dimensional plot is shown in Figure 4.13. It was found that the 303 data patterns are grouped into four clusters. Three clusters correspond to three products -10#, 5# and 0# and the cluster at the bottom-right corner is found to be a cluster that has a high probability of product off-specification. Before we analyse how this can be used to develop operational strategies it is necessary to validate the clustering result since the first two PCs only account for 53% of the variance. For this purpose, the first three PCs are plotted in a three dimensional diagram (Figure 4.14). It is found that the cluster at the centre of Figure 4.13 is further divided into two clusters. Using the first seven PCs, ART2 gives a similar result as indicated by the dotted curve in Figure 4.13. This demonstrates that for problems having large dimensions, clusters may overlap in a two dimensional PC display. Nevertheless, for the current problem, it is found that the two clusters at the centre of Figure 4.14 both correspond to product 0#. As a result, in the following discussion, we still use the result of Figure 4.13.

Therefore the strategy for operation and product design should be to operate the process in the region of the bottom-left if the desired product is -10#, or the region at the top if the desire product is 5#, or the region at the middle if the desired product is 0#, and try to avoid the region at the bottom-right corner. Another point is that to move from producing -10# to 0#, adjusting PC1 is more important than changing PC2. While to switch from producing 0# to 5#, PC2 is more important than PC1. Both PC1 and PC2 are important in avoiding the region at the bottom-right corner which produces off-specification product.

However, PC1 and PC2 are latent variables. To link PC1 and PC2 to the original variables, contribution plots are used. The contribution plot of PC1 is shown in Figure 4.15, from which it is found that the most important variables are TI12 (the temperature on tray 20 where the product is withdrawn) and TI42 (the temperature on tray 16 close to the flashing zone). Some other variables are not important such as FR-1. The above discovery is confirmed by looking at the change of TI-12 over the 303 data patterns (Figure 4.16). It clearly shows that TI-12 can distinguish product -10# from 0# and 5#, but can not distinguish 0# and 5#.

The contribution plot of PC2 is shown in Figure 4.17 which indicates that FR-1 is the most important variable. The changing profile of FR-1 for the 303 data patterns are shown in Figure 4.18. It clearly shows that FR-1 can distinguish product 5# from 0# and -10#, but not between 0# and -10#. The figure also confirms that FR-1 is not important to PC1.

Therefore the operational strategy for product design should be that if we want to change from producing -10# to 5#, we should increase TI-12 and TR-42 and then increase FR-1. In order to avoid off-specification product we should carefully monitor TI-12, TR-42 and FR-1 to avoid the region at the bottom-right corner. Of course it is important to be aware that fine tuning of all the variables is necessary but this guidance can help operators to move the process from producing one product quickly to another product. The PC1 and PC-2 two dimensional plane can also be used by operators as a monitoring screen as demonstrated by Figure 4.19.

Table.4.2 Summary of the PCs

PCs	Eigenvalues	Percentage of Eigenvalues	Cumulated Percentage
1	5.0729	36.2352	36.2352
2	2.3663	16.9017	53.1369
3	1.9453	13.8953	67.0322
4	1.1271	8.0506	75.0828
5	1.0382	7.4155	82.4983
6	0.7757	5.5406	88.0389
7	0.5845	4.1751	92.2140
8	0.4485	3.2032	95.4173
9	0.3034	2.1670	97.5843
10	0.1350	0.9641	98.5484
11	0.0795	0.5679	99.1163
12	0.0682	0.4873	99.6036
13	0.0385	0.2750	99.8786
14	0.170	0.1214	10.0000

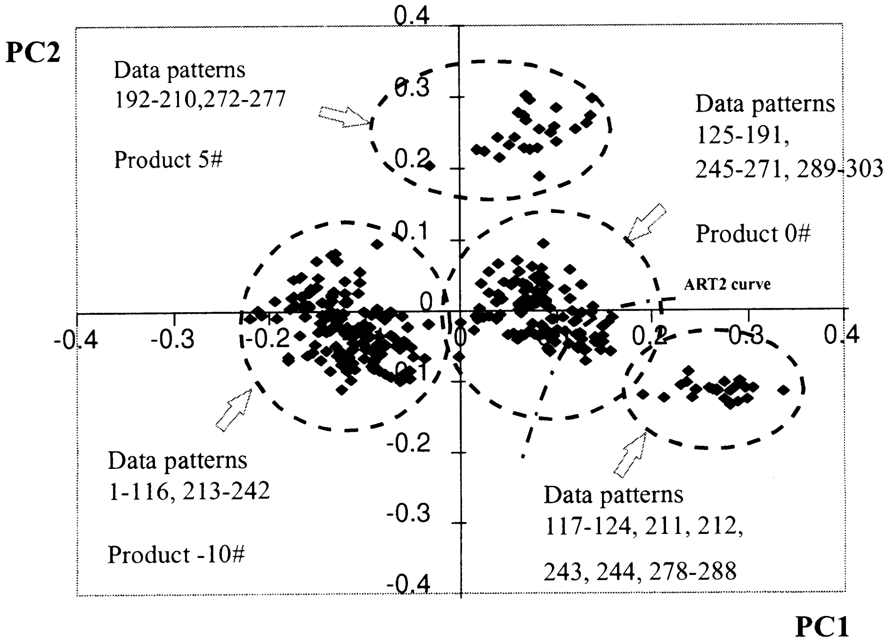


Figure 4.13 The PC1 and PC2 two dimensional plot.

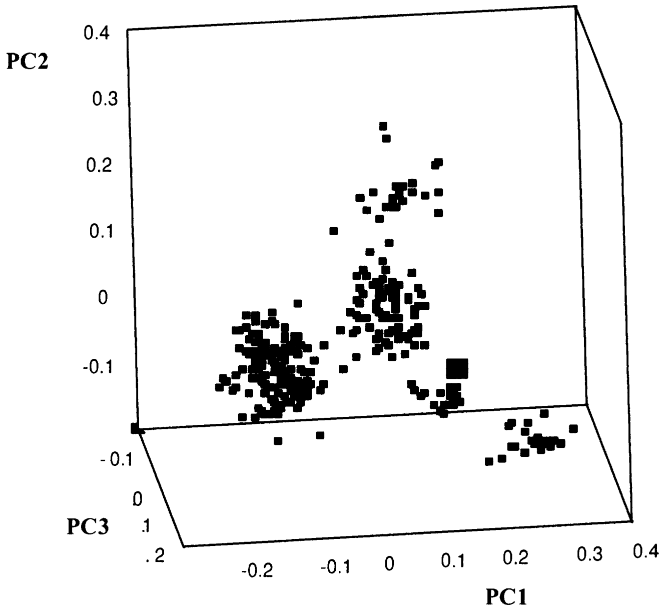


Figure 4.14 The PC1, PC2 and PC3 three dimensional plot.

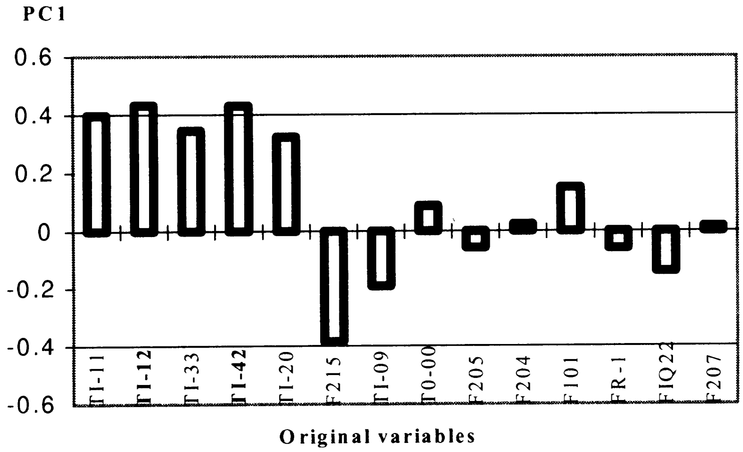


Figure 4.15 The contributing plot of PC1.

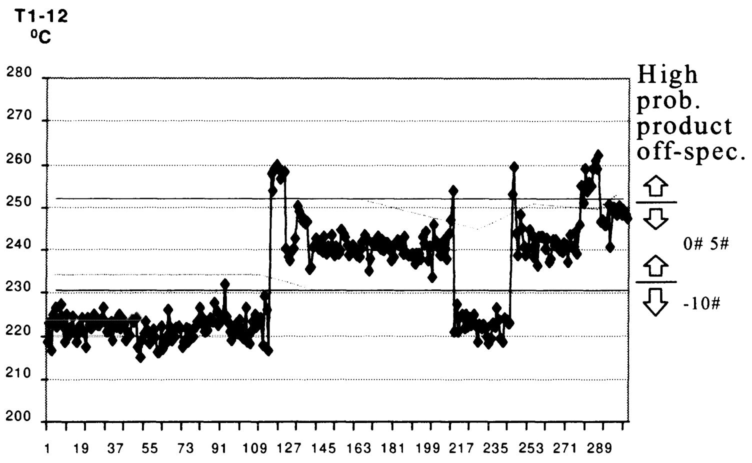


Figure 4.16 The changing profile of TI-12 over the 303 data patterns.

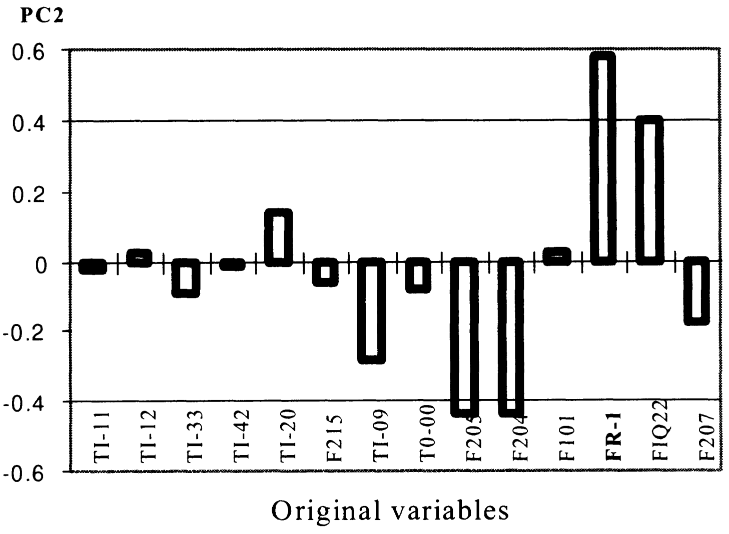


Figure 4.17 The contributing plot of PC2.

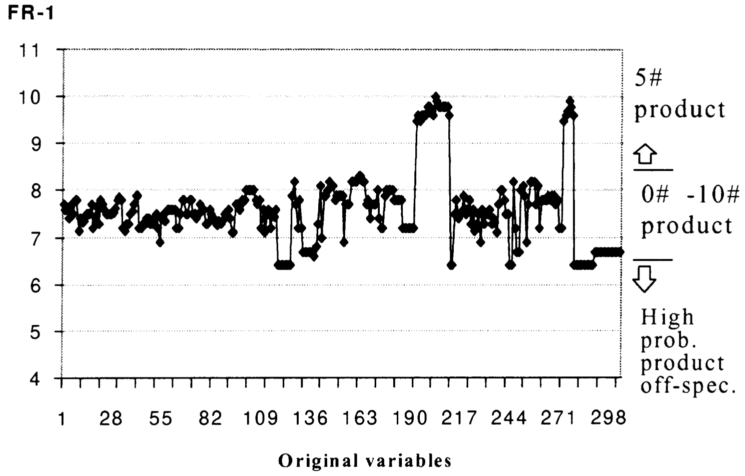


Figure 4.18 The changing profile of FR-1 over the 303 data patterns.

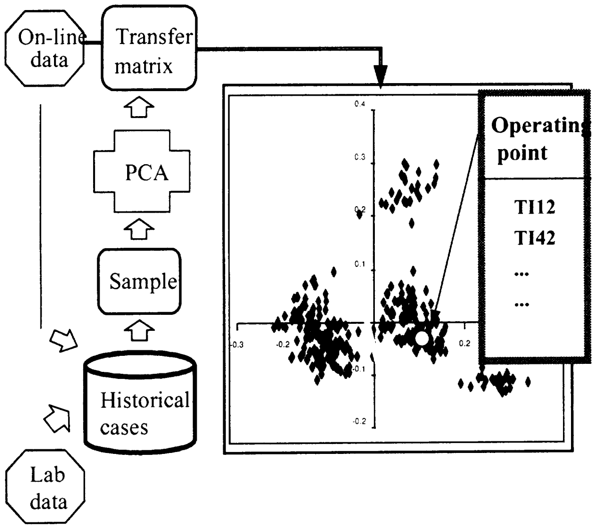


Figure 4.19 A PCA monitoring plane.

4.8 General Observations

PCA and PLS have proved to be powerful tools for operational data analysis and statistical process control. However they still have limitations. PCA and PLS based data analysis for statistical process control has the assumption that the first few PCs can capture most of the variations in a multivariate database. This assumption may be violated in some cases, e.g., when the dimension of the original variables is very large. Multiblock PCA and PLS can tackle this problem for some applications, however, dividing variables into blocks may not always be possible. Sammon [123] gave an example where data generated to contain five groups in four dimensions are projected into the space of two principal eigenvectors. Visual examination of this projection shows only four groups, since two of the clusters overlap completely in the two dimensional space. In such cases some alternative approaches may have to be used such as the unsupervised machine learning approaches to be introduced in Chapter 6, including neural network and Bayesian automatic classification methods. It has reported that one of the Bayesian automatic classification systems - AutoClass has successfully clustered data with 1204 attributes [39]. However, PCA and PLS may still be a useful approach for pre-processing the data to eliminate the linear dependencies in the data. PCA is also a useful approach for pre-processing the data for dimension reduction for neural networks [124].

The variable contributing plots such as Figure 4.9 may not be applicable in cases where the contributions of the original variables to the PCs are not equally distributed. Use of other approaches to compensate this limit of PCA can be a good alternative. For example, neural network models can be developed and used as sensitivity study tools to identify the contributions of variables.

In the above applications, PCA and PLS are used mainly for statistical process control for long term performance monitoring and the data dealt with are averaged over hours or days. PCA and PLS are also potentially useful for on-line real time data analysis. In Chapter 3 we have introduced the application of PCA for feature extraction and concept formation from dynamic trend signals. Bakshi [125] combined wavelet multiscale analysis with PCA for developing on-line monitoring systems. Tabe et al. [126] combined Fourier and wavelet analysis and PCA and developed an approach called dynamic PCA.

PCA can also be categorised as an unsupervised learning approach. However its learning is not recursive or incremental. For on-line real time use, it is useful for PCA to be able to learn incrementally, i.e., learn from a single example when it is presented. There has been an report on such on-line learning for principle component analysis [127].

Only a few case studies were mentioned above mainly for the propose of illustration the methods. There are many successful applications in using PCA and PLS approaches to analysing databases about continuous and batch operations. These include analysis of data of emulsion batch processes [128], product design [129], inferential process model development [130], reactor analysis [131], fault diagnosis [132], sensor fault identification [133, 134], normal operational region identification [136] and monitoring [137]. These applications not only explored the potential applications, but also provide valuable experience in overcoming some of the limitations of the PCA and PLS in solving practical problems.

CHAPTER 5
SUPERVISED LEARNING FOR OPERATIONAL SUPPORT

Studies on machine learning have mainly been concerned with automatic learning from examples to develop the knowledge describing these examples. This is clearly different from the kind of learning as learning to ride a bicycle. In supervised learning, each example used is typically described by a number of attributes. The attributes are divided into inputs and outputs, and the learning process is to develop a model mapping the multiple inputs and outputs. The model is gradually refined during learning to minimise the errors between the predictions and real values of outputs, i.e., so-called supervised learning. The most widely studied supervised learning approach is the feedforward neural network (FFNN). The FFNN model and its application to process operational support will be introduced in this Chapter. The discussion on FFNN will be focused on many of the practical issues that have to be considered in applying FFNN. While the focus will be on FFNN, other supervised models will also be described and compared with FFNN. These include fuzzy FFNN, fuzzy set covering approach and fuzzy signed digraph.

5.1 Feedforward Neural Networks

5.5.1 FFNN Architecture

There are already a large number of textbooks on FFNNs. Here it is introduced less technically. Simply speaking, a FFNN neural network is an algorithm or computer software that can learn to identify the complex nonlinear relationship between multiple inputs and outputs. The learning process has a number of characteristics.

Firstly, FFNN does not need fundamental domain problem models and is easy to be set up and trained. This is different from conventional statistical methods that usually require the user to specify the functions over which the data is to be regressed. In order to specify the function, the user has to know the forms of the equations governing the correlations between the data. If these functions are incorrectly specified, the data will not be satisfactorily regressed. Furthermore, considerable mathematics and numerical experience is required to obtain convergence if these equations are highly nonlinear. FFNN does not need to specify the forms of the correlations as well as any mathematical and numerical expertise requirements. Secondly, data examples used for training are allowed to be imprecise or noisy, in some cases even incomplete. Thirdly it mimics the human learning process: learning from examples through repeatedly updating the performance.

A FFNN neural network consists of a number of processing elements called neurons. These neurons are divided into layers. Figure 5.1 shows a three layered FFNN architecture including an input, a hidden and an output layer. Typically the input layer nodes correspond to input variables and the output layer to output variables. Hidden neurons do not have physical meanings. Neurons between two adjacent layers are fully connected by branches.

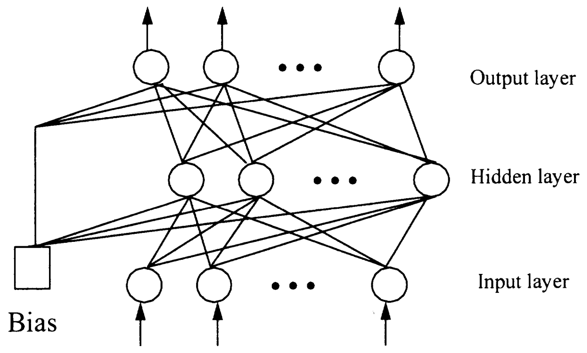


Figure 5.1 A three layer feedforward neural network.

Each neuron in the hidden and output layer is described by a transfer function (or activation function). Usually a sigmoidal function is used,

$$f(z) = \frac{1}{1+e^{-az}} \tag{5.1}$$

f(z) transforms an input z to the neuron to the range of [0.0, 1.0] as shown in Figure 5.2(a). The parameter a in Equation 5.1 is used to change the shape of the sigmoidal function. Some other activation functions can also be used as shown in Figure 5.2 (b) and (c). However for a specific FFNN structure, the neurons in the hidden and output layers are usually fixed on the same transfer function.

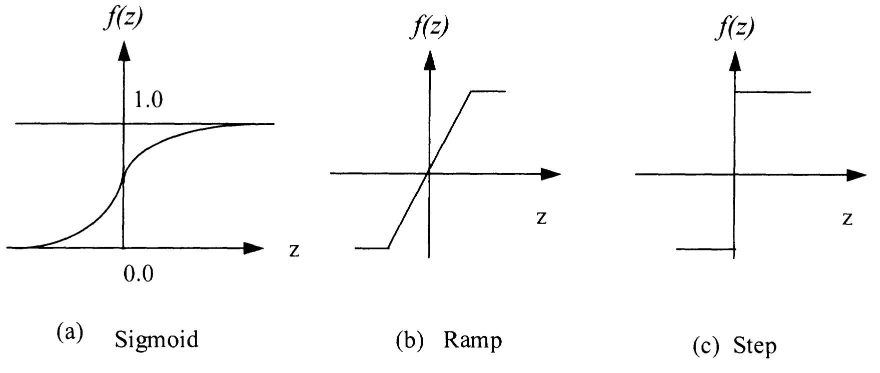


Figure 5.2 Activation functions.

Each connection branch is described by a weight representing the strength of connection between two linked nodes. The so called learning or training process is the procedure to adjust the weights. A bias neuron that supplies an invariant output is connected to each neuron in the hidden and output layers. The bias provides a threshold force activation of the neuron, and is essential in order to classify networks input patterns into various subspaces.

5.1.2 FFNN Training Algorithm

Given some arbitrary values for all the connection weights, for a specific data pattern, the FFNN makes use of the weights and input values to predict the outputs. The training is intended to gradually update the connection weights to minimise the mean square error E,

$$E = \sum_{m=1}^{M}\sum_{i=1}^{N}(t_i^{(m)} - y_i^{(m)})^2 \tag{5.2}$$

where M - number of training data patterns

N - number of neurons in the output layer

$t_i^{(m)}$ - the target value of the ith output neuron for the given mth data pattern

$y_i^{(m)}$ - the prediction for the ith output neuron given the mth data pattern

The process involves a forward path calculation to predict the outputs and backward path calculation to update the weights. For a neuron in the input layer, its output is equal to the input so there is in fact no activation function for an input neuron. For a neuron in the hidden and output layers, it receives the values of the outputs of its front layer nodes and takes the weighted sum as its input. The weighted sum is then transformed by the activation function to give an output. The outputs of the output layer neurons are compared with the target values using Equation 5.2 to calculate an error. The error is used to backwards updating the weights.

Given the *m*th data pattern, the weight updating in a supervised learning algorithm follows the formulation,

$$w_{ji}^{(m)} = w_{ji}^{(m-1)} + \Delta w_{ji}^{(m)} \tag{5.3}$$

where

$w_{ji}^{(m)}$ - the weight of the connection between the *j*th neuron of the upper layer and the *i*th neuron of the lower layer, in the *m*th learning iteration.

$w_{ji}^{(m-1)}$ - the weight of the connection between the *j*th neuron of the upper layer and the *i*th neuron of the lower layer, in the *(m-1)*th learning iteration.

$\Delta w_{ji}^{(m)}$ - the weight change.

In backpropagation learning approach, the weight change is calculated by,

$$\Delta w_{ji}^{(m)} = \eta \delta_j^{(m)} o_i^{(m)} + \alpha \Delta w_{ji}^{(m-1)} \tag{5.4}$$

where

η - learning rate, providing the step size during gradient descent. Generally to assure rapid convergence, larger step sizes which do not lead to oscillation are used

α - coefficient of momentum term, $0 < \alpha < 1$

$\theta_i^{(m)}$ - the output value of the *i*th neuron of the previous layer, in the *m*th iteration

$\delta_i^{(m)}$ - the error signal of the *j*th neuron in the *m*th learning iteration.

If *j* belongs to the output layer,

$$\delta_j^{(m)} = (t_j^{(m)} - y_j^{(m)}) f_j^{'}(\sum_i w_{ji}^{(m)} o_i^{(m)} + w_{jo}^{(m)}) \tag{5.5}$$

and if *j* belongs to the hidden layers,

$$\delta_j^{(m)} = f_j'(\sum_i w_{ji}^{(m)} o_i^{(m)} + w_{jo}^{(m)}) \sum_k \delta_k^{(m)} w_{kj}^{(m)} \tag{5.6}$$

where f' is the derivative of the transfer function.

Therefore, the error backpropagation approach for adjusting weights computes an error for each neuron in the output and hidden layers using Equations 5.5 and 5.6, and recursively updates the weights of all the layers using Equation 5.4, starting from the output layer and working backwards until the input layer.

Like all first order methods, backpropagation learning is a significantly less efficient optimisation method than are second order optimisation methods such as the conjugate gradient or quasi-Newton algorithms. Leonard and Kramer [138] studied the use of a conjugate gradient approach in order to speed up convergence. Alternative algorithms have also been proposed by Brent [139], Chen and Billings [140] and Peel et al. [141]. An attractive approach to the backpropagation learning algorithm is the quasi-Newton method [142,143].

Since its development, multilayer neural network has shown surprisingly good performance in solving many complex problems. However there is still a lack of theoretical explanation but an interesting theorem sheds some light on the capabilities of multi-layer percetrons [144]. This theorem states that any continuous function of N variables can be computed using linear summations and nonlinear but continuously increasing functions of only one variable. It effectively states that a three layer percetron with $N(2N+1)$ nodes using continuously increasing nonlearities can compute any continuous function of N variables.

5.1.3 Parameter Selection and Training Techniques

From the above discussion, it is clear that FFNN training involves the following initial decisions to begin: network topology, i.e., number of hidden layers and hidden neurons; learning rate; momentum factor; error tolerance or number of iteration; initial values of weights. Learning rate η and momentum factor α are not very difficult to set. We can start with some reasonable values, e.g., $\eta = 0.35$, $\alpha = 0.7$ and then find the most appropriate values in training. The error tolerance apparently depends on the problem to be solved. Understanding how these parameters affect the learning performance, which is due to discuss next, is useful in setting the right values. Initial values of weights can be generated by a random number generator. Therefore here our discussion will focus on network topology and then some other important issues in learning.

5.1.3.1 Network Topology

It is generally accepted that only one hidden layer is necessary in a network using a sigmoid activation function, and that no more than two are necessary if a step activation function is used.

There are no available methods to decide how many hidden neurons are required in a three layered network. The number of hidden neurons depends on the nonlinearity of the problem and error tolerance. Empirically the number of neurons in the hidden layer is of the same order as the number of neurons in the input and output layers. The number of hidden neurons must be large enough to form a decision region that is as complex as required by a given problem, too few hidden neurons hinder the learning process and may not be able to achieve the required accuracy. However, the number of hidden neurons must not be so large that many weights required can not be reliably estimated from available training data patterns. An unnecessary large hidden layer can lead to poor generality. A practical method is to start with a small number of neurons and gradually increase the number.

5.1.3.2 Local Minima

Chemical process models are multidimensional with peaks and valleys [145], which can trap the gradient descent process before it reaches the system minimum. There are several methods of combating the problem of local minima [146, 147]. The momentum factor α , which tends to keep the weight changes moving in the same direction, allowed the algorithm to slip over small minimal. Another approach is to start the learning again with different set of initial weights if it is found that the network keeps oscillating around a set of weights due to lack of improvement in the error. Some times adjusting the shape of the activation function (e.g., through adjusting the constant a in Equation 5.1 can have an effect on the network susceptibility to local minima. Some new optimising approaches have been applied to multilayer neural networks which prove to be able to address the local minima significantly, such as the simulated annealing [147]. In contrast to the comments made by Crowe and Vassiliadis [145] and Chitra [147], Knight [146] thought that FFNN rarely slips into local minima. However, it should be dealt with care.

5.1.3.3 Over fitting or over parameterisation

Overfitting occurs when the network learns the classification of specific training points but fail to capture the relative probability densities of the classes [148]. This can be caused by two situations: (1) oversized network, e.g., due to inclusion of irrelevant inputs in the network structure or too many hidden layers or neurons; and (2) insufficient number of training data patterns.

5.1.3.4 Generality

Over fitting in training a neural network deteriorates the generality of the network.

5.2 Variable Selection and Feature Extraction for FFNN Inputs

Inclusion of redundancy or irrelevant variables demands more training data and causes over fitting and deteriorates generalisation capability. If prior knowledge is available it should be used to remove irrelevant variables and identify correlated ones. Otherwise, mathematical techniques are required to solve the problem. In Chapter 9 by reference to the development of software sensors using FFNN, several approaches are described in detail. These include principal component analysis, sensitivity analysis and network weight matrix analysis. Here only a brief description is given to each method. For detailed discussion, please refer to Chapter 9.

Principal components are the linear combination of all the original variables. They are orthogonal and therefore linearly uncorrelated. The first few PCs can capture the variance. Therefore we can use the first few PCs rather than the original variables as the input variables to develop the FFNN model. Consequently the size of the network can be reduced, less training data is required and as a result can improve the generality. A smaller size is also favourable for training speed. A specific principal component does not represent any specific original input though relative contribution to individual PCs by the original variables can be analysed.

Sensitivity analysis can also be used to refine a network structure. The idea is to develop a FFNN model using all original inputs and use this model to carry out sensitivity studies to find out relative contributions of inputs to an output. Sensitivity studies can be used to develop a simpler network structure. Since the inputs are possibly correlated, sensitivity study should be carried out with care.

An alternative approach is to analyse the weights of a trained FFNN to find out the relative importance of inputs. Several approaches have been proposed which were discussed in Chapter 9. Due to the complex internal structure of FFNN, these approaches should be used with care.

The difficulty in analysing the relative importance of inputs to an output is that there is hidden layers and neurons and neurons between two layers are fully connected. An interesting idea is to develop a model without hidden layer and neurons, which is traditionally called single layer percetron (PCT). The order of magnitude of input - output linkage weights is clear. A PCT model may not be accurate enough but can provide some useful information for analysis. The analysis result can be used to further develop a FFNN model with hidden neurons. In fact, as will be demonstrated, for a problem whose nonlinearity is not high, a PCT can give equally good performance. In this case a PCT clearly should be used rather than a FFNN.

For on-line applications, feature extraction also means dimension reduction from dynamic transients to use minimum data to capture the useful information as well as noise removal. This has been discussed in detail in Chapter 3.

5.3 Model Validation and Confidence Bounds

The advantage of FFNN not requiring fundamental domain knowledge also brings a drawback of being a blackbox with poor extrapolation capability. Therefore, FFNN application always involves training and test procedures: using some data for training and the rest for testing. However, due to the multivariate nature, when a new data pattern is given, the confidence of prediction is still unknown, because the data pattern represents a point in the multidimensional space. The mean squared error to quantify the accuracy of training does not give this kind of information.

A detailed procedure is introduced in Chapter 9 to address this issue in the course of developing a software sensor, which uses an automatic clustering approach to group the multivariate data into clusters, and then training and test data is selected from each cluster. When new data patterns are available the clustering approach is also used to test if the data patterns are within the region of previously trained data, so provides clues of prediction accuracy and information whether the FFNN model needs to be retrained.

Techniques used in statistics for error or residual analysis can perfectly be used. A simple plotting of the error residual can provide useful information. In developing a software analyser for predicting a toxicity measure, Microtox, it was found that nine of the 180 data cases used for training and test have abnormally large errors [124]. It was suspicious that these nine data patterns may contain noise components. The plotting of error distribution supports this. As shown in Figure 5.3 that the error almost follows a normal distribution and there are irregular structure at the two ends which correspond to the nine data patterns.

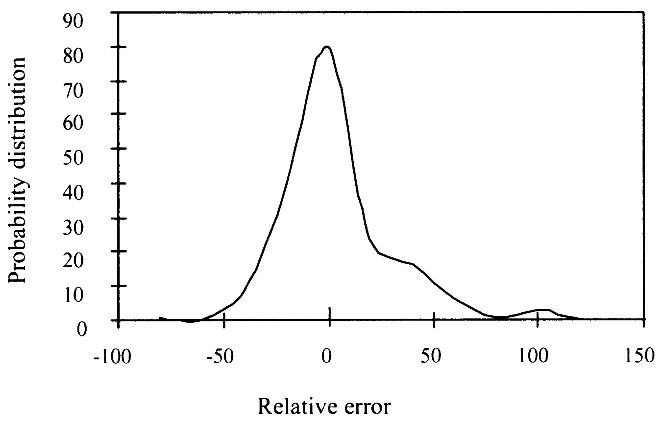


Figure 5.3 Error distribution on predicting a toxicity measure Microtox using FFNN.

Some researchers [149, 150] tried to develop an accurate procedure to calculate the confidence bound of FFNN in prediction. The confidence on predicting an output for a set of given input data depends on a number of factors,

(1) the distribution of residuals of the FFNN model, for training data patterns;

(2) the sensitivity of the output to inputs. For a problem of multiple inputs (x_1, x_2, ...x_N) and multiple outputs (y_1, y_2, ...y_M), all the sensitivities form a Jacob matrix,

$$\begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial y_M}{\partial x_1} & \frac{\partial y_M}{\partial x_2} & \cdots & \frac{\partial y_M}{\partial x_N} \end{bmatrix} \quad (5.7)$$

(3) the density distribution of the data sample in the multidimensional space of training data.

Shao et al. [149] and Zhang et al. [150] developed a complicated procedure for calculating the confidence bounds of using a FFNN mode to predict a new data pattern. Figure 5.4 shows such an example [180].

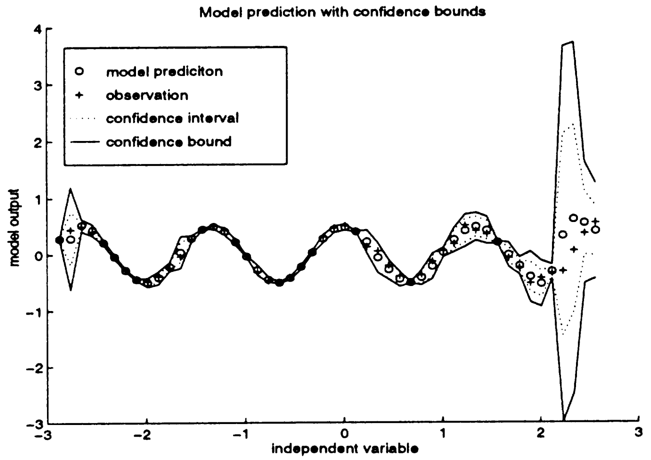


Figure 5.4 An example of confidence bounds.

5.4 Application of FFNN to Process Fault Diagnosis

A multilayered neural network can almost solve any multiple input / output problems. The number of ways in which FFNN can be used in process industries is limited only by the imagination. In this book our discussion only concerns its application in designing software sensors which will be dealt with in Chapter 9, and in process operational state identification and fault diagnosis, which is described in the following sections. The emphasis on operational state identification and fault diagnosis will be based on case studies, which are used to reveal the kind of challenges that need to be addressed in order to design effective and practical systems.

Fault identification involves the assimilation of available measurements to identify *malfunction* and *misbehaviour* of the process [151]. Function of a process equipment is the desired objective. For example the function of a distillation column is to achieve desired products within specifications, and the function of a heat exchanger is to increase the temperature of a cold stream or reduce the temperature of the hot stream to a certain value. The behaviour of a process unit is the status of the process under operation. Function and behaviour are treated separately because there is the potential that misbehaviour can be used to predict the malfunction. Both malfunction and misbehaviour can be described by three types of attributes: *process variables*, *model equations*, and *dynamic trends*.

Process variables here refer to those which change rapidly over a short time. For example, a flowrate may change very fast like a step or pulse function. They can be described simply by high, low, normal, medium high and low. Failure or not failure of some simple equipment can also be described in this way. For example, we can describe the failure of a pump as 1 and normal as 0.

Model equations. Model equations derived from the material and energy balances, equilibrium relations and rate equations impose constraints on process variables. These equations can be written in the form so that they are equal to zero. It is worth drawing attention to the fact that it is not necessary to list all model equations. Associated with each model equation are tolerance limits which are the expected positive and negative values of the residuals for which the constraint equation is to be satisfied. Violation of model equations may indicate a possible fault [152].

Dynamic trends. Graphically represented dynamic trends provide a direct way for process operators to make decisions on the operational status of a processes. However, in order to make efficient use of trends in a computer based system, pre-processing of the trends, so called feature extraction for the purpose of dimension reduction and noise removal is needed. The extracted features can then be used for further analysis by neural networks. Chapter 3 has introduced several approaches.

5.4.1 A Case Study - Fault Diagnosis of a CSTR Reactor

There has been a large number of publications on applying FFNN to process fault diagnosis, many have used continuous stirred tank reactors as case studies. Here an earlier case study on a CSTR reactor [153] is described in order to analyse the kinds of issues that need to be considered. In the CSTR reactor, there is a first-order exothermic reaction A → B. Heat generated by the reaction is taken away by cooling water. A three layered neural network was developed to use symptom variables to predict faults, as shown in Figure 5.5.

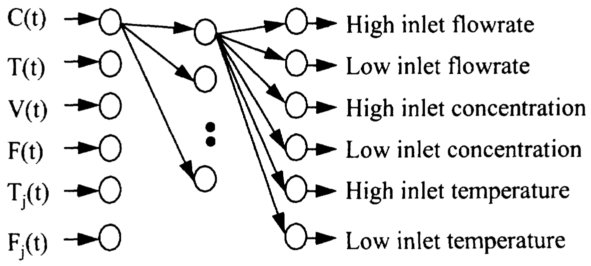


Figure 5.5 A three layer neural network for fault diagnosis of a CSTR reactor.

In Figure 5.5, C(t) is the outlet concentration of component A, T(t) outlet temperature, V(t), reactor hold up, F(t) outlet flowrate, $T_j(t)$ outlet temperature of coolant, $F_j(t)$ coolant flowrate. The output nodes of the network are possible faults or operations that can cause the symptom variables to change in certain ways. Dynamic simulation was used to produce the training data: making a change (5 to 15%) on one or two variables on the output side of Figure 5.5 and recording the values on all the symptom variables. Because the symptom variables experience dynamic transients, they were recorded in two ways, either the new steady state value or an average of four sampling points during the transients. The data was pre-processed before being used for training the network: each fault variable was given a boundary to define a variable being fault or not. If it is regarded as being a fault, it is assigned a value of 1, otherwise, it is assigned 0.

The detailed description of the case study can be found in the original publication [153]. In the following we just want to use this case study as an example to look at some general issues that need to be considered in applying FFNN to fault diagnosis.

5.4.2 Observations on the CSTR Reactor

Based on the above discussion on applying a FFNN to fault diagnosis of a CSTR, the following observations can be made:

(1) The study described in the above section considers open-loop situation because the dynamic simulation was made under open-loop conditions. It is known that under closed loop control variables will interact even stronger, the symptoms will change in different ways. Therefore the dynamic transient behaviour of a variable becomes very important. The time delays, peaks and valleys of a dynamic trend may contain important information. For example in Figure 5.6, the shapes of the three response curves for the same variable may tell important information about the distinction of three disturbances. It is not adequate to use only the new steady state value or an average of a few sampling points on the dynamic transient signal. However, it is obviously not suitable to use all the sampling points comprising a transient signals because the size is too large. Therefore, feature extraction from dynamic transient signals using appropriate technology for the purpose of dimension reduction is required. An input variable to the network of Figure 5.5 may need to be described by several features. Various methods for feature extraction already described in Chapter 3 can be chosen.

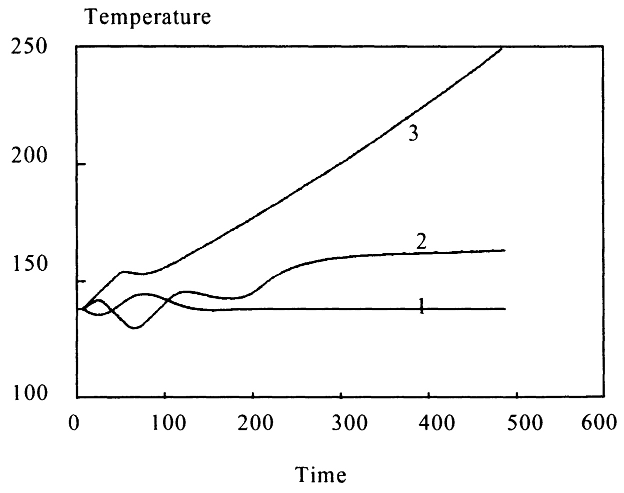


Figure 5.6 Dynamic trends of a temperature of a distillation column under closed loop control.

(2) Noise has to be dealt with carefully. Under large noise to signal ratio, the noise may bury the real trend of a dynamic transient. Methods for noise removal has also been discussed in Chapter 3.

(3) This CSTR example has six inputs and six outputs. For a larger process there may be hundred variables to be monitored. It may not be suitable to use a single neural network. Several neural networks or stacked networks may have to be used as described by Zhang et al.[154].

(4) Multiple faults also pose a challenge. This has been addressed by many researchers. However, some researchers have found that networks trained with single fault data can be used to detect multiple faults. It is rare to have more than two irrelevant faults occurring at the same time. But it is not unusual that one fault may cause other faults to occur.

(5) Most discussions on fault diagnosis using multilayerd neural networks are concerned with failure of equipment, sensors or sudden change of a variable. There is another type of fault or abnormal operation which is concerned with gradual degradation of product quality or other performance measures. Though there have been some discussions on using expert systems to deal with this kind of problems,, little work has been done on how neural networks can be used to deal with this type of problems.

(6) A necessary step before fault diagnosis is fault recognition or identification. It clearly depends on effective assimilation of all the measurements. A neural network may be a useful tool for diagnosis: mapping symptoms to faults, but it is not very effective in fault recognition. Some researchers used neural networks trained with normal operational data to identify faults: if the output was not in the normal region then a fault is expected.

(7) The biggest problem with FFNN is availability of training data. It is unthinkable that a real plant will initiate some faults to provide training data. Though dynamic training simulators have been used to generate training data [151, 155], methods for fault diagnosis should not rely on the assumption that a high flexible customised simulator of high fidelity is available.

The last two points make FFNNs which adopt a supervised learning mechanism less attractive in fault identification and diagnosis than the multivariate analysis approaches introduced in Chapter 4 and unsupervised machine learning methods to be introduced in Chapters 6 and 7.

5.5 Fuzzy Neural Networks

Descriptions commonly used by engineers in describing a variable being high or low, or a process being normal or abnormal are inherently fuzzy. These fuzzy descriptions are a kind of conceptualisation of numerical values, and are more qualitative and meaningful to operators. Fuzzy mathematics provides a technique for bridging the gap between qualitative descriptions and numerical values. Figure 5.7 shows how a process variable can be transformed to fuzzy concepts using fuzzy membership functions. It states that the variable takes three fuzzy values, high, normal and low. When it takes a fuzzy value, it is also attached with a fuzzy membership value. For example, (the variable = High, 0.8) is a statement that the value is high with a fuzzy membership value 0.8.

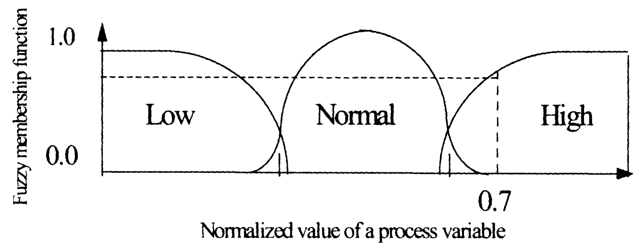


Figure 5.7 Fuzzification of a process variable.

Conventional neural networks have real number inputs and weights. There are three main types of fuzzy neural networks (FNNs) [156]: FNNs with fuzzy input signals but real number weights, FNNs with real number input signals but fuzzy weights, and FNNs with both fuzzy input signals and fuzzy weights. It is the first type of FNNs, i.e., fuzzy input signals and real number weights that has been studied for process fault diagnosis [151, 155, 154]. In the work of [154] fuzzy membership functions take part in the learning. In our work, fuzzy membership functions do not participate in the learning process but are only used for pre-processing the data. The procedure can be illustrated using Figure 5.8, which illustrates a fuzzy neural network for two input variables, x_1 and x_2 and one output y.

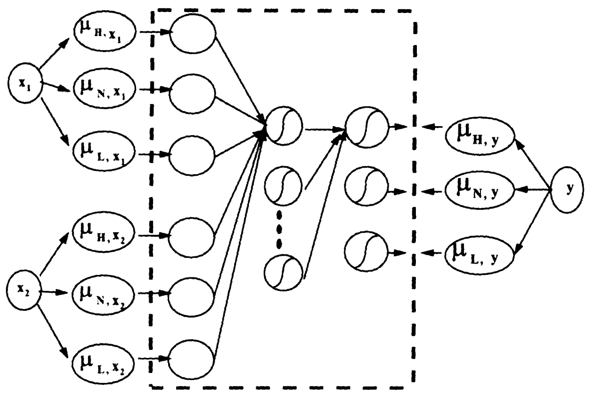
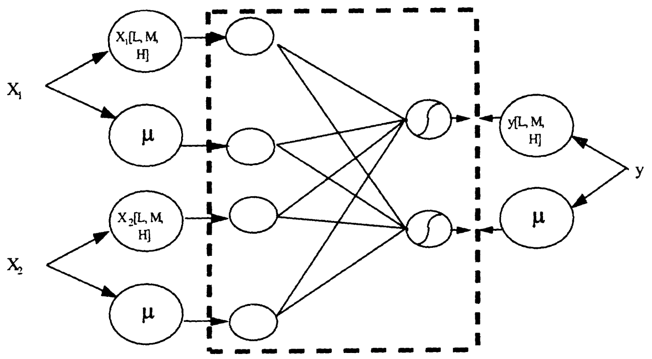


Figure 5.8 A fuzzy neural network architecture. L-low, H-high, N-normal.

Inside the dashed box of Figure 5.8 is a normal feed forward neural network. Outside of the dashed box represents fuzzy processing of the data before used for training. However, Such a FNN increases the size of the network dramatically. Each nput variable needs to be represented with three nodes. If each input variable is to e expressed by five fuzzy values, i.e., high, medium high, normal, medium low and ow, it will require five nodes. The FNN used in [154] has a similar problem. An lternative is to use the structure of Figure 5.9. In this structure each input variable always split into a pair of nodes, one is used to describe the fuzzy value being ther L, M or H, the other to represent the membership value. Case studies of plying FNNs for fault diagnosis will be given in Section 5.8



ıre **5.9** An alternative structure of fuzzy neural network.

5.6 Fuzzy Set Covering Method

If we regard all the symptoms of faults as a symptom fuzzy set $\tilde{M}$,

$$\tilde{M} = (m_1, m_2, ..., m_N) \tag{5.8}$$

and all faults form a fault fuzzy set $\tilde{D}$,

$$\tilde{D} = (d_1, d_2, ... d_T) \tag{5.9}$$

then fault diagnosis is to develop a mapping between the two fuzzy sets. Based on this idea, a fuzzy set covering (FSC) method was developed [151]. It is designated this name because it was initially inspired by the crisp set covering approach developed by Reggia et al.[157] which was also appreciated by Penalva et al. [158]. In [151] an approach was discussed to convert three types of symptoms, namely process variables, model equations and dynamic trends to fuzzy symptom concepts. With the progress in technology for pre-processing measurement signals as introduced in Chapter 3, it is possible to use better approaches. Therefore here we do not introduce the way to convert symptoms to fuzzy concepts introduced in [151], but focus the discussion on the fuzzy set mapping algorithm.

For a given diagnostic task, the extent to which symptom m_i exists is called the grade of membership of m_i in $\tilde{M}$, and is represented as a_i. All a_i form a fuzzy set $\tilde{A}$,

$$\tilde{A} = (a_1, a_2, ... a_N) \tag{5.10}$$

The degree to which $m_i(i=1, N)$ is caused by $d_j(j=1, T)$ represents the fuzzy relationship between fuzzy sets $\tilde{M}$ and $\tilde{D}$. This fuzzy relationship can be represented as a fuzzy matrix **R**,

$$\mathbf{R}=(r_{i,j})_{NxT},\ 0 \le r_{i,j} \le 1,\ i=1,N,\ j=1,T \tag{5.11}$$

The failure likelihood of fault d_j is represented by b_j and can be obtained from the following fuzzy set operation,

$$\tilde{A}\, \mathbf{O}\, \mathbf{R} = \mathbf{B} = (b_1, b_2, ..., b_T) \tag{5.12}$$

$$bj=(a_1 \overset{\bullet}{*} r_{1,j}) \overset{\wedge}{*} (a_2 \overset{\bullet}{*} r_{2,j})...(a_N \overset{\bullet}{*} r_{N,j}),\ j=1,T$$

where operator **O** is represented as $\mathbf{M}(\overset{\bullet}{*}, \overset{\wedge}{*})$ and can have different forms, as shown in Table 5.1. Wang et al. [151] only discussed the $M(\bullet, +)$ operator, comparisons of different operators were not carried out. A learning approach adapted from feedforward neural networks were used to learn the fuzzy matrix.

Table 5.1 Fuzzy matrix operators.

Model 1	$M(\wedge,\vee)$	$b_j=\bigvee_{l=1}^{N}(a_i \wedge r_{i,j})$	$j=1,2,...T$
Model 2	$M(\bullet,+)$	$b_j=\sum_{i=1}^{N} a_i r_{i,j}$	$j=1,2,...T$
Model 3	$M(\bullet,\vee)$	$b_j=\bigvee_{l=1}^{N} a_i r_{i,j}$	$j=1,2,...T$
Model 4	$M(\wedge,\oplus)$	$b_j=\oplus\sum_{i=1}^{N} a_i \wedge r_{i,j}$ $a \oplus b=\min\{1,a+b\}$	$j=1,2,...T;$
Model 5	$M(\bullet,\oplus)$	$b_j=\oplus\sum_{i=1}^{N} a_i r_{i,j}$	$j=1,2,...T$

7 Fuzzy Signed Digraphs

recent years a notable development in fault diagnosis has been the signed directed ıph (SDG). Since it was first proposed for fault diagnosis by Iri et al. [159] it has racted much attention [160 - 171, 194]. It is attractive because it provides an gant and straight forward tool for qualitatively analyse the cause-effect ıtionships between variables. Neural networks are clearly not capable in this ect. However, there are common limitations in all the SDG models. Firstly, the ressive capability is very limited since they are crisp graphs - a node or a branch only take three values, i.e., -, 0 and +. As a result it will give ambiguous tions in complicated fault diagnosis. The application of fuzzy concepts by Han . [172] and Shih and Lee [173] only makes the input nodes to be able to ert numerical data to qualitative expression but the graph as a whole is still a one. Secondly the reasoning methods in a SDG model are often dependent on over simplified assumptions. As a result large errors can be expected when ning in a complex structure. Thirdly, the development of the SDG has been ation driven. For example, the algorithms have been developed specifically ault diagnosis and can't be applied directly to qualitative simulation.

Furthermore, the SDG models are not able to deal with uncertainty in data and reasoning simultaneously. Finally, they are not able to learn from data.

A fuzzy-SDG method [59, 60, 174] was developed which has far more features to overcome many of the limitations of SDG models, which was later further improved by Huang and Wang (1999).

Fuzzy graph is a natural generalisation of the crisp graph using fuzzy set concepts. A crisp graph is defined by the pair **G** = (**X**, **E**) where **X** is a finite set of nodes and E a non fuzzy relation on **X** x **X**. A fuzzy graph [179] is a pair ($\tilde{X}$, $\tilde{E}$), where $\tilde{X}$ is a fuzzy set on **X** and $\tilde{E}$ is a fuzzy relation on **X** x **X** such that $\mu_{\tilde{E}} \le \min(\mu_{\tilde{X}}(x), \mu_{\tilde{X}}(x'))$. Here $\mu_{\tilde{E}}$ is the membership function of the binary effect of two adjacent nodes x and x' over a branch, $\mu_{\tilde{X}}$, the membership function of the node. However, in some situations it may be desirable to relax this inequality [178]. Algorithms about fuzzy graphs can be found in [178]. Obviously, if $\mu_{\tilde{E}}$ and $\mu_{\tilde{X}}$ only take the values -1, 0 or 1, then a fuzzy graph becomes crisp. A fuzzy-SDG is defined by nodes defining variables, branches representing the effects between two variables and reasoning propagation algorithms associated with the graph.

5.7.1 Nodes

Each node in the fuzzy-SDG is represented by a variable which can take a number of values from the fuzzy value space. An example of value space of a node is shown in Figure 5.7, in which the process variable takes three fuzzy values, high, normal and low. Another example is shown in Figure 5.10, in which L is the liquid level in a tank changing from 0 to 2 meters, *v* is the normalised value of L in the range of -1.0 to 1.0, and the fuzzy membership value changes from 0 to 1. Each fuzzy value in Figure 5.10, such as medium low or low, is a fuzzy set $\tilde{M}$ defined by Equation 5.13.

$$\tilde{M} = \{x, \mu\}, \quad \mu = [0,1] \tag{5.13}$$

$\tilde{M}$ is therefore represented by its membership function, μ, such that the value of μ illustrates the degree of membership of the element x belonging to $\tilde{M}$. The membership function can have many shapes, such as triangular and trapezoidal. The fuzzy value medium low in Figure 5.10 is a half-declined trapezoidal form which can be represented by Equation 5.14.

$$m = \begin{cases} 1 & 0.6 \le v < a_1 \\ \dfrac{a_2 - v}{a_2 - a_1} & a_1 \le v \le a_2 \\ 0 & v > a \text{ or } v < 0.6 \end{cases} \tag{5.14}$$

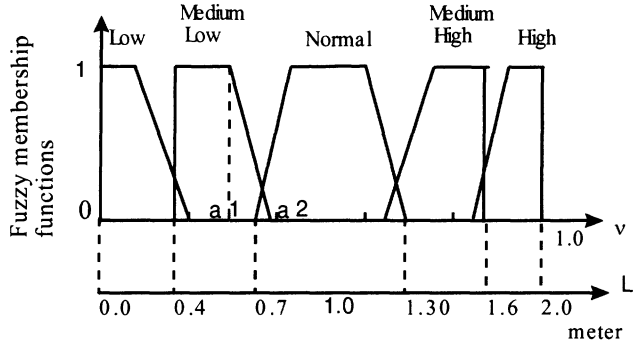


Figure 5.10 The fuzzy value space of a variable.

5.7.2 Branches

Attached to each branch connecting two nodes is an arrow representing the effect direction and an effect strength. The effect strength is measured by a weight. Suppose that x_{j+1} and x_j are two nodes linked by a branch directed from x_{j+1} to x_j, then the effect strength of x_{j+1} on x_j is determined by Equation 5.15,

$$e(x_{j+1} \rightarrow x_j) = S_{j,j+1} \frac{R_{x_{j+1}}}{R_{x_j}} \tag{5.15}$$

in which R_{xj+1} and R_{xj} are the value range (i.e., maximum -minimum) of nodes x_{j+1} and x_j respectively and $S_{j,\,j+1}$ is the sensitivity of x_j to x_{j+1}, determined by Equation 5.16.

$$S_{j,\,j+1} = \frac{\partial x_j}{\partial x_{j+1}} \tag{5.16}$$

The value range for a node consists of positive and negative ranges, corresponding to fuzzy values, v in the range [0, 1] and [-1, 0] respectively. Obviously, the larger the value of $e(x_{j+1} \rightarrow x_j)$, the stronger the effect of x_{j+1} on x_j. If the relationship uses a time derivative to account for the dynamics, this can be approximated using a backward difference. The sensitivity will be derived from the partial derivative estimated using the partial with respect to the rate of change of the quantity as shown by Equation 5.17.

$$s_{j,j+1} = \frac{\partial (dx_j / dt)}{\partial x_{j+1}} \tag{5.17}$$

There are three basic connections in a fuzzy-SDG, i.e., serial (Figure 5.11(a)), divergent (Figure 5.11(b)) and convergent (Figure 5.11(c)) connections. Combination of the three can form any complicated networks such as Figure 5.11(d). Wang et al. [59] described the reasoning strategies for the basic connections.

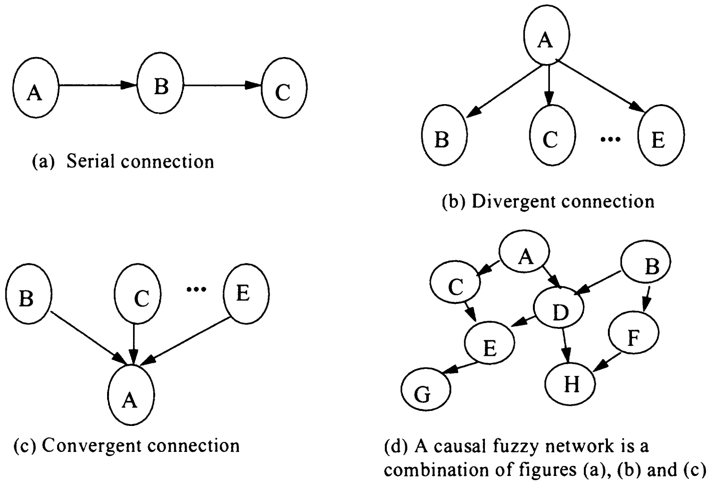

A
B
C
(a) Serial connection
A
B
C
E
(b) Divergent connection
B
C
E
A
(c) Convergent connection
A
B
C
D
E
F
G
H
(d) A causal fuzzy network is a combination of figures (a), (b) and (c)

Figure 5.11 Basic connections in a fuzzy - SDG.

Fuzzy -SDG proved to be able to overcome many of the limitations of SDG mentioned earlier. More importantly, fuzzy-SDG has the learning capability from data, therefore can be used as a tool for data mining and knowledge discovery.

5.7.3 The extended fussy-SDG

In many situations, using a single weight (defined by Equation 5.15) is not sufficient because it implies a linear relationship between two nodes. The extended fuzzy-SDG developed by Huang and Wang [175] developed a more sophisticated method describing the connection relationship.

Two adjacent layers of a fuzzy-SDG is shown in Figure 5.12(a), which depicts the cause - effect relationships between variables [X_1, X_2, X_3] and [Z_1, Z_2, Z_3], indicating that Z_1 is dependent on X_1 and X_3 but independent on X_2. Figure 5.12(b) shows the detail for training this substructure. The substructure involving X_1, X_3 and

Z_1 can be trained independently. The node X_1 is converted into two types of nodes: X_1[L, M, H] and μ. The first type of nodes takes only discrete values such as H (high), M (medium) and L (low). The second type of nodes takes continuous values μ between 0 and 1 representing the fuzzy membership values when the first node takes the value of H, M or L. The outside of the dashed box of Figure 5.12(b) represents fuzzy processing of the original data. The arrangement is different from that of the fuzzy neural network previously studied [151], that requires three nodes to represent a variable if the variable takes three fuzzy values. However, the present method always uses two nodes, therefore, the size of the network does not increase with increased values in the fuzzy space. Inside the box of Figure 5.12(b) is a single layer percetron with no hidden layers. But it also allows to have one hidden layer.

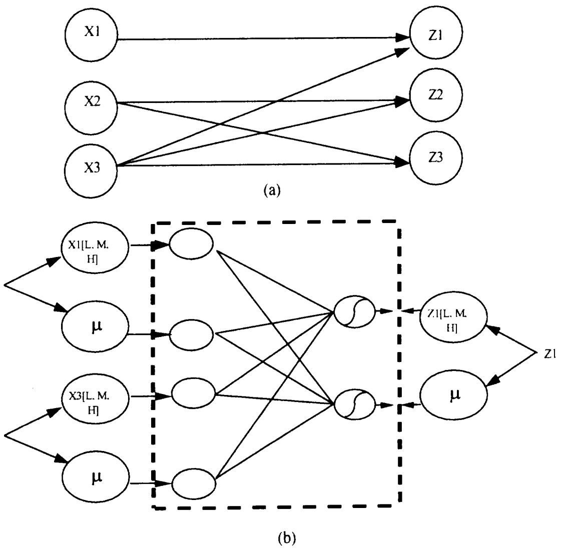


Figure 5.12 Learning of a convergent fuzzy-SDG.

Figure 5.13 compares a fuzzy - SDG with a fuzzy neural network. Suppose all the variables in Figure 5.13, X1 to X3, Z1 to Z11 and Y1 to Y3 represent variables in a process. The non-linearity between the input variables [X1, X2, X3] and the output variables [Y1, Y2, Y3] are expected to be high due to their distances. A neural network (or a fuzzy neural network) with only one hidden layer as shown in Figure 5.13(b) can usually be able to deal with the high non-linearity between [X1, X2, X3] and [Y1, Y2, Y3].

If we have the knowledge of the cause – effect relationships between [X1, X2, X3] and [Y1, Y2, Y3] via a number of intermediate variables, e.g., Z1-Z11, we can make use of the knowledge to develop a cause-effect diagram like Figure 5.13(a). The procedure for training the convergent connection in a fuzzy-SDG has been discussed above (refer to Figure 5.12) using the first layer of Figure 5.13(a). A single layer percetron (Figure 5.12(b)) can normally give good performance for a substructure in a fuzzy-SDG. It is because the non-linearity between any directly connected layers is normally not high. If we view the fuzzy-SDG network in the horizontal direction, it is a different way of linear summation of a number of small non-linear (e.g., sigmoid) functions.

There is no doubt that the relationship between two connected nodes becomes more complex compared with the original fuzzy-SDG because the weight is replaced by a complicated relationship. However, the relative magnitudes of weights are not significant, because they depend on the determination of the maximum and minimum boundary values of variables in normalisation. In fact during reasoning we are mainly concerned with the values of individual nodes and the propagation of reasoning in the whole network, not the weight of a branch. Similar observations can be found in the Bayesian networks in which the branches only mean a link between two nodes. The reasoning in a Bayesian network is based on the conditional probability calculation, which requires a complex conditional probability table.

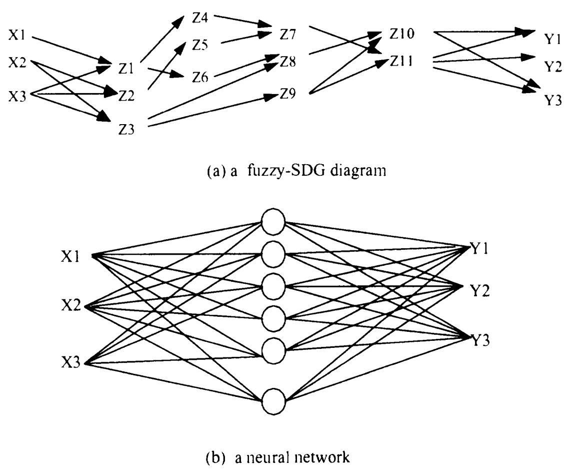


Figure 5.13 Comparison of a fuzzy-SDG and a neural network.

5.8 Case Studies

In this section we describe two case studies. The first case study compares fuzzy neural network, fuzzy single layer percetron and fuzzy set covering approaches to fault diagnosis of a FCC process. It will demonstrate that three layered fuzzy neural networks normally give more accurate results than single layer percetron and fuzzy set covering approaches. However all three can identify significant disturbances or faults and the difference is at identifying small changes. In terms of qualitative interpretation of connection weights, FSC and PCT are advantageous compared with FFNN. In the second case study, it shows that for a problem whose nonlinearity is not high, a fuzzy - SDG gives equally good result as FFNN and has much simpler structure and clearer causal - effect picture.

5.8.1 Application to Fault Diagnosis of a FCC Process

The refinery residual fluid catalytic cracking process (R-FCC) described in the appendix B is used as a case study. The data used for this study is summarised in Table B2 of Appendix B. To avoid confusion in what follows, we use data patterns to refer to the 67 data patterns, and fault types to refer to the 13 types of faults.

5.8.1.1 Fuzzyfication of Input - Output Variables

For a process variable that changes rapidly over a short time, violations of prescribed high and low limits are indications of possible faults. They are described by two variable pairs: *variable_name (high, x)* and *variable_name (low, x)*, where *x* is the membership value representing the degree of high and low measures.

Model equations impose constraints on the process variables which are derived from the material and energy balances, equilibrium and rate equations. The equations are written so that they are equal to zero when satisfied. Associated with each model equation are tolerance limits which are the expected positive and negative values of the residual for which the constraint equation is to be satisfied. Here the residual is defined as the deviation of the constraint equations from zero. They are also described by the two variable pair: *constraint _name(positive, x)* and *constraint _name(negative, x)*. In this case study, no model equations are considered.

Chapter 3 has introduced a number of approaches for pre-processing dynamic trends, such as wavelets, principal component analysis and episode approaches. Here we use a fuzzy approach. The fuzzy approach may not be superior over methods introduced in Chapter 3 but can be used directly by fuzzy neural networks. The approach is an adaptation of the method proposed by Chen and Jong [176], with the introduction of fuzzy concepts and can be explained by reference to Figure 5.14. The dynamic response is divided into three stages (I, II and III in Figure 5.14). Each stage corresponds to a particular feature of the system behaviour. Stage I is associated with the order of the system, II the maximum rate of change of the system when maximum control input is applied and III relates to the settling stage and is an indication of the stability of the system.

For stage I, a variable pair: *trend_section_1(inc, x)* and *trend_section_1(dec, x)* is used. The treatment of section II has similar measures: *trend_section_2(inc, x)* and *trend_section_2(dec, x).* However, five values are used to interpret stage III, namely:

trend_section_3(divergence, x)
trend_section_3(oscillation, x)
trend_section_3(stable, x)
trend_section_3(mean_value_high, x)
trend_section_3(mean_value_low, x).

In the first three pairs x has a value 0 or 1. While in the last two measures which represent the deviation of the mean value of this stage from that for normal operation x has values from 0 to 1. This means that a dynamic trend can be represented by nine inputs in a network for fault diagnosis, as shown in Figure 5.15 (the part in the box of dashed line) of a diagnostic network.

Figure 5.15 is a fuzzy neural network for diagnosing the FCC process. The output layer represents faults. The first output node, fresh feed F (*high*, x), refers to fault of high fresh feed flowrate. The x is in the range [0, 1] measuring the degree of high. While the x in node 12, compressor (*failure, x)* only takes 0 or 1, with 1 meaning failure and 0 normal.

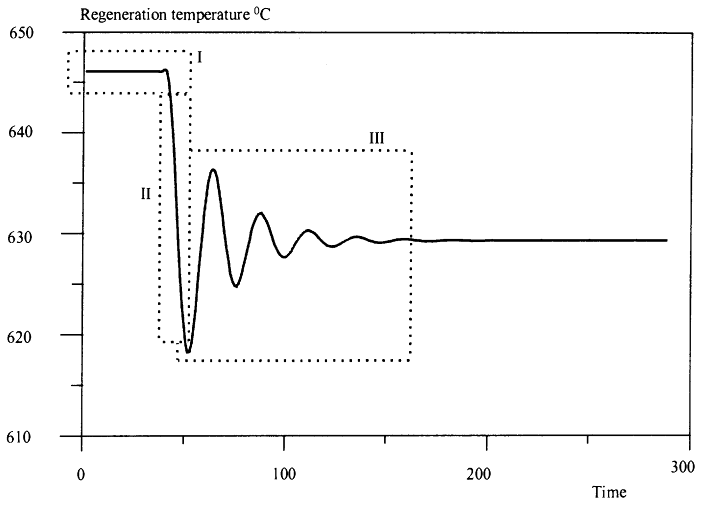


Figure 5.14 Fuzzy qualitative interpretation of a dynamic trend.

The input layer nodes in Figure 5.15 refer to symptoms. Some are described by dynamic trends represented by nine nodes, and some as process variables and are described by two nodes, high and low. By independently changing the parameters of the outputs and recording the responses of inputs, data corresponding to Table B2 was obtained.

Figure 5.15 shows the fuzzy neural network structure which has one hidden layer. A fuzzy single layer percetron which has the same input - output structure but no hidden layers, and a fuzzy set covering model is also used to the same problem in order to make comparisons. It might be difficult to have a completely fair comparison, considering the difference in structure and training details. The comparison is based on the assumption that all three, are in their optimum structure and training parameters.

5.8.1.2 Discussion of Results

The broad nature of results is summarised in Table 5.2, which shows that the Fuzzy FFNN model is able to identify 65 of the 67 faults in the samples, compared with 55 and 53 out of 67 for fuzzy PCT and FSC respectively. The result might be anticipated since FFNN is the best technique for dealing with non-linear data followed by PCT and FSC. However there are more interesting observations.

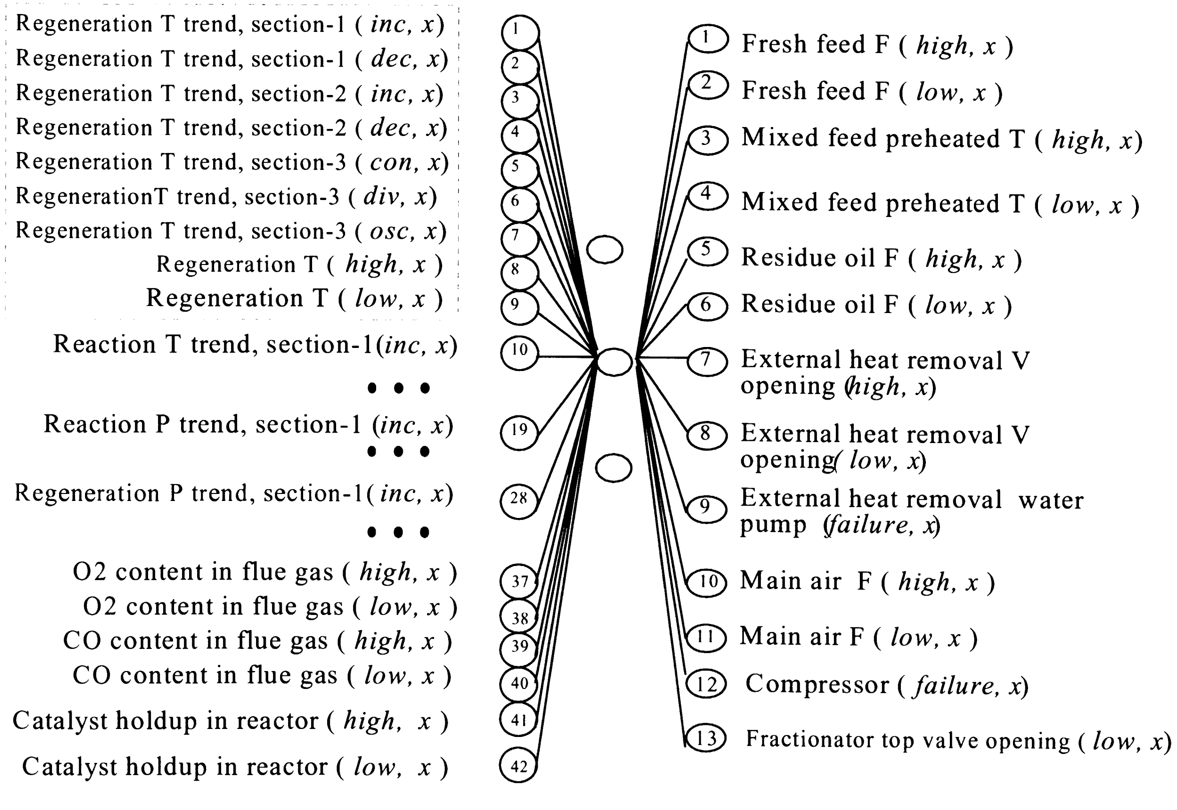

Regeneration T trend, section-1 (inc, x)
Regeneration T trend, section-1 (dec, x)
Regeneration T trend, section-2 (inc, x)
Regeneration T trend, section-2 (dec, x)
Regeneration T trend, section-3 (con, x)
RegenerationT trend, section-3 (div, x)
Regeneration T trend, section-3 (osc, x)
Regeneration T (high, x)
Regeneration T (low, x)
Reaction T trend, section-1(inc, x)
• • •
Reaction P trend, section-1 (inc, x)
• • •
Regeneration P trend, section-1(inc, x)
• • •
O2 content in flue gas (high, x)
O2 content in flue gas (low, x)
CO content in flue gas (high, x)
CO content in flue gas (low, x)
Catalyst holdup in reactor (high, x)
Catalyst holdup in reactor (low, x)
1
2
3
4
5
6
7
8
9
10
19
28
37
38
39
40
41
42
1 Fresh feed F (high, x)
2 Fresh feed F (low, x)
3 Mixed feed preheated T (high, x)
4 Mixed feed preheated T (low, x)
5 Residue oil F (high, x)
6 Residue oil F (low, x)
7 External heat removal V opening (high, x)
8 External heat removal V opening(low, x)
9 External heat removal water pump (failure, x)
10 Main air F (high, x)
11 Main air F (low, x)
12 Compressor (failure, x)
13 Fractionator top valve opening (low, x)

Figure 5.15 The fuzzy neural network structure for fault diagnosis of FCC.

Close inspection of the fault data patterns in Table 5.2 shows that the faults which are not identified by any of the procedures are for smaller disturbances. For instance, data pattern 24 represents an increase of 10% in the preheated temperature of the mixed feed. This is a relatively small change compared with other data patterns, and the system controllers are able to bring the system back to designed operating condition fairly quickly. In fact, all the significant disturbances and faults can be recognised by three models. It is apparent that faults not identified by FFNN remain unidentified by PCT and FSC as well. In the same way, those not identified by PCT are missed by FSC. So the order for describing effectiveness in accounting for small disturbances is FFNN, PCT and FSC.

Table 5.2 Comparison of three approaches in fault diagnosis.

	Number of patterns identified	Data patterns that are not identified
Fuzzy FFNN	65 out of 67	24, 51
PCT	55 out of 67	24, 51, 10, 11, 23, 24, 27, 28, 29, 39, 44, 45
FSC	51 out of 67	24, 51, 10, 11, 23, 24, 27, 28, 29, 39, 44, 45, 33, 34

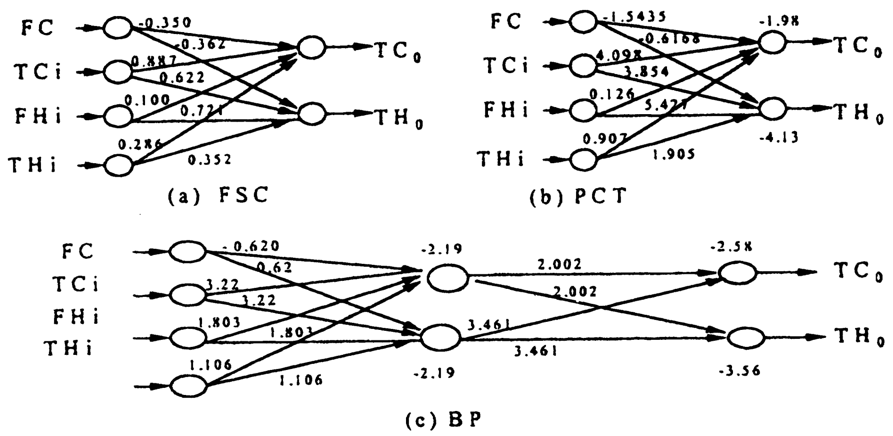


Figure 5.16 The effect weights of three approaches for a heat exchanger and the cause-effect explanation.

Table 5.3 Comparison of three methods in predicting faults[a,b,c].

	1	2	3	4	5	6	7	8	9	10	11	12	13
No 7	**0.7778**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
BP	**0.7258**	0.0093	0.0004	0.0006	0.0000	0.0174	0.0120	0.0000	0.0000	0.0000	0.0155	0.0000	0.0003
PCT	**0.5567**	0.0409	0.0651	0.0359	0.0322	0.0474	0.0338	0.0196	0.0004	0.0198	0.0069	0.0013	0.0117
FSC	**0.5613**	0.0141	-0.0111	-0.0097	-0.0279	0.0083	0.0146	0.0162	-0.0352	-0.0373	0.0157	-0.0307	0.0197
No 32	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	**1.0000**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
BP	0.0006	0.0024	0.0000	0.0001	0.0001	0.0000	**0.8491**	0.0001	0.0000	0.0000	0.0035	0.0000	0.0000
PCT	0.1224	0.0476	0.0200	0.0295	0.0305	0.0293	**0.7479**	0.0085	0.0002	0.0264	0.0061	0.0009	0.0067
FSC	0.2161	0.0393	-0.0762	-0.0184	0.0309	0.0079	**0.6593**	-0.0189	-0.0003	0.0171	0.0049	-0.0028	0.0132
No 38	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	**1.0000**	0.0000	0.0000	0.0000	0.0000
BP	0.0000	0.0022	0.0000	0.0000	0.0000	0.0000	0.0000	0.0327	**0.9871**	0.0008	0.0000	0.0000	0.0106
PCT	0.0322	0.0229	0.0199	0.0167	0.0134	0.0161	0.0017	0.3529	**0.9018**	0.0648	0.0030	0.0675	0.0156
FSC	-0.0429	0.0317	-0.0662	0.0316	0.0007	-0.0205	0.0020	0.2796	**0.7681**	0.0282	-0.0864	-0.1079	0.0460
No 43	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	**1.0000**	0.0000	0.0000	0.0000
BP	0.0000	0.0128	0.0005	0.0000	0.0000	0.0001	0.0000	0.0000	0.0002	**0.9557**	0.0006	0.0004	0.0035
PCT	0.0022	0.1458	0.0270	0.0064	0.0024	0.0171	0.0112	0.0030	0.0078	**0.8170**	0.0041	0.0075	0.0885
FSC	-0.195	0.0313	0.0427	-0.0124	-0.0432	-0.0211	0.0401	-0.1144	0.0695	**0.7781**	0.0196	0.0316	0.0190
No 60	**0.7222**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	**0.8454**	0.0000	0.0000	0.0000	0.0000	0.0000
BP	**0.7347**	0.0000	0.0000	0.0002	0.0019	0.0000	0.0013	**0.8203**	0.0019	0.0000	0.0000	0.0007	0.0000
PCT	**0.6892**	0.0050	0.0305	0.0536	0.1205	0.0292	0.0208	**0.7579**	0.0669	0.0013	0.0058	0.0246	0.0027
FSC	**0.5345**	0.0153	-0.0145	-0.0559	0.0241	-0.1010	0.1218	**0.7848**	0.1320	-0.2026	0.0058	0.1114	-0.0328
No 61	**0.7778**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	**1.0000**	0.0000	0.0000	0.0000	0.0000
BP	**0.7763**	0.0006	0.0000	0.0000	0.0000	0.0000	0.0000	0.0075	**0.9908**	0.0000	0.0000	0.0033	0.0000
PCT	**0.6969**	0.0039	0.0082	0.0263	0.0456	0.0181	0.0025	0.1887	**0.9362**	0.0050	0.0042	0.0586	0.0045
FSC	**0.7340**	-0.0615	-0.0813	0.0339	0.1096	-0.377	-0.0388	-0.0442	**1.1698**	0.0665	0.0399	0.0896	-0.0729
No 65	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	**1.0000**	0.0000	0.0000	**1.0000**	0.0000
BP	0.0000	0.0003	0.0000	0.0000	0.0000	0.0000	0.0000	0.0087	**0.9812**	0.0015	0.0000	**0.9784**	0.0037
PCT	0.0022	0.0104	0.0049	0.0097	0.0119	0.0098	0.0010	0.1603	**0.8441**	0.0433	0.0018	**0.8598**	0.0679
FSC	-.0223	0.0473	-0.0160	-0.0128	-0.0527	0.0374	-0.0055	0.2691	**0.6280**	0.0932	-0.0255	**0.7562**	0.0926
No 67	0.0000	**0.8333**	**0.7500**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
BP	0.0001	**0.8200**	**0.7447**	0.0000	0.0000	0.0012	0.0000	0.0000	0.0002	0.0020	0.0021	0.0000	0.0117
PCT	0.0300	**0.7592**	**0.7144**	0.0040	0.0013	0.0077	0.0146	0.0015	0.0100	0.0630	0.0251	0.0003	0.0787
FSC	0.0390	**0.7977**	**0.7038**	0.0143	0.0211	-0.0044	-0.0135	0.0201	-0.0176	0.1017	-0.030	-0.0273	0.0188

[a] - **1** to **13** are fault types corresponding the 13 output nodes of Figure 5.15

[b] - No 7 is the 7th data pattern

[c] - 1.0 means fault; 0 means no fault.

Despite the three having significantly different capabilities when monitoring small disturbances, they are almost equal in isolating major disturbances and faults. Table 5.3 gives the comparison of three methods for a number of fault situations. All three can identify the faults, even in double fault situations, though in most cases FFNN predictions are slightly closer to 1.0 than the other two (1 means fault). However, PCT and FSC are much faster in convergence. A typical example is that FFNN requires several hours, PCT one hour and FSC 30 minutes on a personal computer.

People are also interested in what FFNN, PCT and FSC can help us to develop a better understanding of the cause-effect behaviour of the process. It is easier to illustrate this in a simple counter current flow heat exchanger shown in Figure 5.16(a). Our interest is to study how the cold stream flowrate (FC) and its inlet temperature (TC_i) and the hot stream flowrate (FH) and its inlet temperature (TH_i) affect the outlet temperature TC_0 and TH_0. The weights obtained for the same set of data for FFNN, PCT and FSC are shown in Figure 5.16(b). For both PCT and FSC, it is very clear to see from the weights how an input affects an output. However it is not so clear from the weights of a FFNN due to the existence of a hidden layer.

5.8.2 Application to a Waste Water Treatment Plant

The extended fuzzy-SDG was applied to a waste water treatment plant and compared with feedforward neural networks by Huang and Wang [175]. The plant has three treatment units in series. Through sensitivity study, a fuzzy-SDG is developed as shown in Figure 5.17. The variables are illustrated in Table 5.4. The advantage of this fuzzy-SDG network is that compared with neural networks, it is more intuitive to engineers and supervisors. Neural networks are fully connected and can give predictions given the inputs. However, it is not straightforward to qualitatively know the weighted contribution of inputs to the outputs. In contrast, the causal network is no longer a blackbox because it is a partially connected graph. Engineers can trace forward and backward the network to analyse problems. For example, the output suspended solids, SS-S is observed as (High, 0.10), tracing back the causal graph and the nodes, SS-D (High, 0.68), SS-P(High, 0.28), RD-SSP(Medium, 1.00), SS-E (High, 0.88) and SSV-E(M, 0.87). So the main reason causing SS-S(High, 0.10) is due to SS-E(High, 0.88).

Table 5.4 The attributes of the database[a].

1	Q-E	(input flow to plant)	20	SSV-D	(input VSSto secondary settler)
2	ZN-E	(input Zinc to plant)	21	SED-D	(input sediments to secondary settler)
3	PH-E	(input pH to plant)	22	COND-D	(input conductivity to secondary settler)
4	BOD-E	(input BOD to plant)	23	PH-S	(output pH)
5	COD-E	(input COD to plant)	24	BOD-S	(output BOD)
6	SS-E	(input SS to plant)	25	COD-S	(output COD)
7	SSV-E	(input VSS to plant)	26	SS-S	(output suspended solids)
8	SED-E	(input sediments to plant)	27	SSV-S	(output VSS)
9	COND-E	(input conductivity to plant)	28	SED-S	(output sediments)
10	PH-P	(input pH to primary settler)	29	COND-S	(output conductivity)
11	BOD-P	(input BOD to primary settler)	30	RD-BOD-P	(performance input BOD in primary settler)
12	SS-P	(input SS to primary settler)	31	RD-SS-P	(performance input SS to primary settler)
13	SSV-P	(input VSS to primary settler)	32	RD-SED-P	(performance input sediments to primary settler)
14	SED-P	(input sediments to primary settler)	33	RD-BOD-S	(performance input BOD to 2nd settler)
15	COND-P	(input conductivity to primary settler)	34	RD-COD-S	(performance input COD to 2nd settler)
16	PH-D	(input pH to secondary settler)	35	RD-BOD-G	(global performance input BOD)
17	BOD-D	(input BOD to secondary settler)	36	RD-COD-G	(global performance input COD)
18	COD-D	(input COD to secondary settler)	37	RD-SS-G	(global performance input SS)
19	SS-D	(input SS to secondary settler)	38	RD-SED-G	(global performance input sediments)

[a] BOD - biological oxygen demand, COD - chemical oxygen demand, SS - suspended solids, VSS - volatile suspended solids

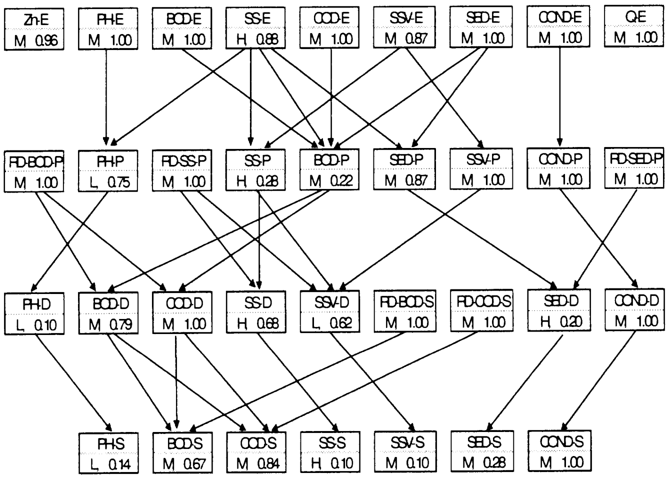


Figure 5.17 The fuzzy-SDG for the wastewater treatment plant.
M - medium; L - low; H - high.

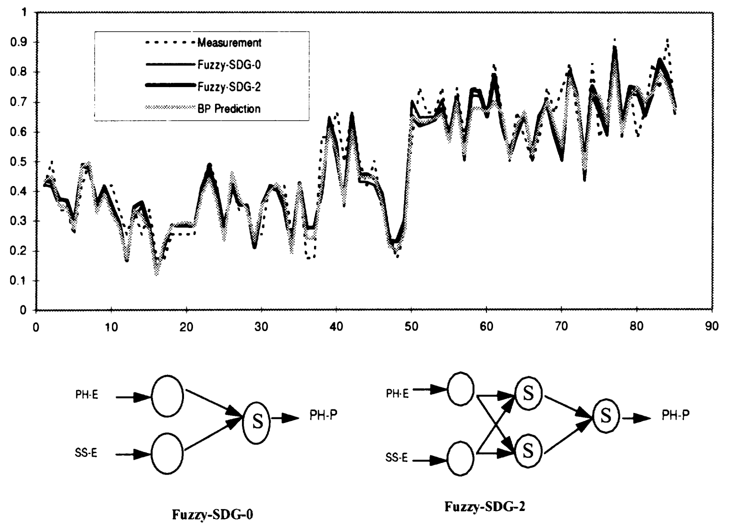


Figure 5.18 Comparisons of fuzzy-SDG without hidden neurons (Fuzzy-SDG-0), fuzzy-SDG with two hidden neurons (Fuzzy-SDG-2) and backpropagation neural networks (BP).

In terms of accuracy of the fuzzy - SDG, a comparison is made as shown in Figure 5.18. It compares the prediction of PH-P (the second node on the second layer from top of the network shown in Figure 5.17) using a fuzzy-SDG with no hidden neurons (Fuzzy-SDG-0), with two hidden neurons (Fuzzy-SDG-2) and a fully connected neural network with ten hidden neurons (BP Prediction). The fully connected neural network has to include all the inputs in the first layer of Figure 5.17. As far as prediction accuracy is concerned, for all data patterns used for training, three models give equivalent predictions as demonstrated in Figure 5.18. The average relative errors for the whole data cases using Fuzzy-SDG-0, Fuzzy-SDG-2 and BP are 11.6%, 12.3% and 11.6% respectively.

5.9 General Observations

Several supervised machine learning approaches are introduced using illustrative case studies. These include neural and fuzzy neural networks, fuzzy set covering and fuzzy-SDG. The focus of discussion has been on issues that need to be addressed in practical applications. These include variable selection and feature extraction for FFNN inputs, model validation and confidence bounds, accuracy and generality, feature extraction from dynamic transient signals as well as the ability to give causal knowledge. There are some other issues which have not or have not fully addressed in the context.

The backpropagation learning is not a recursive approach, which means that when new data is to be used to update the knowledge, it has to be mixed with previous data and the model to be retrained again. Recursive models are particularly useful for on-line systems because they can continuously update their knowledge.

When applied to fault diagnosis, supervised learning approaches described above are suitable for mapping the set of symptoms to the set of so-called essential faults [151]. Faults can be divided into two categories: essential faults and root faults. For example, the symptom - "the trend of liquid level increases dramatically at the bottom of the column" can be explained by the following essential faults: (1) high feedflowrate; (2) low bottom withdrawal flowrate; (3) low flash zone temperature; (4) low flash zone pressure; (5) change in feed properties. These essential faults are indications of root faults. Looked at in this way, low bottom withdrawal flow is an essential fault which can be the result of the root faults of a pump failure or fail to open a valve. Therefore fault diagnosis is a two step procedure, mapping from the set of symptoms to the set of essential faults, and explanation of the essential faults. The unsupervised approaches introduced are suitable for the first step. For the second step, it is more appropriate to use expert systems or graphical models such as signed digraph. Similar strategies have been developed by Becraft and Lee [177] and Wang et al. [151].

Since supervised learning approaches require training data which is difficult to get for the purpose of fault identification and diagnosis, they are less attractive compared with multivariate analysis approaches (Chapter 3), and unsupervised approaches (Chapter 6). However, the good performance in correlating non-linear inputs-outputs makes them very attractive in developing software sensors or software analysers (Chapter 9).

The extrapolation problem of feedforward neural networks has been addressed through input dimension reduction, proper selection of training and test data, model validation and confidence bound evaluation as well as using single layer percetron, FSC or fuzzy-SDG if feasible. There are also efforts in incorporating partial domain knowledge into neural network learning.

CHAPTER 6
UNSUPERVISED LEARNING FOR OPERATIONAL STATE IDENTIFICATION

6.1 Supervised vs. Unsupervised Learning

This chapter describes some representative unsupervised machine learning approaches for process operational state identification. Whether a machine learning approach is regarded as supervised or unsupervised depends on the way it makes use of prior knowledge of the data. Data encountered can be broadly divided into the following four categories:

(1) Part of the database is known, i.e., the number and descriptions of classes as well as the assignments of individual data patterns are known. The task is to assign unknown data patterns to the established classes.

(2) Both the number and descriptions of classes are known, but the assignment of individual data patterns is not known. The task is then to assign all data patterns to the known classes.

(3) The number of classes are known but the descriptions and the assignment of individual data patterns are not known. The problem is to develop a description for each class and assign all data patterns to them.

(4) Both the number and descriptions of classes are not known and it is necessary to determine the number and descriptions of classes as well as the assignments of the data patterns.

The methods introduced in Chapter 5 are powerful tools for dealing with the first type of data where the objective is to assign new data patterns to previously established classes. Clearly this is not appropriate for the last three types of data, since training data is not available. In these cases unsupervised learning approaches are needed and the goal is to group data into clusters in a way such that intraclass similarity is high and interclass similarity is low. In other words, supervised approaches can learn from known to predict unknown while unsupervised approaches learn from unknown in order to predict unknown. Supervised learning can generally give more accurate predictions, but can not be extrapolated: when new data is not in the range of training data, predictions will not generally be reliable. For process operational state identification and diagnosis, supervised learning needs both symptoms and faults. Therefore the routine data collected by computer control systems can not be used directly for training. Faults are unlikely to be deliberately introduced to an industrial process in order to generate training data.

Grouping of data patterns using unsupervised learning is often based on a similarity or distance measure, which is then compared with a threshold value. The degree of autonomy will depend on whether the threshold value is given by the users or determined automatically by the system. In this chapter three approaches are studied, the adaptive resonance theory (ART2), a modified version of it named ARTnet, and Bayesian automatic classification (AutoClass). ART2 and ARTnet though require a pre-defined threshold value are able to deal with both the third and fourth types of data. AutoClass is a completely automatic clustering approach without the need to pre-define a threshold value and the number and descriptions of classes, so is able to deal with the fourth type of data.

6.2 Adaptive Resonance Theory

The adaptive resonance theory (ART) was developed by Grossberg [47, 48], as a clustering-based, autonomous learning model. ART has been instantiated in a series of separate neural network models referred to as ART1, ART2 and ART3 [43, 49, 50, 51]. ART1 is designed for clustering binary vectors and ART2 for continuous-valued vectors.

A general architecture of the ART neural network is shown in Figure 6.1. The network consists of three groups of neurons, the input processing layer (F_1), the cluster layer (F_2), and a mechanism which determines the degree of similarity of patterns placed in the same cluster (a reset mechanism). The input layer can be considered as consisting of two parts: the input and an interface. Some processing may occur in the both (especially in ART2). The interface combines signals from the input to the weight vector for the cluster unit which has been selected as a candidate for learning. The input and interface are designated as F_1 *(a)* and F_1 *(b)* in Figure 6.1.

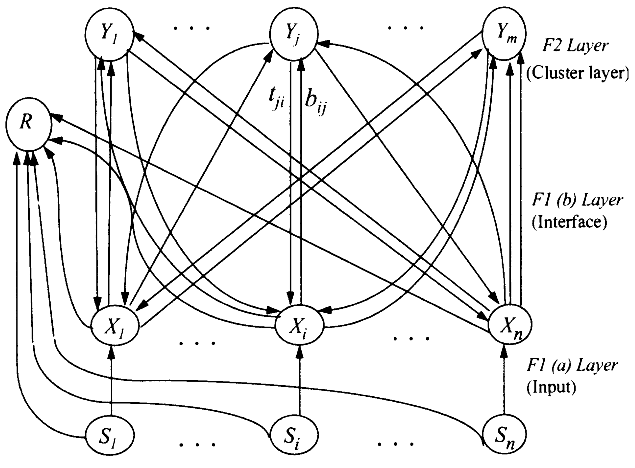


Figure 6.1 The general architecture of ART.

An input data pattern is presented to the network, following some specialised processing by the network, and is compared to each of the existing cluster prototypes in the cluster layer (F_2). The "winner" in the cluster layer is the prototype most similar to the input. Whether or not a cluster unit is allowed to learn an input data pattern depends on how similar its top-down weight vector is to the input vector. The decision is made by the reset unit, based on signals it receives from the input (a) and the interface (b) of the F_1 layer. If the similarity between the "winner" and the input exceeds a predetermined value (the vigilance parameter ρ, in ART terms), learning is enabled and the "winner" is modified slightly to more closely reflect the input data pattern. If the similarity between the "winner" and the input is less than required by the vigilance parameters, a new cluster unit is initialised in the top layer and learning is enabled to create a prototype similar to the input. Thus both the number and descriptions of classes are continuously updated during learning. This is different from the Kohonen network, where the descriptions of classes (called neurons in a Kohonen network) are also continuously update during learning, but the number of classes needs to be determined before learning starts.

To control the similarity of patterns placed in the same cluster, there are two sets of connections (each with its own weights) between each unit in the interface part of the input layer and the cluster units. The F_1 *(b)* layer is connected to the F_2 layer by bottom-up weights; the bottom-up weight on the connection from the ith F_1 unit to the jth F_2 unit is designated b_{ij}. The F_2 layer is connected to the F_1 *(b)* layer by top-down weights; the top-down weight on the connection from the jth F_2 unit to the ith F1 unit is designated t_{ji}.

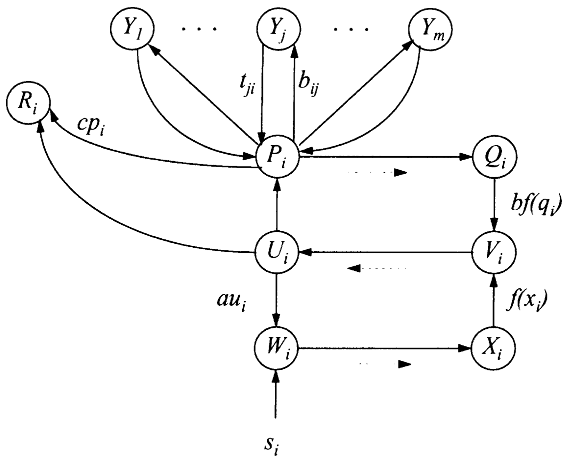


Figure 6.2 Typical ART2 architecture.

ART2 is designed to process continuous-valued data patterns. A typical ART2 architecture [43] is illustrated in Figure 6.2. It has a very sophisticated input data processing layer (F_1) consisting of six types of units (the *W, X, U, V, P and Q* units). There are *n* units of each of these types (where *n* is the dimension of an input pattern). Only one unit of each type is shown in Figure 6.2. A supplemental unit between the *W* and *X* units receives signals from all of the *W* units, computes the norm of the vector **w**, and sends this (inhibitory) signal to each of the *X* units. Each of these also receives an excitation signal from the corresponding *W* unit. Details of this part of the net are shown in Figure 6.3. A similar supplemental unit performs the same role between the *P* and *Q* units, and another does so between the *V* and *U* units. Each *X* unit is connected to the corresponding *V* unit, and each *Q* unit is connected to the corresponding *V* unit also.

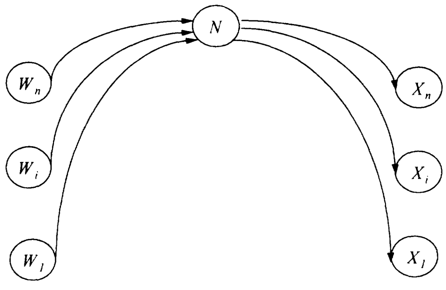


Figure 6.3 Details of connections from W to X units, showing the supplemental unit N to perform normalisation.

The symbols on the connection paths between the various units in the F_1 layer in Figure 6.2 indicate the transformation that occurs to the signals as it passes from one type of unit to the next; they do not indicate multiplication by the given quantity. However, the connections between units P_i (of the F_i layer) and Y_i (of the F_2 layer) do show the weights that multiply the signal transmitted over those paths. The activation of the winning F_2 unit is d, where $0<d<1$. The symbol ---> indicates normalisation; i.e., the vector **q** of activations of Q units is just the vector **p** of activations of the P units, normalised to be approximately unit length.

The U units perform the role of an input phase of the F_1 layer. The P units play the role of the interface to the F_1 layer. Units X_i and Q_i apply an activation function to their net input; this function suppresses any components of the vectors of activations at those levels that fall below the user-selected value θ. The connection paths from W to U and from Q to V have fixed weights a and b, respectively.

The most complete description of the ART2 network can be found in [43]. Although written for ART1, the discussions by Caudil [181] and Wasserman [182] provide a useful but less technical description of many of the principles employed in ART2.

ART2 has shown great potential for analysis of process operational data and identification of operational state due to a number of properties [183, 184, 195]. First, it is considered as an unsupervised machine learning system that does not require training data. It is well known that it is difficult to find training data for the purpose of process fault identification and diagnosis. Second, the approach is recursive, or in the terms used in ART2, it is *plastic*, that is, it is able to acquire new knowledge and retain stable in the sense that existing knowledge is not corrupted.

This property is apparently very useful for on-line monitoring where information is received continuously.

6.3 A Framework for Integrating Wavelet Feature Extraction and ART2

Wang et al. [185] and Chen et al. [82] developed an integrated framework named ARTnet which combines wavelet for feature extraction from dynamic transient signals and adaptive resonance theory. In ARTnet the data pre-processing part uses wavelets for preprocessing the data for feature extraction. In order to introduce ARTnet it is helpful to first examine the mechanism of ART2 for noise removal. ART2 has a data pre-processing unit which is very complicated but the mechanism for removing noise uses a simple activation function A(x),

$$A(x) = \begin{cases} x & x > \theta \\ 0 & x < \theta \end{cases} \tag{6.1}$$

where θ is a threshold value. If an input signal is less than θ, it is considered to be a noise component and set to zero. This has proved to be inappropriate for removing noise components contained in process dynamic transient signals which are often of high frequencies and in certain magnitude.

6.3.1 The Integrated Framework ARTnet

A mechanism was proposed by Pao [186] to replace the data pre-processing part of ART2 with more efficient noise removal and dimension reduction methods. This has been followed in this study by using wavelet to replace the data pre-processing unit of ART2. The integral framework is called ARTnet to distinguish it from ART2. The conceptual architecture of ARTnet is shown in Figure 6.4.

In this new architecture, wavelets are used to pre-process the dynamic trend signals. The extracted features are used as inputs to the kernel of ARTnet for clustering. A pattern feature vector $(x_1, x_2, \cdots, x_N)$ is fed to the input layer of the ARTnet kernel and weighted by b_{ij}, bottom-up weights. In Chapter 3, the extrema of wavelet multiscale analysis should be regarded as the features of dynamic transient signals.

The weighted input vector is then compared with existing clusters in the top layer by calculating the distance between the input and existing clusters. The existing cluster prototype, which has a smaller distance than the input is called the winner. By considering this input the description or knowledge of the wining cluster is updated. Whether or not a winning cluster prototype is allowed to learn from an input data pattern depends on how similar the input is to the cluster. If the similarity measure exceeds a predetermined value, called the vigilance parameter, learning is enabled. If the similarity measure is less than the required vigilance parameter, a new cluster unit is then created which reflects the input. Clearly this is unsupervised and recursive learning process.

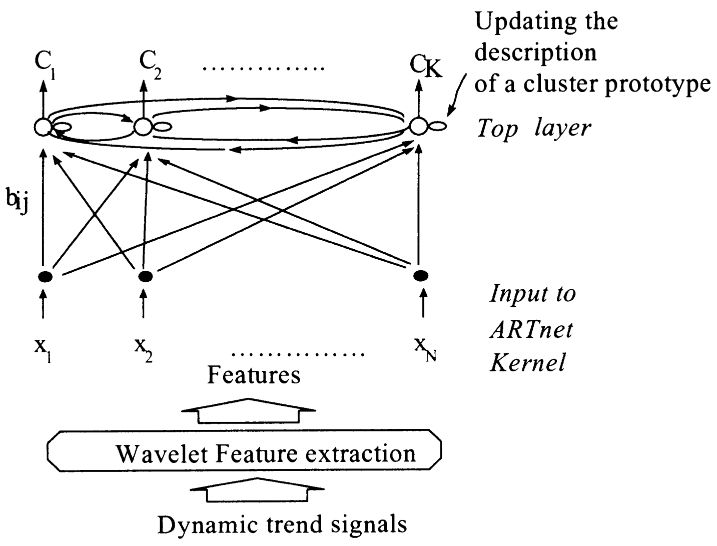


Figure 6.4 The conceptual architecture of ARTnet.

6.3.2 The Similarity Measure in ARTnet

It is apparent that the learning process is concerned with the extent to which how similar two vectors are. There are several ways to measure the distance between two pairs of observations, such as the Hamming or Euclidean distance. For continuous data, the Euclidean distance is the most commonly used [187]. Formally, the Euclidean distance between two vectors **x** and **y** is defined as the root sum-squared error,

$$\|x - y\|_2 = \left\{\sum_{n=1}^{N}(x_n - y_n)^2\right\}^{1/2} \tag{6.2}$$

Suppose there are K existing cluster prototypes. The kth cluster prototype consists of a number of data patterns and is also described by a vector, denoted as $z^{(k)}$, which has considered all data patterns belonging to it. Clearly, if there is only one data pattern in the cluster, $z^{(k)}$, it is equal to that data pattern. When a new input data pattern **x** is received, a distance between **x** and $z^{(k)}$ is calculated according to the expression,

$$\sigma^2(k) = \sum_{(x \in Cluster\, k)} \left(\|x - z^{(k)}\|_2\right)^2 \tag{6.3}$$

Since the distance between **x** and all existing cluster prototypes is calculated, the cluster prototype with the smallest distance is the winner. If the distance measure for the winner is smaller than a pre-set distance threshold, ρ, then the input **x** is assigned to the winning cluster and the description of the cluster is then updated,

$$z_i^{(k)} = z_i^{(k)} + \frac{1}{NF} x_i b_{ij} \qquad i = 1...N,\ j = 1...K \tag{6.4}$$

where $z_i^{(k)}$ refers to the ith attribute of the vector **z** for the cluster k. b_{ij} is the weight between the ith attribute of the input and the jth existing cluster prototype. NF is the number of features.

6.4 Application of ARTnet to the FCC Process

The FCC process and the data used is described in appendix B. To demonstrate the procedure, 64 data patterns are used. Discussion is limited to 64 data patterns in order to keep the discussion manageable and to assist in presentation of the results. The data sets include the following faults or disturbances:

- fresh flow rate is increased or decreased
- preheat temperature for the mixed feed increases or decreases
- recycle slurry flow rate increases or decreases
- opening of the hand valve V20 increases or decreases
- air flow rate increases or decreases
- the opening of the fully open valve 401-ST decreases
- cooling water pump fails

- compressor fails
- double faults occur

The sixty four data patterns were obtained from a customised dynamic training simulator, to which random noise was added using a zero-mean noise generator (MATLAB®). In the following discussions, the term "data patterns" refers to these sixty four data patterns and "identified patterns" to the patterns estimated by ARTnet.

Figure 6.5(a) shows a reactor temperature transient when the fresh feed flowrate increases by 70%. Figure 6.5(b) is the same transient with random noise. The corresponding four scales from multiresolution analysis for this transient are shown in Figure 6.6, together with the corresponding extrema on the right hand side of Figure 6.6.

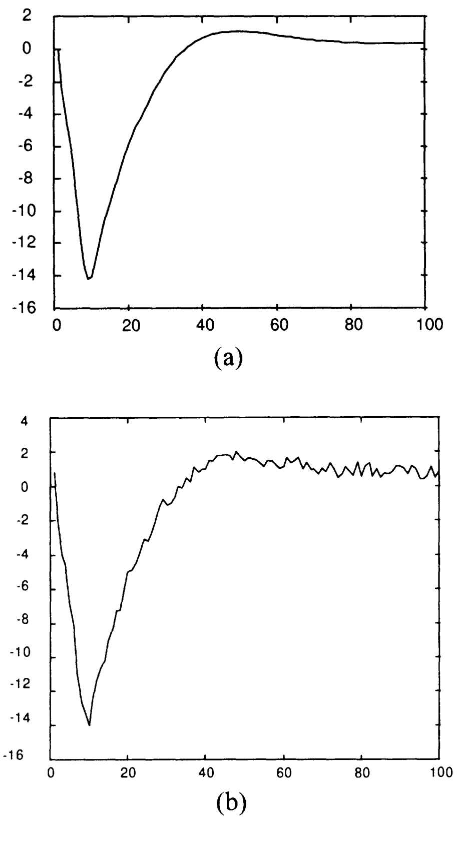


Figure 6.5 A signal from the simulator (a) and the signal with random noise (b).

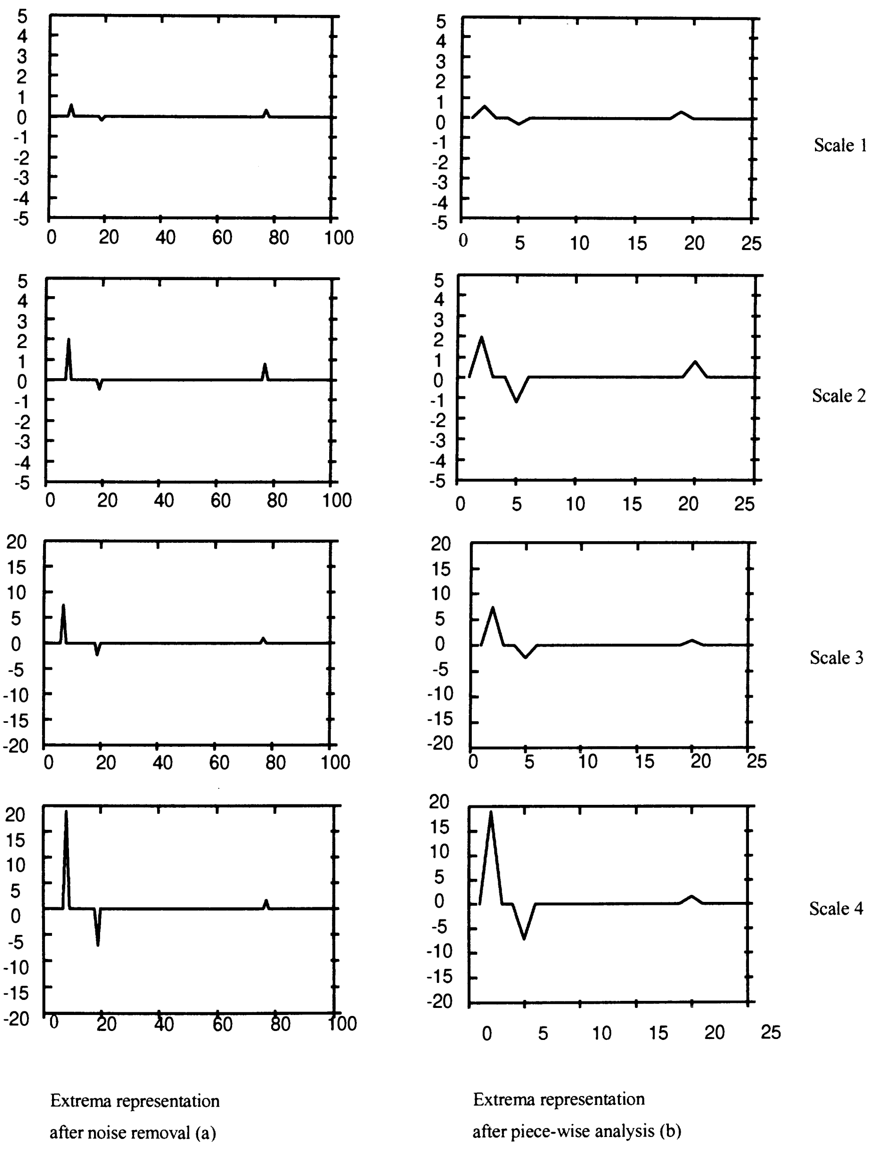


Figure 6.6 Multiresolution analysis (left) and extrema (right).

As stated in Chapter 3, the extrema that are mostly influenced by noise fluctuations are those (1) where the amplitude decreases on average as the decomposition scale increases and (2) do not propagate to large scales. Using these criteria, noise extrema are removed.

The extrema representation after the noise extrema are removed is a sparse vector, so a piece-wise technique is employed to reduce the dimensionality of the signal. The extrema after removing noise and carrying out piece-wise processing are shown in Figure 6.7.

Figure 6.8 shows the result after noise removal and compares it with the multiresolution analysis of the original noise-free signal. The extrema are the same in positions but are slightly different in value. The noise removal algorithm is therefore suitable in this case and the 4th scale extrema are selected as input to the ARTnet for pattern identification

It is important that a suitable threshold for pattern recognition is used when applying ARTnet. For a threshold ρ = 0.8, all 64 data patterns are identified as individual patterns. A more suitable threshold is obtained by analysing clustering results for increased threshold values as shown in Table 6.1.

Table 6.1 ARTnet clustering result using different distance threshold[a].

Threshold ρ	Number of patterns identified	Identified grouping of data samples
0.8	64	
1.0	63	[56 57]
2.0	60	[5 7] [25 26] [27 28] [56 57]
3.0	57	[5 7] [19 20 23 24] [25 26] [27 28] [56 57]
4.0	54	[5 6 7 8] [19 20 21 23 24] [25 26] [27 28] [56 57]
4.5	49	[3 4 5 6 7 8 9] [19 20 21 22 23 24] [25 26] [27 28]
		[35 36] [56 57]
5.0	48	[3 4 5 6 7 8 9] [19 20 21 22 23 24 29]
		[25 26] [27 28] [35 36] [56 57]
6.0	47	[3 4 5 6 7 8 9] [19 20 21 22 23 24 29 61]
		[25 26] [27 28] [35 36] [56 57]

a - [56 57] means that data patterns 56 and 57 are identified in the same cluster.

With the threshold ρ increased to 1.0, data patterns 56 and 57, which represent cases where the opening of valve 401-ST is decreased from 100% by 80% and 90% are grouped together. When ρ is 2.0, further groupings are [5, 7], representing the fresh feed flowrate increasing by 50% and 70%, [25, 26] recycle oil flowrate increasing by 70% and 90%, and [27, 28] recycle oil flowrate decreasing by 70% and 90%. It is obvious that these are all reasonable groupings.

When the threshold value is 4.5, the groupings are [3,4,5,6,7,8,9], [19 20 21 22 23 24], [25 26], [27, 28], [35,36] and [56 57]. The pairing of identified patterns and original data patterns are shown in Table 6.2. The clustering is justified by inspecting the results in detail. Figure 6.9 shows the trends of three measurements for data pattern 5. It shows that regenerator temperature and concentration of oxygen in regenerator flue gas drop sharply while catalyst hold-up in reactor increases dramatically. All of which mean abnormal operations. Very similar scenarios can be found for data patterns 3, 4, 5, 6, 7, 8, and 9, so the result of regarding them as a single pattern is acceptable. The grouping [35, 36] can also be justified by inspecting the dynamic responses (Figure 6.10). In both cases, the dynamic responses of catalyst recycle rate lead to a steady state with the process remaining under control.

Table 6.2 ATRnet identified clusters when the distance threshold is 4.5 and the corresponding data patterns^a.

Identified clusters	Corresponding data patterns	Identified clusters	Corresponding data patterns	Identified clusters	Corresponding data patterns
1	1	**19**	32	**37**	51
2	2	**20**	33	**38**	52
3	[3 4 5 6 7 8 9]	**21**	34	**39**	53
4	10	**22**	[35 36]	**40**	54
5	11	**23**	37	**41**	55
6	12	**24**	38	**42**	[56 57]
7	13	**25**	39	**43**	58
8	14	**26**	40	**44**	59
9	15	**27**	41	**45**	60
10	16	**28**	42	**46**	61
11	17	**29**	43	**47**	62
12	18	**30**	44	**48**	63
13	[19 20 21 22 23 24]	**31**	45	**49**	64
14	[25 26]	**32**	46		
15	[27 28]	**33**	47		
16	29	**34**	48		
17	30	**35**	49		
18	31	**36**	50		

a - [3 4 5 6 7 8 9] means data patterns 3 to 9 are identified in the same cluster

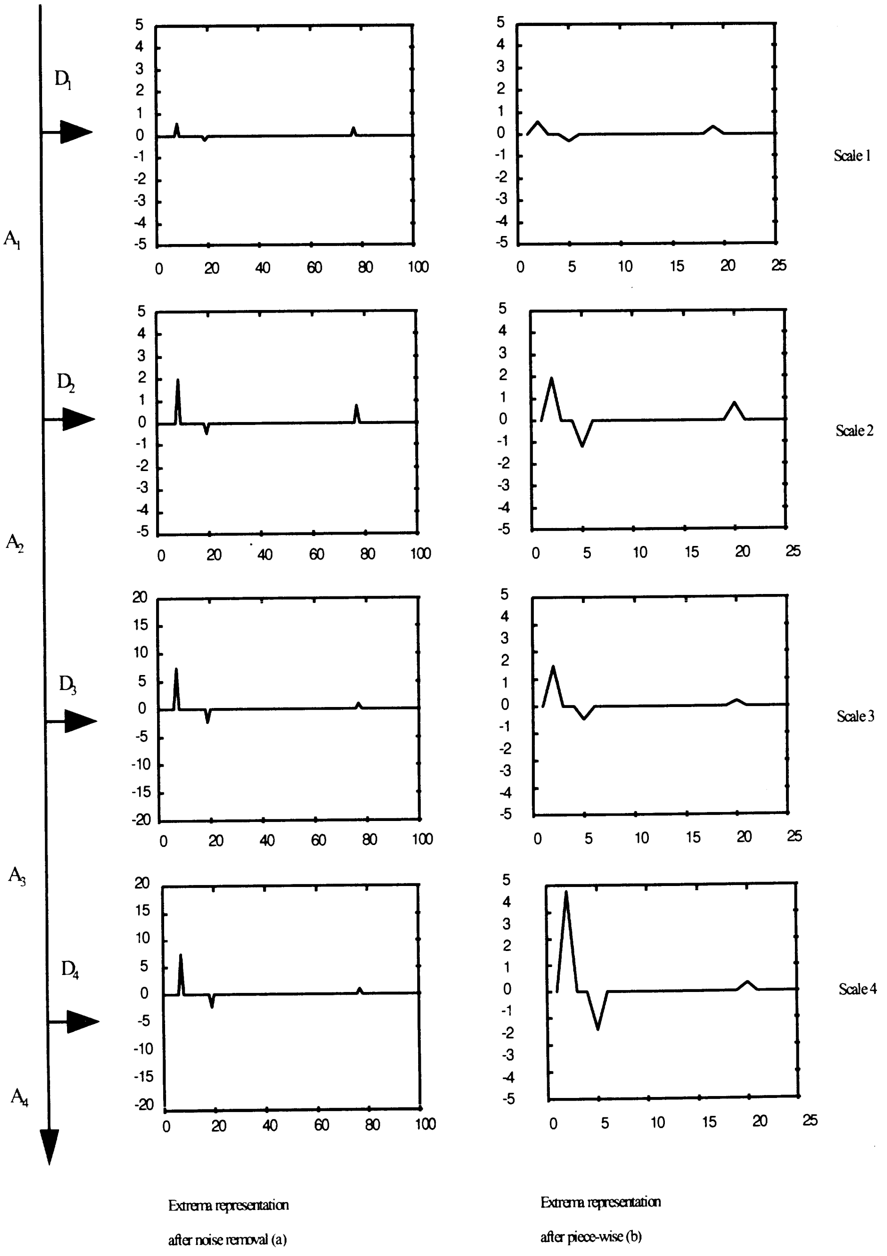
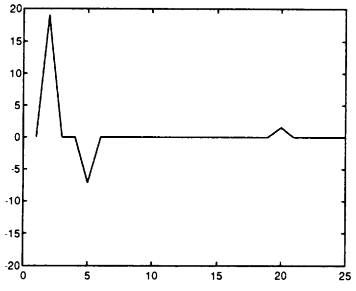
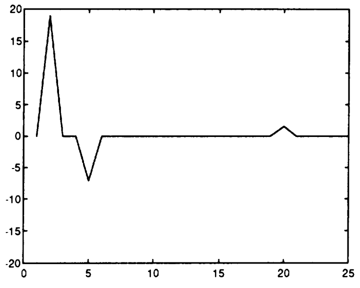


Figure 6.7 Extrema after removing noise (left) and piece wise analysis (right). D_i - detail of multiscale wavelet analysis; A_i - approximation.

However, any further increase in threshold is not useful because some data patterns that are significantly different are grouped in the same cluster. For instance, when the threshold value is 5, data pattern 29 (opening ratio of the hand-valve V20 increasing by 5%) is merged with the clusters representing increase and decrease in the preheat temperature of the mixed feed. Therefore, the threshold $\rho = 4.5$ is considered as the most appropriate value for this case.



(a)



(b)

Figure 6.8 Comparison of the result after noise removal (b) with the multiresolution analysis of the original simulation signal (a).

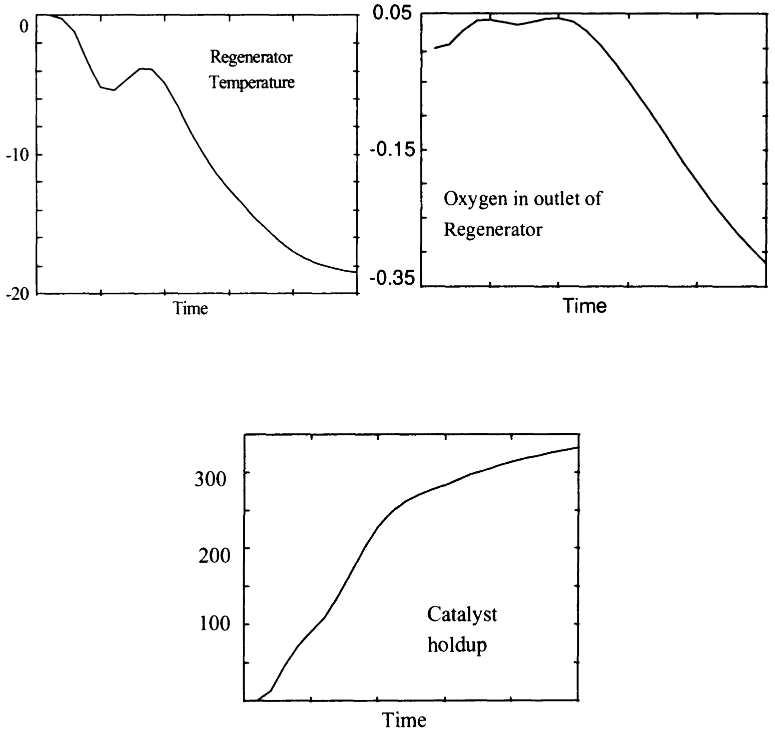


Figure 6.9 Variable dynamic trends of data pattern 5.

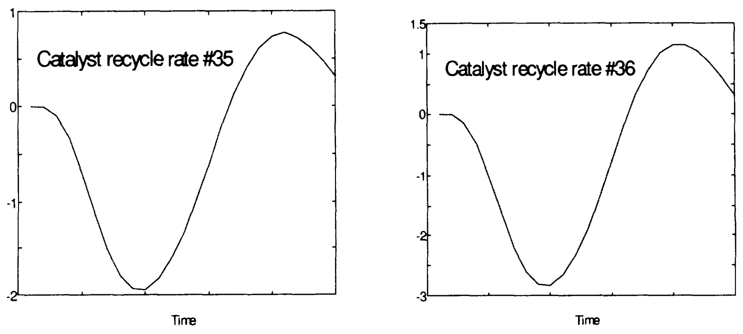


Figure 6.10 Dynamic trends of catalyst recycle rate for data patterns 35 and 36.

6.4.1 Comparison between ARTnet and ART2

It is apparent that the data pre-processing part of ARTnet is able to effectively reduce the dimension of the dynamic trend signals using wavelet feature extraction and piece wise processing. ARTnet has also shown other advantages over ART2 in operational data analysis. These include the determination of threshold values, the ability to deal with noise and computational speed. In the comparison followed only the first fifty seven data patterns were used.

6.4.1.1 Threshold Determination

In this case, only 57 data patterns are used to compare the distance threshold for using ARTnet and the vigilance value in ART2 using noise-free data. For noise free data, ARTnet and ART2 give the same results if the ARTnet distance threshold and the ART2 vigilance are appropriately adjusted, as shown in Table 6.3. To understand the table, consider the last row, which shows that when the distance threshold of ARTnet is 4.5 it gives the same grouping result as ART2 with a vigilance value of 0.9985. From Table 6.3, for the same groupings, the ARTnet distance threshold changes from 0.8 to 4.5 while the vigilance of ART2 varies from 0.9998 down to 0.9985. So the distance threshold for ARTnet is less sensitive than the vigilance of ART2. The ART2 clustering is too sensitive to the vigilance value, making it difficult to set a value.

6.4.1.2 Robustness with Respect to Noise

The following demonstrates that ARTnet gives consistent clustering result regardless of the magnitude of noise to signal ratio, providing it is in a reasonable range. ART2 gives fewer clusters at a low noise to signal ratio and more clusters at a larger ratio. 57 data patterns are considered with white noise added. A constant C_{noise} is introduced to control the magnitude of noise defined by

$$\text{The magnitude of noise} = \frac{\begin{array}{c}\text{The magnitude of noise from}\\ \text{the noise generator}\end{array}}{C_{noise}} \tag{6.5}$$

Table 6.3 Comparison of the value ranges of the distance threshold of ARTnet and the vigilance value of ART2, for the same grouping schemes[a,b,c].

ARTnet distance threshold	ART2 vigilance value	Grouping of data samples
0.8	0.9998	
1.0	0.9996	[56 57]
2.0	0.9992	[5 7] [25 26] [27 28] [56 57]
3.0	0.9990	[5 7] [19 20 23 24] [25 26] [27 28] [56 57]
4.0	0.9987	[5 6 7 8] [19 20 21 23 24] [25 26] [27 28] [56 57]
4.5	0.9985	[3 4 5 6 7 8 9] [19 20 21 22 23 24] [25 26] [27 28] [35 36] [56 57]

[a] [56 57] means that data patterns 56 and 57 are grouped in the same cluster, [b] Only the first 57 data patterns are considered and the data is noise free, [c]The ARTnet distance threshold changes in a wider range while ART2 vigilance is too sensitive making it difficult to set a value.

Table 6.4 Clusters predicted by ARTnet when the distance threshold is 4.5 and C_{noise} varies over a wide range, from 0.001 to 100[a].

Identified patterns	Corresponding data patterns	Identified patterns	Corresponding data patterns	Identified patterns	Corresponding data patterns
1	1	**15**	[27, 28]	**29**	43
2	2	**16**	29	**30**	44
3	[3 4 5 6 7 8 9]	**17**	30	**31**	45
4	10	**18**	31	**32**	46
5	11	**19**	32	**33**	47
6	12	**20**	33	**34**	48
7	13	**21**	34	**35**	49
8	14	**22**	[35 36]	**36**	50
9	15	**23**	37	**37**	51
10	16	**24**	38	**38**	52
11	17	**25**	39	**39**	53
12	18	**26**	40	**40**	54
13	[19 20 21 22 23 24]	**27**	41	**41**	55
14	[25 26]	**28**	42	**42**	[56 57]

[a][3 4 5 6 7 8 9] means that data patterns 3 to 9 are grouped in the same cluster.

In Equation 6.5, C_{noise} changes ranging from 0.001 to 100 are examined in what follows where the smaller the C_{noise}, the larger the noise to signal ratio.

The best clustering results are obtained when the distance threshold of ARTnet is 4.5. This result is not affected by changing C_{noise} from 0.001 to 100, as can be seen in Table 6.4. For ART2, the best value of the vigilance is 0.9985 and $C_{noise} = 100$, and is the same result as ARTnet (Table 6.4). However, as C_{noise} decreases to 10, i.e., larger noise to signal ratio, ART2 splits the cluster [3 4 5 6 7 8 9] into two [3 4 5 6 7] and [8 9]. As C_{noise} decreases to 0.001, i.e., a much larger noise to signal ratio, there are further new groupings, [20 42] and [29 51]. The new groups are not able to be satisfactorily explained. Although the inappropriate groupings [20 42] and [29 51] can be avoided by changing the vigilance value, other unreasonable groupings are generated.

6.3.1.3 Computational Speed

It is found that ARTnet is faster than ART2. After optimum values of the distance threshold of ARTnet and the vigilance of ART2 are found, for the same data, ARTnet is typically two times faster than ART2.

6.5 Bayesian Automatic Classification

6.5.1 The Bayesian Automatic Classification System - AutoClass

The approach used is based on an unsupervised Bayesian classification scheme developed by NASA [39, 40, 41, 42]. For a given number of data patterns (some times called cases, observations, samples, instances, objects or individuals), each of which is described by a set of attributes, AutoClass can devise an automatic procedure for grouping the data patterns into a number of classes such that instances within a class are similar, in some respect, but distinct from those in other classes. The approach has several advantages over other clustering methods.

- The number of classes is determined automatically. Deciding when to stop forming classes is a fundamental problem in classification [188]. More classes can often explain the data better, so it is necessary to limit the number of classes. Many systems rely on an *ad hoc* stopping criterion. For example, ART2 (Adaptive Resonance Theory) is strongly influenced by a vigilance or threshold value which is set by users based on trial and error. The Kohonen network requires the number of classes to be determined beforehand. The Bayesian solution to the problem is based on the use of prior knowledge. It assumes that simpler class hypotheses (e.g., those with fewer classes) are more likely than complex ones, in advance of acquiring any data, and the *prior probability* of the hypothesis reflects this preference. The prior probability term prefers fewer classes, while the likelihood of the data prefers more, so both effects balance at the most probable number of classes. Because of this, AutoClass finds only one class in random data.

- Objects are not assigned to a class absolutely. AutoClass calculates the probability of membership of an object in each class, providing a more intuitive classification than absolute partitioning techniques. An object described equally well by two class descriptions should not be assigned to either class with certainty, because the evidence cannot support such an assertion.
- All attributes are potentially significant. Classification can be based on any or all attributes simultaneously, not just the most important one. This represents an advantage of the Bayesian method over human classification. In many applications, classes are distinguished not by one or even by several attributes, but by many small differences. Humans often have difficulty in taking more than few attributes into account. The Bayesian approach utilises all attributes simultaneously, permitting uniform consideration of all the data. At the end of learning, AutoClass gives the contributing factors to class formation.
- Data can be real or discrete. Many methods have difficulty in analysing mixed data. Some methods insist on real valued data, while others accept only discrete data. The Bayesian approach can utilise the data exactly as they are given.
- It allows missing attribute values.

6.5.2 Overview of Bayesian Classification

AutoClass is based on Bayes's theorem, for combining probabilities. Given observed data D and a hypothesis H, it states that the probability that the hypothesis explains the data $p(H \mid D)$, (called the *posterior* probability of the hypothesis given the data) is proportional to the probability of observing the data if the hypothesis were known to be true $p(D \mid H)$ (the *likelihood* of the data) times the inherent probability of the hypothesis regardless of the data ($p(H)$, the *prior* probability of the hypothesis). Bayes's theorem is commonly expressed as,

$$p(H|D) = \frac{p(H)p(D/H)}{p(D)} \tag{6.6}$$

For classification, the hypothesis H is the number and descriptions of the classes from which the data D is believed to have been drawn. Given D, the goal is to determine H so as to maximise the posterior $p(H|D)$. For a particular classification hypothesis, calculation of the likelihood of the data $p(D/H)$ involves a straightforward application of statistics. The prior probability of the hypothesis $p(H)$ is less transparent and is taken up later. Finally, the prior probability of the data, $p(D)$ in the denominator above, need not be calculated directly. It can be derived as a normalising constant or ignored so long as only the relative probability of hypotheses is considered.

6.5.3 Application to Classification.

The fundamental model of AutoClass is the classical finite mixture model of Everitt and Hand [188] and Titterington et al. [189], made up of two parts. The first is the probability of an instance being drawn from a class C_s (s = 1, k), denoted λ_s. Each class C_s then is modelled by a class distribution function, $p(x_i \mid x_i \in C_s, \theta_s)$, giving the probability distribution of attributes conditional on the assumption that instance x_i belongs to class C_s. These class distributions are described by a *class parameter vector,* θ_s, which for single attribute normal distribution consists of the class mean, μ_s, and variance σ_s^2.

Thus, the probability of a given datum coming from a set of classes is the sum of the probabilities that it came from each class separately, weighted by the class probabilities.

$$p(x_i \mid \theta, \lambda, k) = \sum_{s=1}^{k} \lambda_s \, p(x_i \mid x_i \in C_s, \theta_s) \tag{6.7}$$

It is assumed that the data is unordered and independent, given the model. Thus the *likelihood* of measuring an entire database is the product of the probabilities of measuring each object

$$p(\boldsymbol{x} \mid \theta, \lambda, k) = \prod_{i=1}^{n} p(x_i \mid \theta, \lambda, k) \tag{6.8}$$

For a given value of the class parameters, the probability that instance i belongs to a class using Bayes's theorem is calculated using

$$p(x_i \in C_s \mid x_i, \theta, \lambda, k) = \frac{\lambda_s \, p(x_i \mid x_i \in C_s, \theta_s)}{p(x_i \mid \theta, \lambda, k)} \tag{6.9}$$

These classes are "fuzzy" in the sense that even with perfect knowledge of object attributes, it will be possible to determine only the probability that it is a member of a given class.

The problem of identifying a mixture is done in parts: determining the classification parameters for a given number of classes and the number of classes. Rather than seeking an *estimator* of the classification parameters (i.e., the class parameter vectors, θ, and the class probabilities, λ), the full *posterior* probability distribution is sought. The posterior distribution is proportional to the product of the prior distribution of the parameters $p(\theta, \lambda \mid k)$ and the likelihood function $p(\boldsymbol{x} \mid \theta, \lambda, k)$.

$$p(\theta, \lambda \mid \boldsymbol{x}, k) = \frac{p(\theta, \lambda \mid k) \; p(\boldsymbol{x} \mid \theta, \lambda, k)}{p(\boldsymbol{x} \mid k)} \tag{6.10}$$

The pseudo-likelihood $p(\boldsymbol{x} \mid k)$ is simply the normalising constant of the posterior distribution, obtained by normalising (integrating) out the classification parameters - in effect, treating them as "nuisance" parameters:

$$p(\boldsymbol{x} \mid k) = \iint p(\theta, \lambda \mid k) \; p(\boldsymbol{x} \mid \theta, \lambda, k) \, d\theta \, d\lambda \, . \tag{6.11}$$

To solve the second half of the classification problem (i.e., determining the number of classes k) the posterior distribution of the number classes k has to be calculated. This is proportional to the product of the prior distribution $p(k)$ and the pseudo- likelihood function $p(\boldsymbol{x} \mid k)$.

$$p(k \mid \boldsymbol{x}) = \frac{p(k) \, p(\boldsymbol{x} \mid k)}{p(\boldsymbol{x})} \tag{6.12}$$

In principle, the most probable number of classes are determined by evaluating $p(k \mid \boldsymbol{x})$ over the range of k for which the prior $p(k)$ is significant . In practice, the multi-dimensional integrals of Equation 6.6 are computationally intractable, and the maximum of the function has to be found so that it can be approximated at about that point.

6.5.4 The AutoClass Attribute Model

In AutoClass, it assumed that attributes are independent for each class. This permits an extremely simple form for the class distributions used in Equation 6.2.

$$p(x_i \mid x_i \in C_s, \theta_s) = \prod_{j=1}^{m} p(x_{ij} \mid x_i \in C_S, \theta_{sj}) \tag{6.13}$$

where θ_{sj} is the parameter vector describing the j th attribute in the sth class C_s. AutoClass models for real valued attributes are Gaussian normal distributions parameterised by a mean and a standard deviation , and thus θ_{sj} takes the form

$$(\theta_{sj}) = \begin{vmatrix} \mu_{sj} \\ \sigma_{sj} \end{vmatrix}$$

The class distribution is thus

$$p(x_{ij} \mid x_i \in C_s, \mu_{sj}, \sigma_{sj}) = \frac{1}{\sqrt{2\pi\sigma_{sj}}} \exp\left[\frac{-1}{2}\left(\frac{x_{ij} - \mu_{sj}}{\sigma_{sj}}\right)^2\right] \tag{6.14}$$

6.5.5 Search Algorithm

As mentioned earlier AutoClass breaks the classification problem into two parts: determining the number of classes and determining the parameters defining them. It uses a Bayesian variant of Dempster and Laird's EM (expectation and maximisation) algorithm [190] to find the best class parameters for a given number of classes. To derive the algorithm, the posterior distribution is differentiated with respect to the class parameters and equate with zero. This yields a system of non-linear equations which hold at the maximum of the posterior:

$$\hat{\lambda}_s = \frac{W_S + w' - 1}{n + k(w' - 1)} \qquad s = 1 \ldots k \tag{6.15}$$

$$\frac{\partial}{\partial \theta_s} \ln p(\hat{\theta}_s) + \sum_{i=1}^{n} w_{is} \frac{\partial}{\partial \theta_s} \ln p(x_i \mid \hat{\theta}_s) = 0 \tag{6.16}$$

where w_{is} is the probability that the datum, x_i, was drawn from class s (given by Equation 6.4) and W_S is the total weight for class C_s :

$$w_{is} = p(x_i \in C_S \mid x_i, \hat{\theta}, \hat{\lambda})$$

$$W_s = \sum_{i=1}^{n} w_{is}$$

To find a solution to this system of equations is found to be iterative based on Equations 6.10 and 6.11 (treating w as a constant) and Equation 6.4 (treating λ and θ as constants).

For any given iteration, the membership probabilities are constant, so Equation 6.11 can be simplified by using w_{is} via the derivative, so

$$\frac{\partial}{\partial\theta_s}[\, p(\hat{\theta}_s) \prod_{i=1}^{n} p(x_i \mid \hat{\theta}_s, x_i \in C_s)^{w_{is}} \,] = 0 \qquad (6.17)$$

Thus far, the discussion of the search algorithm has related to general class model with an arbitrary θ_{sj}. The Equation 6.12 is now applied to the specific AutoClass - model of Equation 6.8 through 6.9.

For real valued attributes, the equations for the updated $\hat{\mu}_{sj}$ and $\hat{\sigma}_{sj}$ are a function of the prior information and the empirical mean, $\bar{x}_{sj}$ and σ^2_{sj} of the jth attribute in class C_s, weighted by w_{is} :

$$\bar{x}_{sj} = \frac{\sum_{i=1}^{n} w_{is} x_{ij}}{W_s}$$

$$\sigma^2_{sj} = \frac{\sum_{i=1}^{n} w_{is} x^2_{ij}}{W_s} - \bar{x}^2_{sj}$$

The update formulas are then:

$$\hat{\mu}_{sj} = \frac{w'\bar{x}_j' + W_s\bar{x}_{sj}}{w' + W_s} \qquad s = 1\ldots k,\ j = 1\ldots m \qquad (6.18)$$

$$\hat{\sigma}^2_{sj} = \frac{w'(\sigma_j')^2 + W_s\sigma^2_{sj}}{w' + W_s + 1} + \frac{w'\,W_s}{(w' + W_s)(w' + W_s + 1)}(\sigma_j' - \bar{x}_{sj})^2 \qquad s = 1\ldots k$$

(6.19)

Equations 6.10, 6.13 and 6.14 do not, of course, give the estimators explicitly; instead they must be solved using an iterative procedure [191]. The simplest way of estimating parameters using *maximum likelihood estimate* method is that suggested by Wolfe [192] which is essentially an application of EM algorithm [190]. Whereas, by the *Bayesian parameter estimation* method, AutoClass uses a Bayesian variant of

Dempster and Laird EM algorithm. Initial estimates of the λ_s, μ_s, σ_{sj}^2 are obtained by one of the variety of methods [188], and these are then used to obtain first estimates of the $p(s|x_i)$ i.e., the weights w_{is} and hence W_s, - the E-step; these are then inserted into Equations 6.10, 6.13 and 6.14 to give revised parameter estimates, which is essentially the M-step. The process is continued until some convergence criterion is satisfied [188].

6.5.6 The EM Algorithm

The steps of EM algorithm are:

E- STEP : The starting values of μ, σ, λ are obtained using a variety of cluster analysis methods and are then used to obtain first estimates of the weights $\hat{w}$ (which is an estimate of $\boldsymbol{w}$) using Equation 6.4.

For instance assuming that the data has 3 classes (k=3), i.e., $\lambda = \lambda_1, \lambda_2$ and λ_3, two attributes $j=1,2$, then there are $3*2=6$ μ 's and 6 σ's. In total there are $6+6+3=9$ parameters to estimate, yielding 3 weights $w_{i,s=1}, w_{i,s=2}$ and $w_{i,s=3}$ for observation i.

M-STEP : This step requires the calculation of μ, σ, λ using Equations 6.10, 6.13 and 6.14.

The iteration of the two steps is then continued until the parameters are maximised.

6.6 Application of AutoClass to the FCC Process

Using the same FCC process as the case study, 42 data patterns are studied, which are summarised in Table 6.5. For every variable in each of the data patterns, 60 sampling points are recorded. For example, the dynamic trend represented by 15 (reaction temperature) in Figure 6.11 is composed of 60 data points when the valve opening on the top of the distillation column changes from 100% to 90%. Six process parameters are recorded including reaction and regeneration temperatures (TRA and TRG), reactor and regenerator pressures (PRA and PRG), oxygen and carbon monoxide volumetric contents in the flue gas from the regenerator (PTO2 and PTCO). These are known to be the major variables for the FCC process, although more precise characterisation would be expected if more parameters were included.

Table 6.5 Summary of the forty two data patterns.

Cases	Description of cases
1 ~ 11	Normal operation
12	Fresh feed pump (P1321) failure
13	A step increase of 15% in fresh feed flowrate
14	A step decrease of 50% in fresh feed flowrate
15	The valve opening on top of the distillation column changes from 100% to 90%
16	.... 100% to 80%
17	.... 100% to 60%
18	.... 100% to 40%
19	.... 100% to 30%
20	.... 100% to 20%
21	Manual valve (V20) controlling catalyst to heat removal system, 75%→80%
22	.... 75% → 90%
23	.... 75% → 100%
24	.... 75% → 60%
25	.... 75% → 50%
26	.... 75% → 40%
27	.... 75% → 35%
28	.... 75% → 30%
29	Recycle sludge oil flowrate increased to 300%
30	Fresh feed flowrate decreased to 10% of that of normal operation
31	Recycle sludge oil pump (P1329) failure
32	Fresh feed flowrate increased by 15%
33 ~ 42	Normal operation

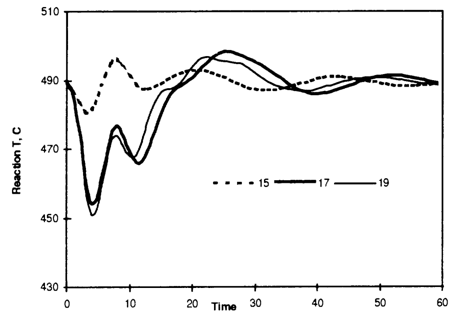


Figure 6.11 Reaction temperature response for cases 15, 17 and 19.

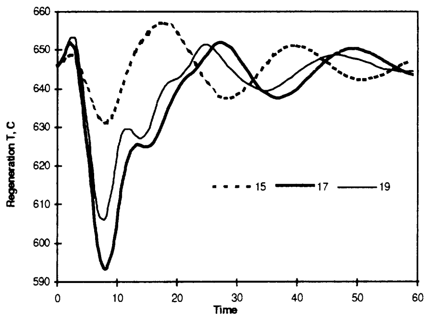


Figure 6.12 Regeneration temperature response for cases 15, 17 and 19.

Clearly prior knowledge about the data makes it possible to test the automatic classification capabilities. Since each data instance involves six variables and each variable is represented by 60 data points, the data base is a 360 x 42 matrix. In

Chapter 3, various methods have been introduced for reducing the dimensionality of a dynamic trend without signal losing its important features. These include wavelets, principal components and episode representations. Here the crude data are used since the concern is the identification of operational states.

6.6.1 Result of Classification

Although the classification using AutoClass is an automatic process, it still has many options which allow users to exercise control over the learning and the way in which the results are output [193]. For example, choices can be made of the attribute probability models including

(i) the single multinomial model which implements a single multinomial likelihood model term for symbolic or integer attributes, which is conditionally independent of other attributes given the class.

(ii) the single normal CN model which uses real valued attributes with a conditionally independent Gaussian normal distribution, assuming no missing values.

(iii) the single normal CM Model which also models real valued attributes but allows missing values.

(iv) the multi normal CN covariant model, also for real valued attributes expresses mutual dependencies within the class. The probability that the class will produce any particular instance is then the product of any independent and covariant probability terms.

In this case the single normal CN model is used. The approach is based on generation of alternative classification schemes which are ranked (the first being the best). It then remains to analyse the classification schemes to determine which is the most acceptable. The best for the current problem is shown in Table 6.6.

Table 6.6 The AutoClass clustering results of the 42 cases in Table 6.5.

Classes	Cases
1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42
2	21, 22, 23 31, 32 13, 29
3	15, 16, 17, 18, 19, 20
4	24, 25, 26, 27, 28
5	12, 14, 30

6.6.3 Analysis of the Classification Results

The classified results are analysed by comparing Table 6.6 and Table 6.5. Class I includes cases 1 ~ 11 and 33 ~ 42, which correspond to normal operations as can be seen from Table 6.5 so it is reasonable to assign them into a single class. The significance of being able to automatically distinguish between normal and abnormal operational data, which represent moderate to significant disturbances as well as faults is that process upsets can be seen.

Class III includes cases 15 ~ 20 which correspond to decreases in the opening of the valve 401-ST opening from 100% to 90%, 80%, 60%, 40%, 30% and 20% which cause the differential pressure (PRG - PRA) between the regenerator and reactor to decrease. Consequently, the regenerated catalyst circulation rate falls and the reaction and regeneration temperatures will be influenced. The closed - loop dynamic responses of reaction and regeneration temperatures for cases 15, 17 and 19 are shown in Figures 6.11 and 6.12. It can be seen that although they are not identical, they show greater similarity than those in class I, i.e., normal operation.

Class IV includes cases 24 ~ 28 which correspond to the changes in the opening of the hand-operated valve V20, which first cause the regeneration temperature to change due to changes in heat transfer and then affect all other parameters. Cases 24 ~ 28 refer to a reduction in the opening of V20 from its normal operation value (75%) to 60%, 50%, 40%, 35% and 30%. All these operations cause the regeneration temperature to increase, so is quite reasonable that they should be grouped into one class. The regeneration temperature TRG and oxygen volumetric percentage in flue gas PTO2 change are shown in Figures 6.13 and 6.14. It is important that cases 21, 22 and 23 are grouped in a different class (class II) although they also represent V20 opening changes, because they represent increases in opening which have different effects on associated variables. Figure 6.15 shows the PTO2 changes for cases 21, 22 and 23. It is clear that they are different from cases 24, 26 and 28 shown in Figure 6.14.

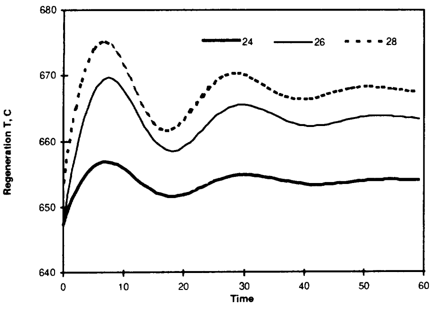


Figure 6.13 Regeneration T responses for cases 24, 26 and 28.

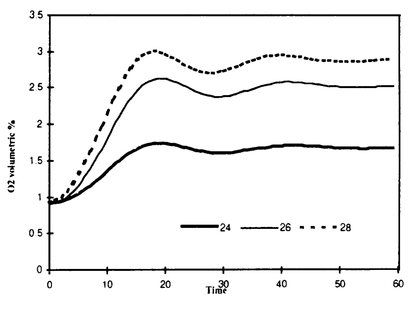


Figure 6.14 Dynamic responses of the O2 content in flue gas for cases 24, 26 and 28.

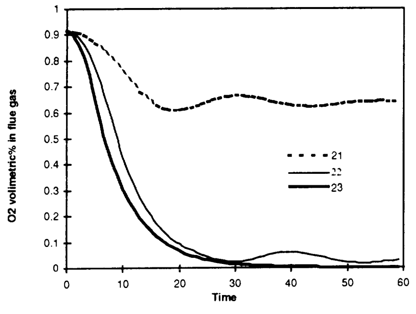


Figure 6.15 Dynamic responses of the O2 content in flue gas for cases 21, 22 and 23.

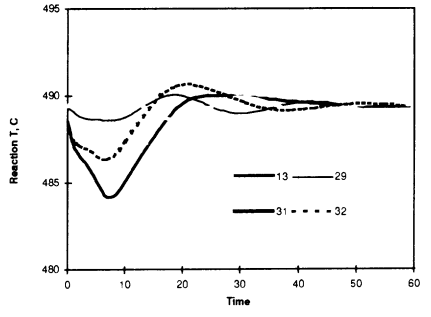


Figure 6.16 Reaction temperature responses for cases 13, 29, 31 and 32.

Class V includes 12, 14 and 30 which are clearly in one class and represent feed pump P1329 failure, feed flowrate decreases by 50% and 90%, all mean a sharp decrease in feed flowrate.

Class II includes 21, 22, 23, 31, 32, 13 and 29. As discussed above, it is reasonable that 21, 22 and 23 should be grouped together since they represent increases in the opening of V20 which cause decrease in regeneration temperature. Both 13 and 32 represent slight increases (15% and 9%) in fresh feed flowrate, while 29 a three times increase in sludge oil flowrate and case 31 sludge oil pump failure. Since sludge oil flow is only very small compared with the fresh feed at normal operations (8000 kg/hr sludge oil vs 150,000 kg/hr fresh feed), it is not surprising that a factor of three time increase in sludge oil has a similar effect on process operation as slight fresh feed increases by 15% and 9%. The major difference between the four cases, 12, 32, 29 and 31 is that 12, 29 and 32 represent increases in feed while 31 indicates a decrease, as indicated in the early stage of reaction temperature responses (Figure 6.16). Because the process is under closed loop control, such a difference is not significant enough to regard 31 as a different class. However it is interesting that 21, 22, 23 and 31, 32, 13, 29 are grouped together which would not have been expected. The explanation is that they are not very significant disturbances and all affect in similar ways the operation of the process mainly by influencing the heat balance of the two reactors which is the dominant factor in FCC operations.

6.7 General Comments

The above discussion has introduced unsupervised machine learning as a powerful method for process operational state identification. The data pre-processing methods described in Chapter 3 have been used to reduce the dimensionality of the data and remove noise before analysis of data using unsupervised machine learning. There are several issues that are important but have not been fully addressed. First, for on-line process monitoring, it is important for the approach to be recursive. ART2 is an recursive method, but AutoClass is not. Second, although unsupervised procedures do not need training data, they are usually not as accurate as supervised methods, therefore interpretation and validation of results becomes an important issue. Furthermore when adapted to on-line monitoring the speed is obviously critical as is the selection of variables used for classification. Process variables tend to be interrelated so it is necessary to remove redundant variables without losing the important ones.

CHAPTER 7
INDUCTIVE LEARNING FOR CONCEPTUAL CLUSTERING AND REAL-TIME PROCESS MONITORING

Multivariate statistics, supervised and unsupervised machine learning approaches have been introduced in previous chapters as a basis for developing process monitoring systems. The approaches often depend on calculating a similarity or distance measure for identifying clusters in data. Apart from giving predictions, however they are not able to provide causal explanations on why a specific set of data is assigned to a particular cluster. In this chapter, conceptual clustering based on an inductive machine learning approach is introduced for use in designing state space based process monitoring systems. As distinguished from similarity or distance based clustering, such conceptual clustering is able to generate conceptual knowledge about the major variables which are responsible for clustering, as well as predicting operational states. The resulting knowledge is expressed in the form of production rules or decision trees.

The first stage is to introduce inductive learning. Following this, an example is given to demonstrate how inductive learning can be used to analyse process operational data which has been averaged over days or weeks so that the embedded information can be exploited to improve process performance. After this a section on application of inductive learning to conceptual clustering and real-time process monitoring is followed, by reference to a relatively simple case study based on a continuous stirred tank reactor (CSTR). A more complicated case study based on a refinery methyl tertiary butyl ether (MTBE) example is then presented.

7.1 Inductive Learning

Inductive learning is probably the most widely studied method based on symbolic learning [202-204]. It attempts to acquire a conceptual language for describing an object by drawing inductive inference from observations. The focus is on deriving rules or decision trees from unordered sets of examples, especially attributes based induction, a formalism where examples are described in terms of a fixed collection of attributes. The discussion excludes such learning methods as feedforward neural networks which learn to develop an implicit rather than explicit and transparent rules or decision trees. An obvious motivation for inductive learning is that it provides a method for solving the bottleneck arising from knowledge acquisition which is necessary to develop expert systems. It is relatively easy for human experts to document cases rather than for them to articulate the expertise explicitly and clearly. Several approaches to inductive learning have been proposed, such as AQ11 [196], VersionSpaces [197] and C5.0 [18, 19, 20]. Here the focus is on C5.0, a system that is designed to learn to develop rules and decision trees from examples.

7.1.1 The Inductive Learning System

The conceptual clustering approach used in C5.0 was developed by Quinlan [18, 19, 20]. Given a database of objects (or in other words data sets) which are described in terms of a collection of attributes, which measure some important feature of an object. Each object belongs to one of a set of mutually exclusive classes, the task is to develop a classification rule that can determine the class of any object from its values of the attributes. The decision tree generated can be used for conceptual clustering. The procedure is iterative and can be summarised as follows [18, 19]:

```
(1) Select a random subset of the given training examples (called the window)
(2) Repeat (a) to (c)
    (a) Develop a decision tree which correctly classifies all objects in the
        window
    (b) Find exceptions to this decision tree in the remaining examples
    (c) Form a new window by adding incorrectly classified objects to the
        window
```

Until there are no exceptions to the decision tree.

The crux of the problem is how to develop a decision tree for an arbitrary collection of objects in the window. To form a decision tree requires selecting the root attribute. To do this, assume that there are only two classes representing all the data, P and N (extension to any number of classes is not difficult). The method of finding the root attribute is adopted from an information based method that depends on two assumptions. Suppose the window C contains p objects of class P and n objects of class N. The assumptions are:

(1) Any correct decision tree for the window C will classify objects in the same proportion as their representation in C. An arbitrary object will be determined as belonging to class P with probability p/(p+n*)* and to class N with probability n/(p+n*).*

(2) When a decision tree is used to classify an object, it returns a class. A decision tree can then be regarded as the source of a message 'P' or 'N'. with the expected information needed to generate this message given by

$$I(p, n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n} \qquad (7.1)$$

If attribute, A, having values $\{A_1, A_2, \ldots A_v\}$, is used for the root of the decision tree, it will partition the window C into $\{C_1, C_2, \ldots C_v\}$ where C_i contains those objects in C that have values A_i of A. Suppose C_i contains p_i objects of class P and n_i of class N. The expected information required for the subtree for C_i is $I(p_i, n_i)$ and for the tree with A as root is then obtained as the weighted average given by

$$E(A) = \sum_{i=1}^{v}\frac{p_i+n_i}{p+n}I(p_i,n_i) \qquad (7.2)$$

where the weight for the ith branch is the proportion of the objects in C that belong to C_i. The information gained by branching on A is therefore

$$\text{gain}(A) = I(p, n) - E(A) \qquad (7.3)$$

The approach calculates the gain for all attributes and chooses the attribute having the biggest gain as the root node. The root node will have as many branches as its values. The branches divide the database into a number of subsets. For each subset, the root node is obtained following the same procedure.

The approach has been used in the commercial software C5.0 [20], which has evolved from the earlier versions C4.5 [18] and ID3 [19]. A major limitation of ID3 was that it assumed that the values of all attributes are discrete, for instance a colour being red or green. C4.5 claimed to be able to deal with continuous-valued attributes, is still weak compared with the way it deals with discrete-valued attributes, as noted by Quinlan [198]. Though Quinlan [198] made a further effort to improve the method so that it could deal with continuous-valued attributes, the outcome is still not very satisfactory. Nevertheless, C5.0 has become one of the most well known tools for use in data mining and knowledge discovery, especially in domains involving only discrete values.

7.1.2 Dealing with Continuous-valued Attributes in Inductive Learning

Like most of the available inductive learning methods, C5.0 was developed for problem domains where attributes only take discrete values. The methods have proved to perform remarkably well with discrete valued attributes. However, when the problem domains contain real numbers, the performance usually decreases in terms of accuracy. Using inductive learning based on continuous-valued attributes requires discretisation of the values into a number of intervals. To deal with this, a number of approaches have been proposed.

The *class-separating method* [199] is based on the assumption that if the assignments of individual examples are known, then such knowledge can be used to discretise the attributes. An example of this is to use the range of boiling points of petroleum fractions into product grades. This approach however sometimes produces too many intervals which are not very informative. Moreover, if the assignment of individual cases is not known, the approach can't be used.

It is possible to use an *equal distance* approach to divide the values of the continuous variable between the minimum and maximum values into a fixed number of intervals. Alternatively *equal numbers* can be used to make each interval contain equal number of examples.

Another approach is the *k-nearest neighbours* approach which tries to estimate to which class the value of a specified attribute most likely belongs. It places a border between two values x_i and x_{i+1} if the estimates for them are different. The estimates are based on the assumption that the most probable class is the most common class among the k nearest examples, where k is normally a user specified parameter.

Wu [199] developed a *Bayesian discretizer*. The approach is shown to be more accurate than some other approaches but can only be used for integer values and requires the assignments for the examples to be known. The method used by C4.5 [18] is based on information gain. But this approach only provides binary discretisation.

7.1.3 Attributes That Take Values Related to Dynamic Trends

In process monitoring and control, dynamic trends of variables might be more important than the instantaneous values. The differences for the seven dynamic trends for a variable shown in Figure 3.4 can have important implications. The issue of dealing with this kind of problem has not previously been considered. An approach using principal component analysis to extract qualitative concepts from dynamic trend signals was given in Section 3.2.3.2, so will not be repeated here. In this chapter it will be shown how the concept formation can be used in inductive learning to develop conceptual clustering systems for process monitoring and diagnosis.

7.2 IL for Knowledge Discovery from Averaged Data

Saraiva [5] and Saraiva and Stephanopoulos [4] divided process plant data and monitoring and control strategies into layers as shown in Figure 7.1 and investigated the application of inductive learning to the highlighted layer in the figure based on analysis of daily/weekly averaged data as a basis to continuous process improvement. The goal is to develop a conceptual language describing the contributions of various operational parameters to one or a number of performance metrics which are product quality related.

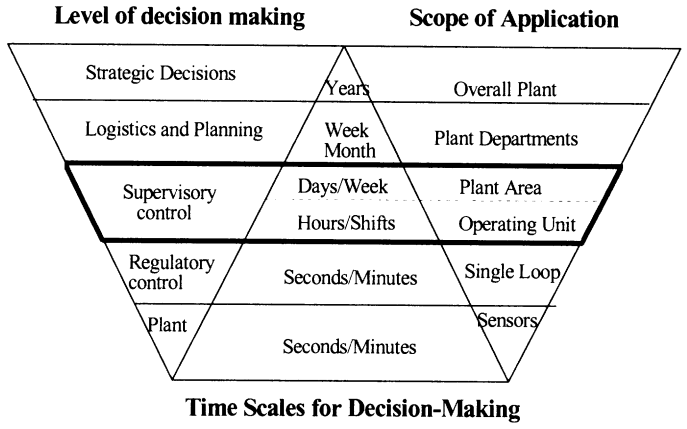


Figure 7.1 Levels, time scales, and application scopes of decision making activities.

Saraiva [5] presented four industrial case studies using this approach. A simple example is used here to illustrate the methodology. The case study is concerned with records of operating data from a refinery unit [200] shown in Table 7.1 which has five variables, x_1, x_2, x_3, x_4 and y. The latter is the octane number of the gasoline product and is discretised as very low if the value is less or equal to 91, low if it is between 91 and 92, and good if greater than 92. x_1, x_2 and x_3 are different measures of the feed composition and x_4 is the log of a combination of process conditions. Figure 7.2 shows the induced tree as well as the partition of the (x_1, x_4) plane defined by the leaves, together with a projection of all the available (**x**, y) pairs on the same plane. These two decision variables clearly influence the current performance of the refinery unit, and the decision tree leaves give a reasonable partition of the plane. To achieve better performance, operating zones that will result in obtaining mostly $y = 3$ values need to be found. Terminal nodes 2 and 7 identify two such zones. The corresponding solutions are

If $x_1 \in [44.6, 55.4]$, then conditional probability of $y = 3$ is 0.9
If $x_4 \in [1.8, 2.3]$, then the conditional probability of $y = 3$ is 1.0

Table 7.1 Data obtained in a refinery unit.

x1	x2	x3	x4	Obsv.	x1	x2	x3	x4	Obsv.
55.33	1.72	54	1.66219	92.19	71.31	3.44	55	1.60325	90.51
59.13	1.2	53	1.58399	92.74	72.3	4.02	55	1.66783	90.24
57.39	1.42	55	1.61731	91.88	68.81	6.88	55	1.69836	91.01
56.43	1.78	55	1.66228	92.8	66.61	2.31	52	1.77967	91.9
55.98	1.58	54	1.63195	92.56	63.66	2.99	52	1.81271	91.92
56.16	2.12	56	1.68034	92.61	63.85	0.24	50	1.81485	92.16
54.85	1.17	54	1.58206	92.33	67.25	0	53	1.72526	91.36
52.83	1.5	58	1.54998	92.22	67.19	0	52	1.86782	92.16
54.52	0.87	57	1.5523	91.96	62.34	0	48	2.00677	92.68
54.12	0.88	57	1.57818	92.17	62.98	0	47	1.95366	92.88
51.72	0	56	1.60401	92.75	69.89	0	55	1.89387	92.59
51.29	0	58	1.59594	92.89	73.13	0	57	1.81651	91.35
53.22	1.31	58	1.54814	92.79	65.09	1.01	57	1.45939	90.29
54.76	1.67	58	1.63134	92.55	64.71	0.61	55	1.38934	90.71
53.34	1.81	59	1.60228	92.42	64.05	1.64	57	1.33945	90.41
54.84	2.87	60	1.54949	92.43	63.97	2.8	60	1.42094	90.43
54.03	1.19	60	1.57841	92.77	70.48	4.64	60	1.5768	89.87
51.44	0.42	59	1.61183	92.6	71.11	3.56	60	1.41229	89.98
53.54	1.39	59	1.51081	92.3	69.05	2.51	60	1.54605	90
57.88	1.28	62	1.56443	92.3	71.99	1.28	55	1.55182	89.66
60.93	1.22	62	1.53995	92.48	72.03	1.28	56	1.6039	90.08
59.59	1.13	61	1.56949	91.61	69.9	2.19	56	1.67265	90.67
61.42	1.49	62	1.4133	91.3	72.16	0.51	56	1.55242	90.59
56.6	2.1	62	1.54777	91.37	70.97	0.09	55	1.45728	91.06
59.94	2.29	61	1.65523	91.25	70.55	0.05	52	1.26174	90.69
58.3	3.11	62	1.29994	90.76	69.73	0.05	54	1.28802	91.11
58.25	3.1	63	1.19975	90.9	69.93	0.05	55	1.36399	90.32
55.53	2.88	64	1.20817	90.43	70.6	0	55	1.4221	90.36
59.79	1.48	62	1.30621	90.83	75.54	0	55	1.67219	90.57
57.51	0.87	60	1.29842	92.18	49.14	0	40	2.1714	94.17
62.82	0.88	59	1.40483	91.73	49.1	0	42	2.31909	94.39
62.57	0.42	60	1.45056	91.1	44.66	4.99	42	2.14314	93.42
60.23	0.12	59	1.54357	91.74	44.64	3.73	44	2.08081	94.65
65.08	0.1	60	1.6894	91.46	4.23	10.76	41	2.1707	97.61
65.58	0.05	59	1.74695	91.44	5.53	7.99	40	1.99418	97.08
65.64	0.05	60	1.74915	91.56	17.11	5.06	47	1.61437	95.12
65.28	0.42	60	1.78053	91.9	67.6	1.84	55	1.64758	91.86
65.03	0.65	59	1.78104	91.61	64.81	2.24	54	1.69592	91.61
67.84	0.49	54	1.72387	92.09	63.13	1.6	52	1.65118	92.17
73.74	0	54	1.73496	90.64	63.48	3.46	52	1.48216	91.56
72.66	0	55	1.71966	91.09	62.25	3.56	50	1.49734	92.16

It effectively says that one should expect to get almost only "good" y values while operating inside these zones of the decision space, as opposed to the current operating conditions, which lead to just 40% "good" y values.

Saraiva [5] presented more complex case studies and methods for defining performance metrics. He used the binary discretisation mechanism for continuous-valued variables embedded in C4.5. In addition, the approach is only suitable for analysing data of daily/weekly averaged data, not on-line data from computer control systems, based on seconds.

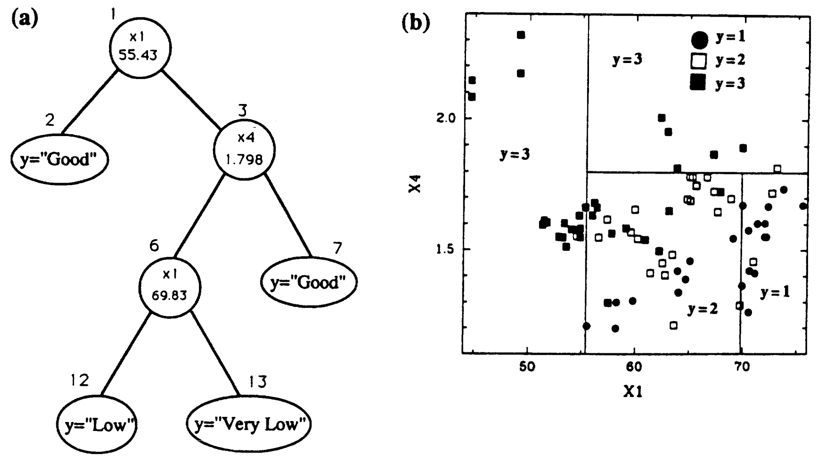


Figure 7.2 (a) Induced decision tree; (b) partition of the plane defined by its leaves.

7.3 Inductive Learning for Conceptual Clustering and Real-time Monitoring

In this section, we present our work on application of inductive learning to the analysis of data collected by computer based control systems which is on second basis. A conceptual clustering approach is thus developed for designing state space based on-line process monitoring systems. The approach is illustrated by reference to a simple case study based on a CSTR reactor (see Appendix A). It is concerned with analysis of an operational database consisting of eighty five data patterns obtained in operating the CSTR. For each data pattern seven variables are recorded including, reaction temperature TR, reaction mixture flow out of the reactor F_o, cooling water flowrate F_w, feed flowrate F_i, feed inlet temperature T_i, feed concentration C_i, and cooling water temperature T_w. Each variable is recorded as a dynamic trend comprising 150 sample points. The goal is to identify operational states using a conceptual clustering approach. This will then be extended to a more complicated case study of a refinery MTBE process in Section 7.4.

The approach basically comprises the following procedures: (1) concept extraction from dynamic trend signals using PCA. (2) identification of operational states using an unsupervised machine learning approach, and (3) application of the inductive machine learning system to develop decision trees and rules for process monitoring.

7.3.1 Concept Extraction from Dynamic Trend Signals

This has been discussed in detail in Chapter 3, so only a brief review is presented here. For a specific set of data, the value of a variable represents a dynamic trend, consisting of tens to hundreds of sampled points. In inductive learning, it is the shape of the trend that matters so for a specific variable, when the trends of all the data sets are considered and processed using PCA, the first two principal components (PCs) can be plotted in a two dimensional plane, as shown in Figure 7.3. Figure 7.3 showing PC-1-TR and PC-2-TR corresponds to the first two PCs of the reaction temperature TR. The data sets are grouped into clusters in this two dimensional plane. This permits a dynamic trend to be abstracted as a concept as typically by PC-1-TR in region A. The following sections will show how this process can be used for conceptual clustering using inductive learning.

7.3.2 Identification of Operational States

The next step is identification of operational states. In this case, this can be done using PCA because there are only eight variables. For more complex processes, more sophisticated approaches need to be used, which will be described later in the case study of MTBE. The first two PCs of the eight variables (TR, Fo, Fw, Fi, Ti, Ci, Twi, L) are plotted in Figure 7.7. The five groups which are identified represent the 85 data cases as five clusters corresponding to five distinct operational modes. Detailed examination of the clusters shows that these groups are reasonable.

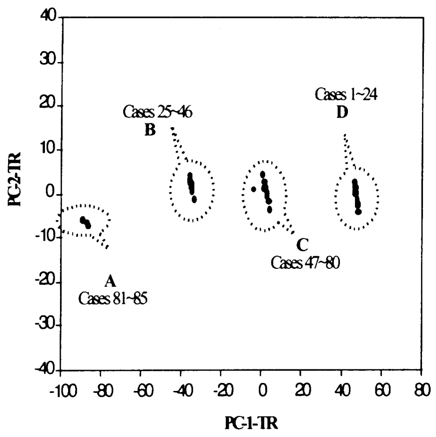


Figure 7.3 PCA two dimensional plot of TR.

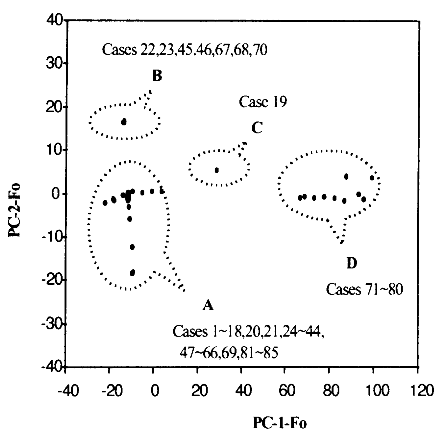


Figure 7.4 PCA two dimensional plot of F_o.

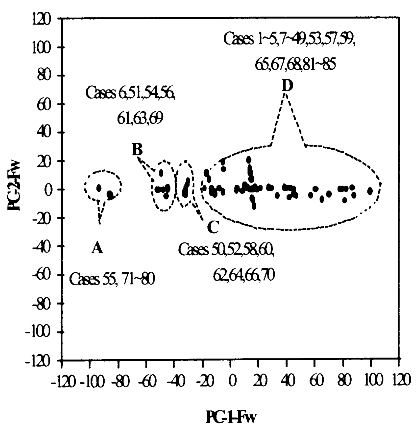


Figure 7.5 PCA two dimensional plot of F_w.

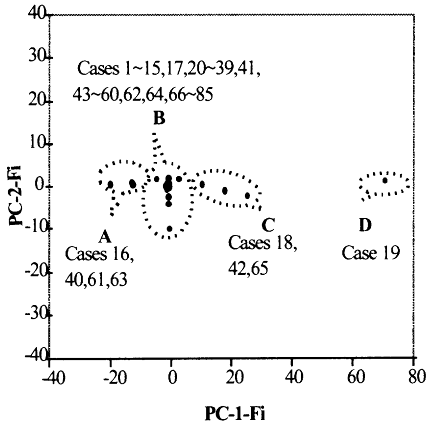


Figure 7.6(a) PCA two dimensional plot of F_i.

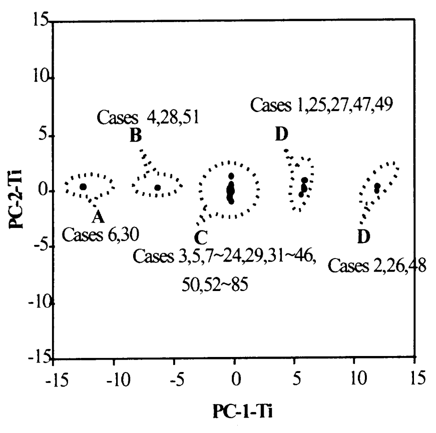


Figure 7.6(b) PCA two dimensional plot of T_i.

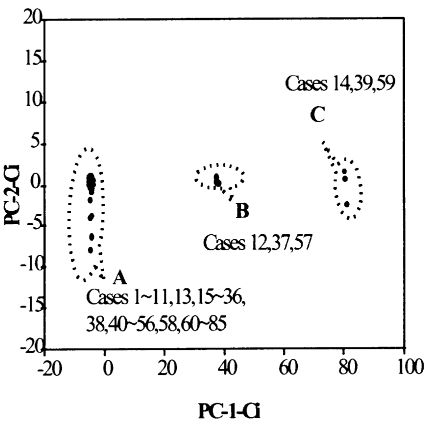


Figure 7.6(c) PCA two dimensional plot of C_i.

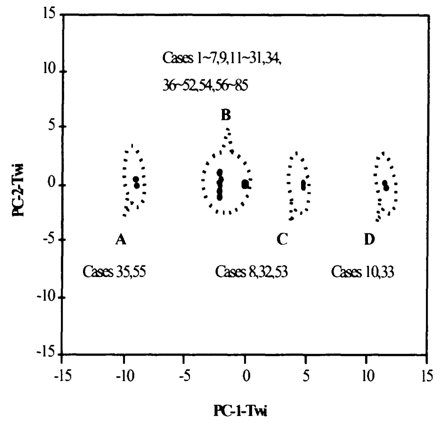


Figure 7.6(d) PCA two dimensional plot of T_{wi}.

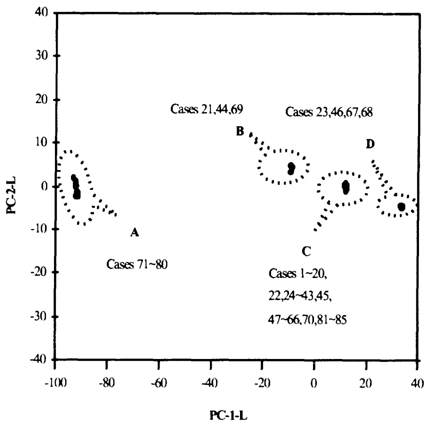


Figure 7.6(e) PCA two dimensional plot of L.

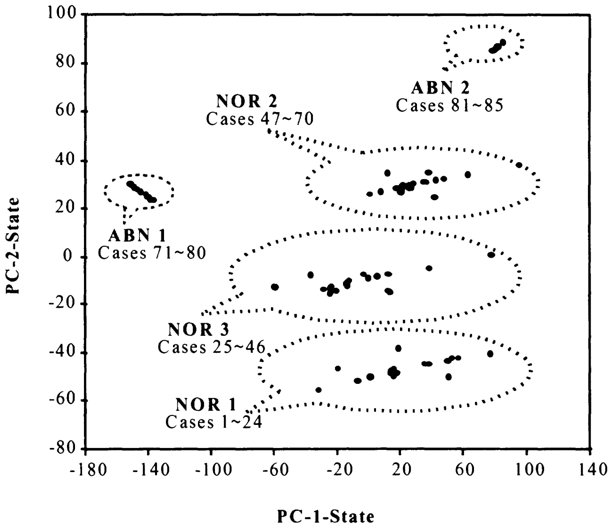


Figure 7.7 PCA two dimensional plot of the CSTR operational states.

7.3.3 Conceptual Clustering

Having characterised the dynamic trend signals and identified the operational states, it is necessary to find out how to generate knowledge which correlates the variables and operational states. To do this requires generating a file as shown in Table 7.2. In fact, each data set in Table 7.2 can be interpreted as a production rule. Thus, the first case is equivalent to the following rule,

```
IF      PC-L = C in Figure 7.6(e)
  AND  PC-TR = D in Figure 7.3
  AND  PC-Fo = A  in Figure 7.4
  AND  PC-Fw = D in Figure 7.5
  AND  PC-Twi = B in Figure 7.6(d)
  AND  PC-Ci = A   in Figure 7.6(c)
  AND  PC-Ti = D   in Figure 7.6(b)
  AND  PC-Fi = B   in Figure 7.6(a)
THEN     States = NOR1 in Figure 7.7
```

Table 7.2 The data structure used by C5.0 for conceptual clustering.

PC_L	PC_TR	PC_Fo	PC_Fw	PC_Twi	PC_Ci	PC_Ti	PC_Fi	States
C	D	A	D	B	A	D	B	NOR1
C	D	A	D	B	A	E	B	NOR1
...	...	...	...	...	...	...	...	...
A	C	D	A	B	A	C	B	ABN1
A	C	D	A	B	A	C	B	ABN1
...	...	...	...	...	...	...	...	...

Variable name,	Value space,	Ref. figure	Variable name,	Value space,	Ref. figure
PC_L,	[A, B, C, D],	Figure 7.6(e)	PC_Ci ,	[A, B, C],	Figure 7.6(c)
PC_Fo,	[A, B, C, D],	Figure 7.4	PC_Ti,	[A, B, C, D, E],	Figure 7.6(b)
PC_Fw,	[A, B, C, D],	Figure 7.5	PC_Fi,	[A, B, C, D],	Figure 7.6(a)
PC_Twi,	[A, B, C, D],	Figure 7.6(d)	States,	[NOR1, NOR2, NOR3, ABN1, ABN2]	Figure 7.7
PC_TR,	[A, B, C, D],	Figure 7.3			

Obviously this is simply an explanation of the database and the decision tree developed will be very complex. C5.0 makes it possible to develop a simpler tree. Simple tree is preferable because it can usually perform better than a complex tree for data cases outside the training data set.

The decision tree developed for the CSTR case study is shown in Figure 7.8 and can be converted to production rules, as shown in Table 7.3. C5.0 identifies the reactor temperature as the root node and states that if TR is in the region of A, B or D of Figure 7.3, then the operation will be in regions ABN2 (abnormal mode 2), NOR2 (Normal operation mode 2), or NOR1 (Normal operation mode 1) of Figure 7.7. If TR is in the region C in Figure 7.3, then there are three possible situations depending on Fo. If Fo is in the region D of Figure 7.4, then the operation will correspond to ABN1 (Abnormal operation 1): if Fo is in A or B of Figure 7.4, then the operation will be NOR3 (Normal operation 3). The result effectively states that it is possible to focus on monitoring TR in Figure 7.3. If TR is in the region C, then Fo in Figure 7.4 should be examined. It also shows the variables responsible for the location placing the operation in a specific region of Figure 7.7.

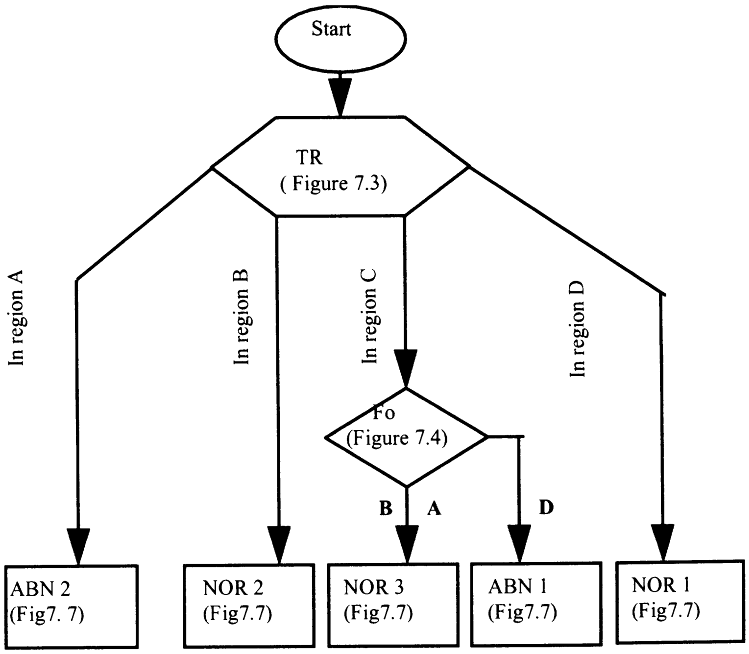


Figure 7.8 The decision tree developed for the CSTR after conceptualization of the variable trends.

Table 7.3 The production rules converted from the decision tree in Figure 7.8.

Rule 1: IF TR = A in Figure 7.3
THEN Operational state = ABN 2 in Figure 7.7

Rule 2: IF TR = B in Figure 7.3
THEN Operational state = NOR 2 in Figure 7.7

Rule 3: IF TR = C in Figure 7.3
AND Fo = A or B in Figure 7.4
THEN Operational state = NOR 3 in Figure 7.7

Rule 4: IF TR = C in Figure 7.3
AND Fo = D in Figure 7.4
THEN Operational state = ABN 1 in Figure 7.7

Rule 5: IF TR = D in Figure 7.3
THEN Operational state = NOR 1 in Figure 7.7

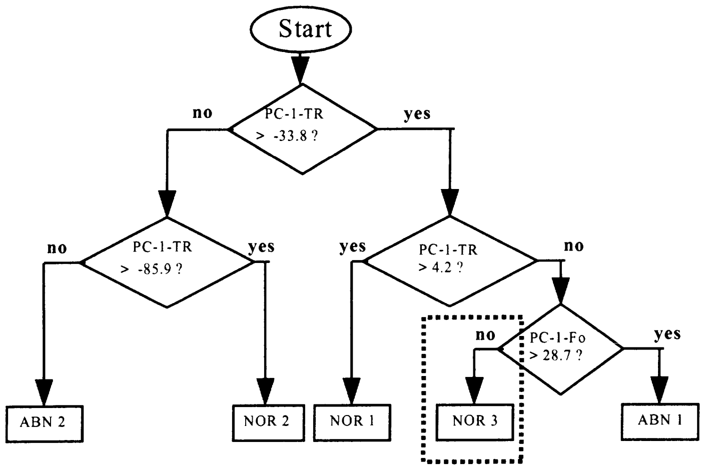


Figure 7.9. The decision tree developed for the CSTR using the eigenvalues of the first two principal components of each variable.

Table 7.4 The numerical values of the first two principal components of each variable are used by C5.0 to develop the tree in Figure 7.9.

Fi		Ti		Ci		Twi		Fw		Fo		TR		L		State
PC-1	PC-2	PC-1	PC-2	PC-1	PC-2	PC-1	PC-2	PC-1	PC-2	PC-1	PC-2	PC-1	PC-2	PC-1	PC-2	
-1.07	0.47	5.66	-0.45	-4.47	0.19	-0.10	-0.08	-41.9	-5.57	-11.5	-0.92	47.83	-2.48	11.9	0.04	NOR 1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
-0.92	0.47	-0.29	0.01	-4.72	0.28	-0.11	-0.06	86.7	0.64	-11.7	-0.98	-88.9	-6.19	11.96	0.30	ABN 2

Table 7.5 The production rules converted from the decision tree of Figure 7.9.

Rule 1: IF PC-1-TR =< -33.8
AND PC-1-TR =< -85.9
THENOperational state = ABN 2
Rule 2: IF PC-1-TR =< -33.8
AND PC-1-TR > -85.9
THEN Operational state = NOR 2
Rule 3: IF PC-1-TR > 4.2
THEN Operational state = NOR 1
Rule 4: IF PC-1-TR =< 4.2
AND PC-1-Fo =< 28.7
THEN Operational state = NOR 3
Rule 5: IFPC-1-TR > 4.2
AND PC-1-Fo > 28.7
THEN Operational state = ABN 1

The decision tree shown in Figure 7.8 and the rules in Table 7.3 provide guidance for operation clustering which is transparent.

In the above discussion, the dynamic trends in the two-dimensional PCA planes have been abstracted. An alternative way is to use the numerical values of the first two PCs directly. The data structure to be processed by C5.0 is then put in the format of Table 7.4. A decision tree thus developed is shown in Figure 7.9 and the rules shown in Table 7.5. Inspection of Figure 7.9 reveals that it is very similar to the tree in Figure 7.8. For example, the conditions leading to ABN2 in Figure 7.9 are

```
IF      PC-1-TR < -33.8
  AND  PC-1-TR < -85.9
THEN  ABN2
```

From Figure 7.3 it can be seen that the two preconditions of the rule are equivalent to IF TR = region A and so gives the same result as Figure 7.8. The only difference is the last branch leading to NOR3 (the dashed line box of Figure 7.9). In Figure 7.9 the rule is,

```
IF      PC-1-TR > -33.8
  AND PC-1-TR < 4.2
  AND PC-1-Fo < 28.7
THEN  NOR3
```

By making reference to Figures 7.3 and 7.4, the rule condition is equivalent to IF TR = C in Figure 7.3 and Fo = A or B or C in Figure 7.4. But in Figure 7.8, the corresponding rule condition is TR = C in Figure 7.3 and Fo = A or B. This slight difference does not say which approach is better. However, in the next section involving a larger case study, conceptualisation of dynamic trends first gives better results.

7.4 Application to the Refinery MTBE Process

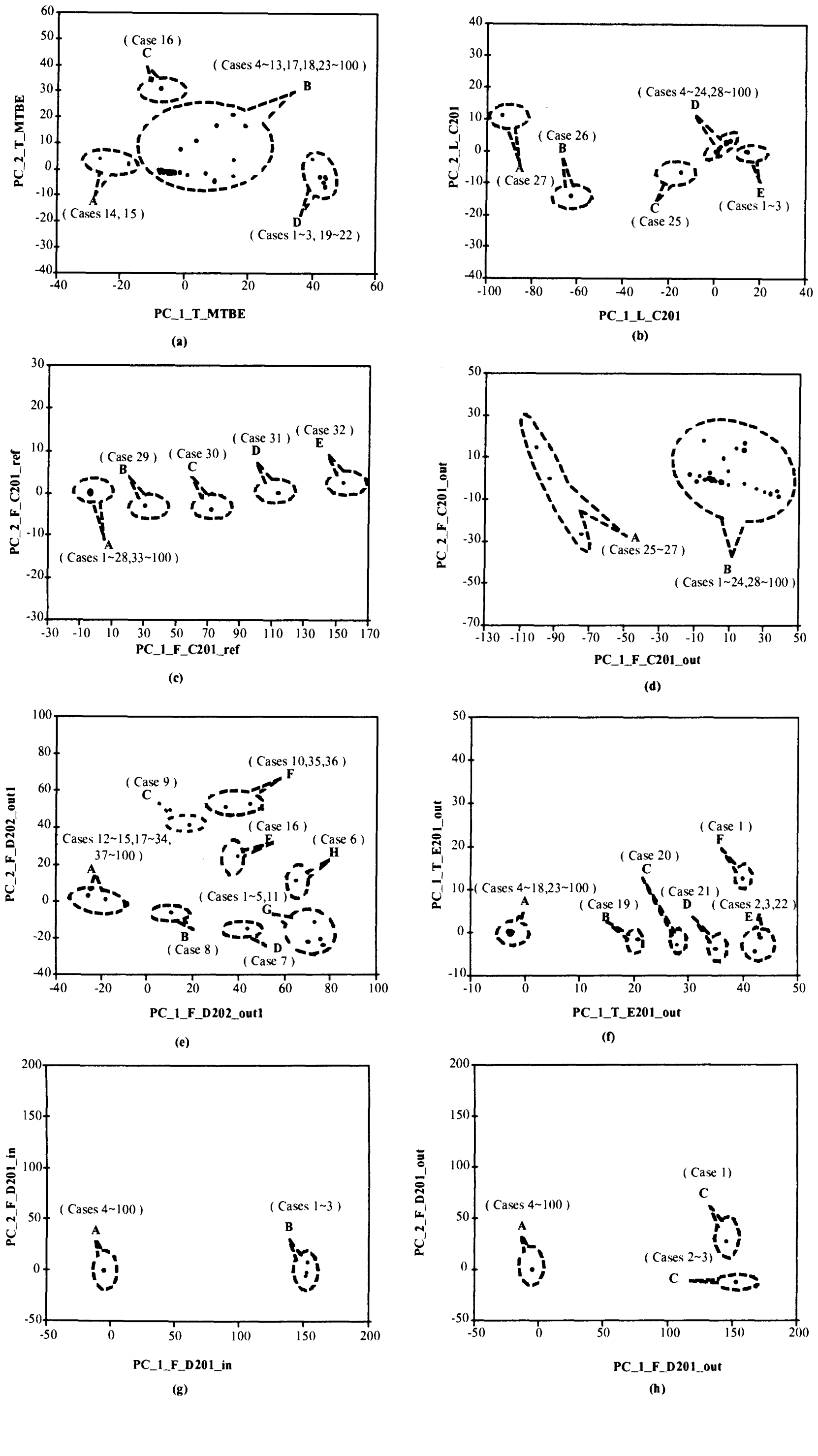
The above description of conceptual clustering is based on a CSTR. Here we apply the approach to a more complicated case study, the refinery methyl tertiary butyl ether (MTBE), which is described in more detail in Appendix C. A database of 100 sets is obtained using a process simulator and the output is summarised in Table C1 of Appendix C. Thirty six data sets correspond to various operations which are regarded as abnormal or subject to significant disturbances. The rest are considered as operating normally. The study is restricted to 100 data sets since increasing the size of the data set by including more normal operational data does not make difference to the result and it makes it easy to understand the analysis and presentation of the result. Each data set consists of twenty one variables, which are listed in appendix C. For each variable, a dynamic trend consisting of 256 points is used to record the response to changes. Therefore the size of the data to be analysed is 100x21x256.

Table 7.6 The clusters of operational states using ART2.

Clusters	Cluster Name	Cases	Clusters	Cluster Name	Cases
1	ABN1	1, 2, 3	7	ABN7	14, 15
2	ABN2	4, 5, 7, 8, 11	8	ABN8	16
3	ABN3	6	9	ABN9	19, 20, 21, 22
4	ABN4	9	10	ABN10	23, 24, 25,2 6, 27
5	ABN5	10, 35, 36	11	ABN11	29, 30, 31, 32
6	Normal	12, 13, 17, 18	12	Normal	28, 33, 34, 37-100

7.4.1 Concept Formation from Dynamic Trends Using PCA

The projections of the dynamic trends of some variables onto the PCA plane are shown in Figures 7.10 (a) to (l). Only those variables that appear later in the decision trees are shown in Figure 7.10. In Figure 7.10(a), the regions A, C and D are clearly distinguished. However, region B is fuzzy. This simply means that the cases in region B (4-13, 17, 18 and 23-100) can not be distinguished in Figure 7.10(a). Their differences can only be identified by reference to other variables. A similar explanation applies to Figure 7.10(j) which classifies the responses of the variable F_D202_in1 into two classes for all data sets. This requires other variables to discriminate the data sets. When all the variables are considered together, some variables might be more important than others to the classification. For most of the twenty one variables, the grouping based on visual examination of the two dimensional PCA plane is straightforward. Even for a few variables, the groupings may not be very clear, it does not affect the final result significantly. When grouping for one variable is not clear, other variables will play more important roles in the operational state clustering.


PC_2_T_MTBE
PC_1_T_MTBE
(Case 16) C
(Cases 4~13,17,18,23~100) B
A (Cases 14, 15)
D (Cases 1~3, 19~22)
(a)
PC_2_L_C201
PC_1_L_C201
(Cases 4~24,28~100) D
(Case 26) B
A (Case 27)
C (Case 25)
E (Cases 1~3)
(b)
PC_2_F_C201_ref
PC_1_F_C201_ref
(Case 29) B
(Case 30) C
(Case 31) D
(Case 32) E
A (Cases 1~28,33~100)
(c)
PC_2_F_C201_out
PC_1_F_C201_out
A (Cases 25~27)
B (Cases 1~24,28~100)
(d)
PC_2_F_D202_out1
PC_1_F_D202_out1
(Cases 10,35,36) F
(Case 9) C
Cases 12~15,17~34, 37~100) A
(Case 16) E
(Case 6) H
(Cases 1~5,11) G
B (Case 8)
D (Case 7)
(e)
PC_1_T_E201_out
PC_1_T_E201_out
(Case 1) F
(Case 20) C
(Cases 4~18,23~100) A
(Case 21) D
(Case 19) B
(Cases 2,3,22) E
(f)
PC_2_F_D201_in
PC_1_F_D201_in
(Cases 4~100) A
(Cases 1~3) B
(g)
PC_2_F_D201_out
PC_1_F_D201_out
(Cases 4~100) A
(Case 1) C
(Cases 2~3) C
(h)

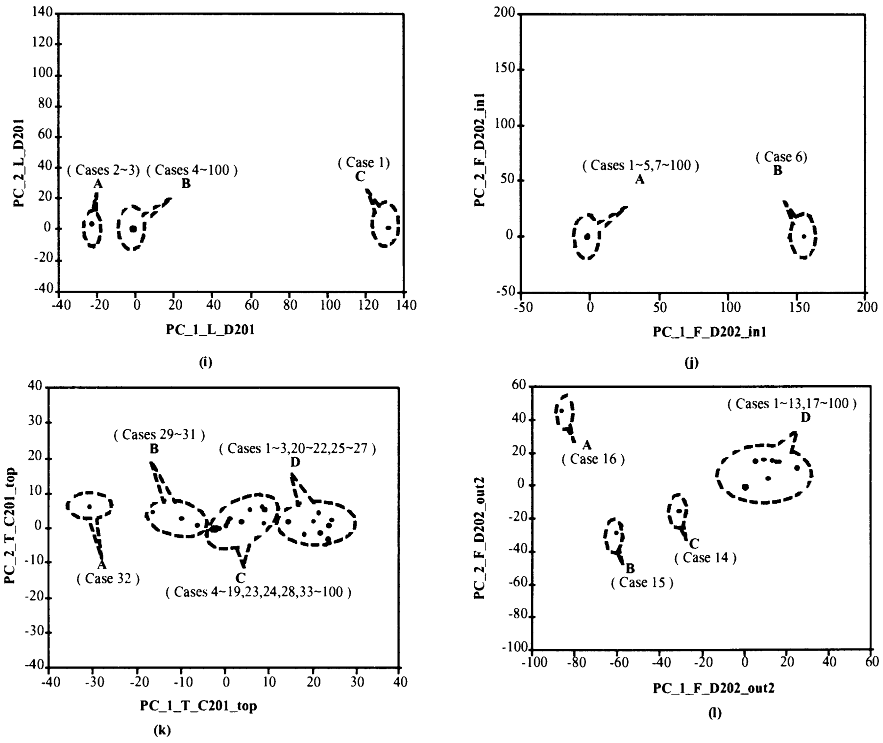


Figure 7.10 (Including preceding page) PCA two dimensional plots of variables for MTBE.

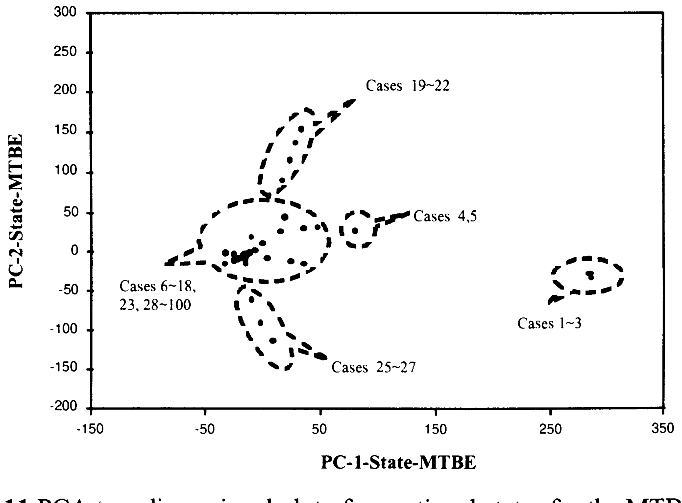


Figure 7.11 PCA two dimensional plot of operational states for the MTBE process.

7.4.2 Identification of Operational States

The CSTR case study showed that PCA can classify the operational states satisfactorily. Here, both PCA and the adaptive resonance theory (ART2, introduced in Chapter 6) are used. The PCA analysis shown in Figure 7.11 results in five clusters and the ART2 result is shown in Table 7.6 which predicts twelve clusters. Both results are reasonable, but ART2 gives a more detailed picture and appears to be more accurate. Sammon [123] indicated that since PCA is a linear method, it may not give an adequate representation in two or three dimensions when the original number of attributes is large, visual examination may not be possible. He also gave an example where data generated giving five groups in four dimensions are projected into the space of the two principal eigenvectors. Visual examination of this projection shows only four groups, since two of the clusters overlap completely in the two dimensional space. In analysis of process operational data, similar observations have been made by other researchers [122, 119, 201]. In the following discussion, only the ART2 clustering result will be used.

It is apparent that clusters with only one data set are correct. These are clusters three, four and eight. Data sets 1, 2 and 3 in cluster one all cause the flowrate of the C4 hydrocarbons feed to fall to zero and so should be in the same class. Data sets in cluster two comprising 4, 5, 7, 8 and 11 cause the methanol flow to the mixer M201 to be either completely cut off or greatly reduced to cause them to be in the same class. Data sets in class five, which include cases 10, 35 and 36 corresponding to changes of the output of the controller FC202D from 33% to 59%, 33% to 50% and 33% to 55% respectively. Cluster six has cases 12, 13, 17 and 18 representing reduction or cut in methanol flow to the tank D211 and column C201. The two cases in class seven, 14 and 15 represent the changes in the opening of the valve HC211D from 18% to 40% and 60% respectively. Cases 19 to 22 in cluster nine refer to changes of the output of the controller TC201R from 39% to 30%, 20%, 10% and 0%. 23 to 27 are cases corresponding to changes in the output of the controller FC203E from 37% to 33%, 29%, 16%, 6% and 0%, and are classified as cluster ten. Cluster eleven has four cases, 29-32 corresponding to changes of the output of the controller FC201D from 40% to 30%, 20%, 10% and 0%. The last cluster, cluster twelve, has the normal operational data sets 37 to 100. The assignment of cases 28, 33 and 34 to this cluster is not apparent but is nevertheless not unreasonable given that they represent insignificant changes.

7.4.3 Decision Tree Development

Conceptual clustering not only predicts the operational states but also interprets the prediction using causal knowledge in terms of decision trees or production rules. Here, a variable takes discrete values from a region of the two dimensional PCA plane of the variable. For example, the liquid level at the bottom of column C201, L_C201, takes values from its PCA plane in Figure 7.10(b) including A, B, C, D, and E. For the data case number 24, L_C201 takes the value of D. For each data set, the operational state takes values from Table 7.6. For example, data set 24 has the value of ABN10.

The decision tree developed is shown in Figure 7.12. The decision tree can be easily translated into rules. For example, the rules that lead to ABN4 and ABN11 are,

Rule7:	IF	T_MTBE = B in Figure 7.10(a)
	AND	F_D202_out1 = C in Figure 7.10(e)
	THEN	Operational state = ABN4
Rule 11:	IF	T_MTBE = B in Figure 7.10(a)
	AND	F_D202_out1 = A in Figure 7.10(e)
	AND	T_C201_top = A or B in Figure 7.10(k)
	THEN	Operational state = ABN11

The root node is T_MTBE, the bottom temperature of the reactive distillation column C201. It indicates that it is the most important variable that distinguishes operational modes representing the one hundred data sets. Detailed examination of the decision tree and all the dynamic responses in conjunction with the MTBE process flowsheet revealed that the tree is reasonably good. An example illustrating this is the rules leading to ABN11. From Table 7.6, it is known that ABN11 covers data cases 29, 30, 31 and 32, corresponding to changes of the output of the controller FC201C (reflux flowrate control) in manual mode from 40% to 30%, 40% to 20%, 40% to 10% and 40% to 0%. Clearly the most important variable which discriminates between these data sets from others is the column top temperature T_C201_top. This is confirmed by Figure 7.12, in which the nearest node to ABN11 is T_C201_top.

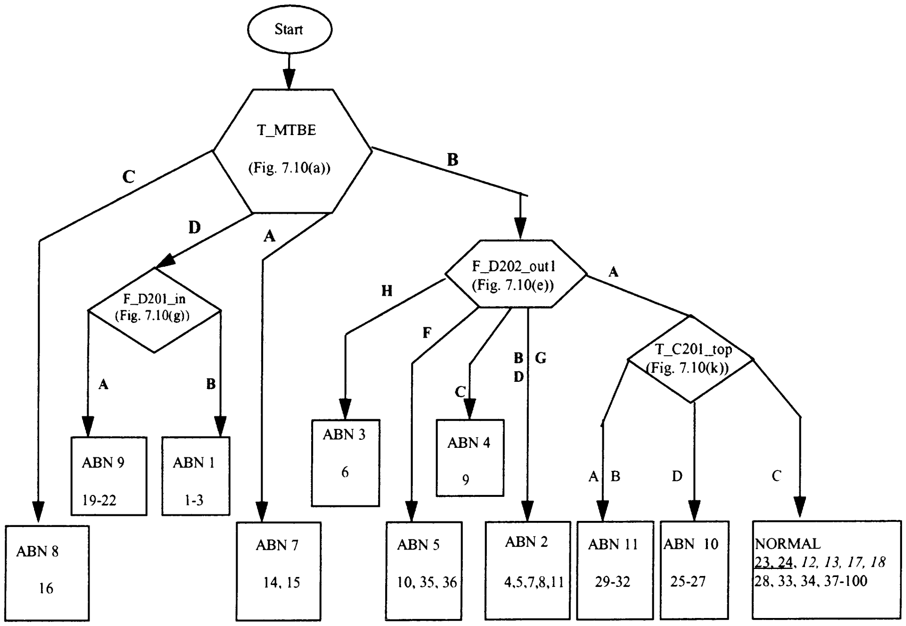


Figure 7.12 Decision tree developed for the MTBE process.

In Figure 7.12, the numbers at the bottom of the nodes indicates the data sets. For instance, the node ABN8 has only one data set, namely 16. Comparing Figure 7.12 and Table 7.6, it is found that the decision tree in Figure 7.12 gives correct predictions except for the nodes ABN10 and NORMAL in Figure 7.12.

Data sets 23 and 24 (underlined in Figure 7.12) are assigned to ABN10 by ART2, as shown in Table 7.6, but to the node NORMAL in Figure 7.12 by C5.0. Data sets 23 and 24 represent the cases where the output of the steam flowrate controller FC203E at the bottom of the reactive distillation column C201 is changed from 37% to 33% and 37% to 29%. In fact these are insignificant changes so assigning them to the NORMAL operational state is acceptable. This inconsistency with ART2 can be attributed to two factors. Firstly ART2 is based on numerical values which is more precise than the conceptual clustering using C5.0. Secondly, the conceptualisation of variables by visual examination of the PCA two dimensional plane may give rise to some inaccuracies.

Data sets 12, 13, 17 and 18 (in italics in Figure 7.12) are clustered in a separate class by ART2, as shown in Table 7.6. They are mixed with other cases in the node labelled NORMAL in Figure 7.12. Referring to Table C1 and Figure C1, all the four cases cause flowrate changes on the methanol stream to tank D211 and then to the column C201. In reality, this flowrate is very small compared to the total methanol and C4 hydrocarbons flows to the mixer M201, (about 1/12 or 1/100 respectively). As a result, the changes of the methanol flow to D211 are insignificant. Consequently it is reasonable to regard cases 12, 13, 17 and 18 as being in the NORMAL operation class.

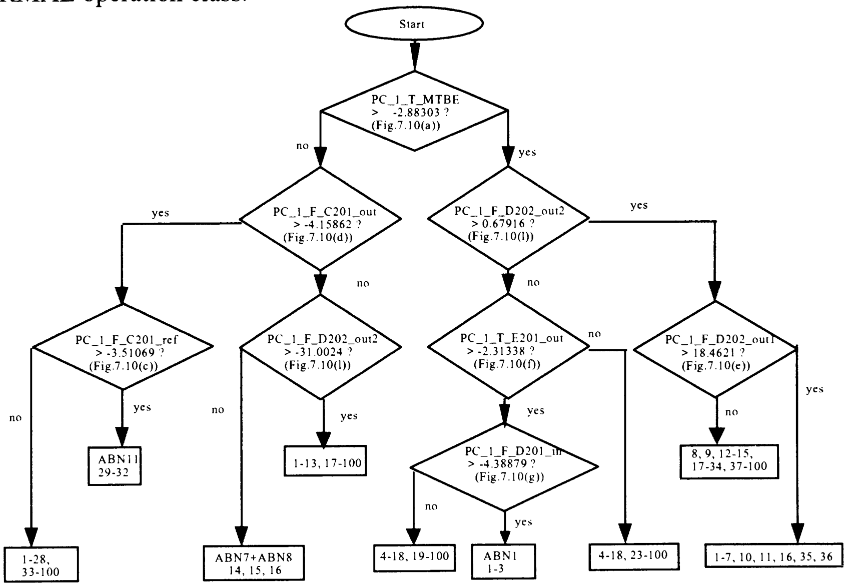

Start
PC_1_T_MTBE > -2.88303 ? (Fig.7.10(a))
no
yes
PC_1_F_C201_out > -4.15862 ? (Fig.7.10(d))
PC_1_F_D202_out2 > 0.67916 ? (Fig.7.10(l))
PC_1_F_C201_ref > -3.51069 ? (Fig.7.10(c))
PC_1_F_D202_out2 > -31.0024 ? (Fig.7.10(l))
PC_1_T_E201_out > -2.31338 ? (Fig.7.10(f))
PC_1_F_D202_out1 > 18.4621 ? (Fig.7.10(e))
PC_1_F_D201_in > -4.38879 ? (Fig.7.10(g))
ABN11 29-32
1-13, 17-100
8, 9, 12-15, 17-34, 37-100
1-28, 33-100
ABN7+ABN8 14, 15, 16
4-18, 19-100
ABN1 1-3
4-18, 23-100
1-7, 10, 11, 16, 35, 36

Figure 7.13 Decision tree developed if variables are not conceptualised.

In the above discussion, the dynamic trend signals of a variable are converted to qualitative concepts in a PCA two dimensional plane and then the inductive learning approach is used. An alternative way is using the eigenvalues of the first two PCs of a variable directly. The resulting decision tree derived in this way is shown in Figure 7.13. The tree is found to be completely unreasonable because most clusters are overlapped therefore it cannot be used for predictions. Figure 7.14 shows the discretisation of two variables T_MTBE and F_D220_out2. The dashed lines are the boundary values for the discretisatiom. Since for each variable the discretisation is always binary, this is obviously not satisfactory. For example, Figure 7.10(i) shows that the variables L_D201 can clearly take three values. A two-valued discretisation is not able to capture all the features which causes the inaccuracy of the tree.

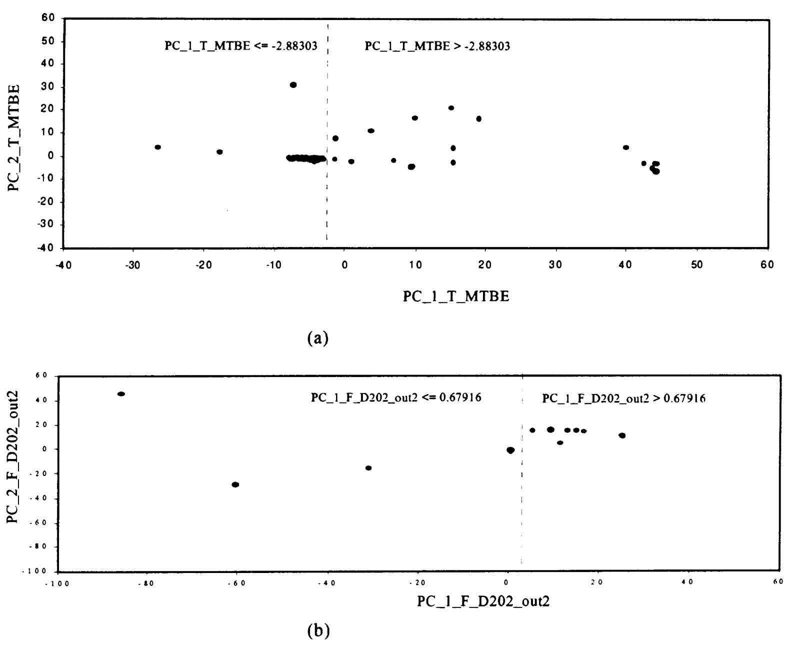


Figure 7.14 Binary discretisation of variables in C5.0.

7.5 General Review

Inductive learning has been introduced as a method for analysis of data records averaged over days or weeks and a conceptual clustering tool for developing on-line operational monitoring systems. It can learn from a large number of examples to develop explicit and transparent knowledge in the form of decision trees and production rules. It is also able to identify the most important variables that contribute to clustering, which is clearly valuable for analysing process operational data and process monitoring. There are several issues that need to be addressed. Like most inductive learning systems, C5.0 is not recursive, it means it can only deals with data as a batch, not be able to learn as an example is presented. In addition though PCA has proved to be an effective way of concept extraction from dynamic trend signals, it is expected that the combination of PCA and Wavelet will deliver more effective pre-processing methods. Compared with similarity or distance based methods which have been widely studied, conceptual clustering clearly needs more research attention.

CHAPTER 8
AUTOMATIC EXTRACTION OF KNOWLEDGE RULES FROM PROCESS OPERATIONAL DATA

An alternative technique for developing intelligent process monitoring and control systems is expert systems (ESs). ESs use logic rules to carry out heuristic reasoning. Knowledge used to reach a conclusion is transparent because ESs have explanation capabilities. In this respect, ESs are superior to neural networks because the knowledge embedded in neural networks is opaque. It has long been recognised that a critical issue in developing knowledge based ESs is the bottleneck of knowledge acquisition. Traditionally knowledge acquisition has been dependent on consulting domain experts. A disadvantage of this approach is that experts are usually better in collecting and archiving cases than in expressing the experience and cases explicitly into production rules. There are more difficulties in compiling knowledge about process operations due to the large number of interacting variables.

A promising approach for acquiring heuristic knowledge rules is to extract knowledge automatically from data. In this chapter we describe methods for this purpose which include fuzzy sets, neural networks, rough sets and fuzzy neural networks.

8.1 Rules Generation Using Fuzzy Set Operation

Generating fuzzy rules based on fuzzy set method developed by Wang and Mendel [205-208] is concerned with the following context. Suppose we are given a set of input-output data pairs,

$$(x_1^{(1)}, x_2^{(1)}, \ldots, x_N^{(1)})\,(y_1^{(1)}, y_2^{(1)}, \ldots, y_T^{(1)})$$
$$(x_1^{(2)}, x_2^{(2)}, \ldots, x_N^{(2)})\,(y_1^{(2)}, y_2^{(2)}, \ldots, y_T^{(2)})$$
$$\cdots \quad \cdots$$
$$(x_1^{(M)}, x_2^{(M)}, \ldots, x_N^{(M)})\,(y_1^{(M)}, y_2^{(M)}, \ldots, y_T^{(M)}) \tag{8.1}$$

where x_1, , x_2 , ...x_N are inputs and y_1, y_2, ...y_T are outputs. M is the total number of given data patterns. The procedure for generating fuzzy rules is divided into five steps which are explained based on a problem of two inputs and one output which is a special case of Expression 8.1.

$$(x_1^{(1)}, x_2^{(1)})\,(y^{(1)})$$
$$(x_1^{(2)}, x_2^{(2)})\,(y^{(2)})$$
$$\cdots \quad \cdots$$
$$(x_1^{(M)}, x_2^{(M)})\,(y^{(M)}) \tag{8.2}$$

Step 1: Divide the input and output variables into fuzzy regions of fuzzy concepts. Each region of a fuzzy concept is represented by a membership function. Figure 8.1(a) shows that the input variable x_1 is divided into five regions of fuzzy concepts, i.e., CE (Centre), B1 (Big 1), B2 (Big 2), S1 (Small 1) and S2 (Small 2). The shape of the membership function, m(x_1) is triangular. x_2 is divided into seven regions and y into five regions.

Step 2: Generate fuzzy rules from given data pairs. For given $x_1^{(i)}$, $x_2^{(i)}$and $y^{(i)}$ find corresponding membership values. For example, in Figure 8.1(a), since $m(x_1^{(1)})$ in B1 = 0.8, $m(x_1^{(1)})$ in B2 = 0.2 and 0.8 > 0.2, we take $m(x_1^{(1)})$ = 0.8 and consider $x_1^{(1)}$ belonging to B1. Following the same consideration, from Figure 8.1(b), we have $x_2^{(1)}$ = 0.7 in S1. Then we can obtain one rule from one pair of desired input-output data, e.g.,

$$\because\ (x_1^{(1)}, x_2^{(1)})\,(y^{(1)})$$
$$\Rightarrow\ [x_1^{(1)}(0.8 \text{ in B1}),\ x_2^{(1)}(0.7 \text{ in S1})]\ [y^{(1)}\ (0.9 \text{ in CE})]$$

∴ we generate rule 1 as

IF x_1 is B1 and x_2 is S1, THEN y is CE.

Similarly,

$$\because\ (x_1^{(2)}, x_2^{(2)})\,(y^{(2)})$$
$$\Rightarrow\ [x_1^{(2)}(0.6 \text{ in B1}),\ x_2^{(2)}(1 \text{ in CE}]\ [y^{(2)}\ (0.7 \text{ in B1})]$$

∴ we generate rule 2 as

IF x_1 is B1 and x_2 is CE, THEN y is B1.

The rules generated in this way are only "and" rules. However, if two rules generated have the same THEN part then they represent "or" relation.

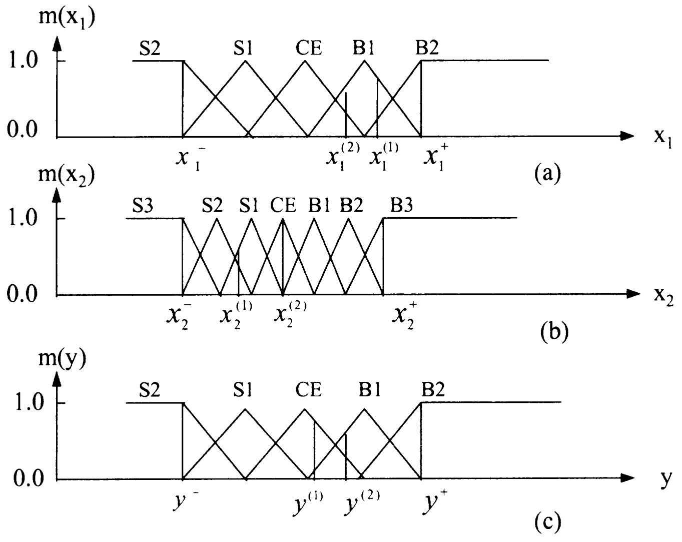


Figure 8.1 Divisions of the input and output spaces into fuzzy regions and the corresponding membership functions. (a) *m(x₁)*, (b) *m(x₂)*, (c) *m(y)*.

Step 3: Assign a degree to each rule. Since there are normally lots of data pairs and each data pair generates one rule, it is highly probable that there are conflicting rules, i.e., rules have the same IF part but a different THEN part. This conflict is solved by assigning a degree to each rule and accept only the rule from a conflict group that has the maximum. The equation to calculate the degree for a rule is, for example rule 1,

$$\begin{aligned}\text{Degree of Rule 1} &= m_{B1}(x_1)m_{S1}(x_2)m_{CE}(y)m^{(1)}\\ &= 0.8*0.7*0.9*m^{(1)}\\ &=0.504*m^{(1)}\end{aligned} \tag{8.3}$$

Where $m^{(1)}$ is a degree given by human expert about the importance and trustiness of the data pair.

Step 4: Create the combined fuzzy rule base. The combined fuzzy rule base is created by filling the box of Figure 8.2 with the rule produced in the above steps. If there is more than one in one box, use the rule with maximum degree.

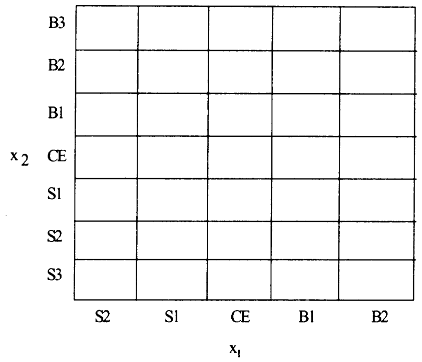


Figure 8.2 The form of a fuzzy rule base.

The approach uses algorithms of fuzzy set and is simple and straightforward. The rules generated by this method may be repeating and the set of rules may be large [209]. Srinivasan et al. [210] improved this method in this aspect using an inductive learning to determine the fuzzy rules.

Other methods of generating fuzzy rules from numerical data using fuzzy sets have slightly different procedures and are not discussed here [211-216].

8.2 Rules Generation from Neural Networks

The rules generation procedure from neural networks by Fu [217] can be illustrated by reference to Figure 8.3.

A rule generated has the form of

IF the premise, THEN the action (conclusion).

Specifically for the case in Figure 8.3,

$$\text{IF } A_1^+,\ \ldots A_i^+,\ \ldots\ \neg A_1^-,\ \ldots \neg A_j^-,\ \ldots,\ \text{ THEN C (or } \neg\text{C)}$$

where A_i^+ is a positive antecedent (an attribute in the positive form), $\neg A_j^-$ is a negative antecedent (an attribute in the negative form), C the concept (conclusion), and $\neg$ reads "not". Each node in the hidden or output layer is designated by a symbol which represents a concept to be confirmed or disconfirmed, such as node C in Figure 8.3. Confirmation (or disconfirmation) of a node concept is measured by the activation of the node which is calculated by

$$o_j = F(\sum_i w_{ji} x_i - \theta_j) \tag{8.4}$$

where,

o_j - the activation of node j (a hidden or output node);

w_{ji} - the weight on the connection from unit i to unit j;

o_j - the threshold on unit j, which is adjustable;

F - the activation function which represents the nonlinearity on the hidden or output nodes and is defined as the sigmoid form.

$$F(a) = \frac{1}{1 + e^{-\lambda a}} \tag{8.5}$$

where λ determines the steepness of the function.

In the case of a hard-limiting activation function, the output of a node is given by

$$o_j = \begin{cases} 1 & \text{if } \sum_i w_{ji} x_i > \theta_j \quad \text{(activation > threshold)} \\ 0 & \text{else} \end{cases} \tag{8.6}$$

The method needs to train the neural network first and then based on the above concepts a detailed procedure is developed to extract rules. Although it shows advantages compared with some other methods of extracting rules (e.g., Gallant [218]) from a trained neural network, the method still has its weakness: it produces rules with concepts that have no physical meaning since it uses hidden neurones.

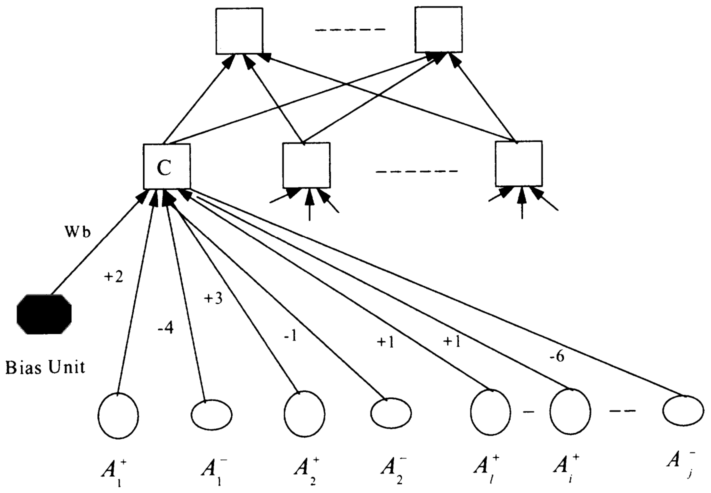


Figure 8.3 The post-attrs and neg-atts relative to a concept. An attribute can be either a post-att or a neg-att, depending on the concept.

8.3 Rules Generation Using Rough Set Method

Learning production rules by rough set approach has been studied by Chan [219], Srinivasan and Chan [210], Chmielewski et al. [220], Quafafou and Chan [209] and Ziarko [226]. The approach makes use of the rough set theory introduced by Pawlak [221, 222, 223]. Let *U* be a nonempty set called universe, and *R* be an equivalence relation on *U* called indiscernibility relation. An ordered pair *A* = (*U*, *R*) is called an approximation space. For each subset in *U*, *X* is characterised by a pair of sets, the lower and upper approximation of *X* in *A*. The lower approximation of *X* in *A* is defined as

$$\underline{R}X = \{x \in U | [x]_R \subseteq X\} \text{ and}$$

the upper approximation of *X* in *A* is defined as

$$\overline{R}X = \{x \in U | [x]_R \cap X \neq \varnothing\}$$

in which $[x]_R$ denotes the equivalence class of *R* containing *x*. A subset *X* of *U* is said definable in *A* iff (if and only if) $\underline{R}X = \overline{R}X$.

The rules generating procedure based on the above concepts is better to be illustrated by reference to an example as shown in Table 8.1.

The universe *U* is the set of example objects,

U= {e1, e2, ... e8}

and each object is characterised by the attributes

Density, Colour, Boiling range and Class.

The attributes are classified as condition attributes and decision attributes. (The production rule for a single example object is IF condition attributes THEN decision attributes. For e1 as an example, IF Density = Low, Colour = Light and Boiling range = Narrow, THEN Class = 1). The automatic rules generating method is dependent on the calculation of the lower and upper approximations of the class 1 and 2 (denoted as *X1* and *X2*) as well as the definability.

The concept "class = 1" is the subset *X1* = {e1, e2, e4, e5, e7} and the concept "class = 2" is the subset *X2* = {e3, e6, e8}.

These define the relations on *U*.

The relations on *U* can also be defined on the basis of single attributes, then we have the following attributes on *U* induced by those equivalence relations on *U*.

{Density}* = {{e1, e5, e8}, {e2, e3, e4, e6, e7}},

{Colour}* = {{e1, e3, e7, e8}, {e2, e4, e5}, {e6}},

{Boiling range}*={{e1, e2, e7}, {e3, e4, e5, e6, e8}}, and

{Class}*= {{e1, e2, e4, e5, e7}, {e3, e6, e8}}

The above partitions will also be called classification of *U* generated by single attribute sets.

Table 8.1 A training set for the rough set approach represented by a decision table.

Examples	Attributes			
	Density	Colour	Boiling range	Class
e1	Low	Light	Narrow	1
e2	High	Very light	Narrow	1
e3	High	Light	Wide	2
e4	High	Very light	Wide	1
e5	Low	Very light	Wide	1
e6	High	Dark	Wide	2
e7	High	Light	Narrow	1
e8	Low	Light	Wide	2

The lower approximations of the class *X1* by each condition attribute are:

$\{\underline{\text{Density}}\}X1 = \emptyset$

$\{\underline{\text{Colour}}\}X1 = \{e2, e4, e5\}$, and

$\{\underline{\text{Boiling range}}\}X1 = \{e1, e2, e7\} = U$

The upper approximation of the class X1 by each condition attribute are

$\{\overline{\text{Density}})X1 = \{e1, e2, e3, e4, e5, e6, e7, e8\} = U$,

$\{\overline{\text{Colour}})X1 = \{e1, e2, e3, e4, e5, e7, e8\}$, and

$\{\overline{\text{Boiling range}})X1 = \{e1, e2, e3, e4, e5, e6, e7, e8\} = U$.

Given a decision table *S*, the concepts to be learned are subsets of the universe *U* in *S*. One question is how do we know that a concept *X* can be described by condition attributes in *S*. This is solved by the introduction of definability of sets. Let *P* be a nonempty subset of the set of condition attributes in *S*, then a subset *X* of *U* is said to be *P*-definable in *S* iff $\underline{P}X = \overline{P}X$, and *X* is *P*-undefinable in *S* if $\underline{P}X \neq \overline{P}X$.

Still consider the decision table in Table 8.1. Let *P* be the set {Density, Colour} of attributes, then the classification of *U* generated by *P* is

$P^* = \{\text{Density, Colour}\}^* = \{\text{Density}\}^*.\{\text{Colour}\}^*$

$= \{\{e1, e8\}, \{e3, e7\}, \{e5\}, \{e2, e4\}, \{e6\}\}$.

The lower approximation of *X1* by *P* is

$\underline{P}X1 = \{e1, e4, e5\}$, and

the upper approximation of *X1* by *P* is

$\overline{P}X1 = \{e1, e2, e3, e4, e5, e7, e8\}$.

Therefore, the concept *X1* is P-undefinable in *S* because $\underline{P}X1 \neq \overline{P}X1$.

Let $R = \{$Density, Colour, Boiling range$\}$, then,

$R^* = \{\{e1\}, \{e2\}, \{e3\}, \{e4\}, \{e5\}, \{e6\}, \{e7\}, \{e8\}\}$, and

$\underline{R}X1 = \{e1, e2, e4, e5, e7\} = X1 = \overline{R}X1$.

Thus, the concept *X1* is R-definable in *S*.

Using the concepts of lower and upper approximations and definability of sets, programs LEM2 [219] and LERS [220] have been developed for extracting production rules from data which are able to deal with noise.

8.4 A Fuzzy Neural Network Method for Rules Extraction

We have developed a fuzzy neural network method for generating production rules from data [174]. A conventional NN has real number inputs and weights. There are three main types of fuzzy neural networks (fuzzy NNs) [156]: fuzzy NNs with fuzzy input signals but real number weights, fuzzy NNs with real number input signals but fuzzy weights and fuzzy NNs with both fuzzy input signals and fuzzy weights. The first type is used here. The method involves the following steps:

(1) The input and output variables are divided into fuzzy regions. Figure 8.4 shows examples of such regions, such as (Normal, High and Low) or extended to five values (Normal, Medium High, High, Medium Low and Low).

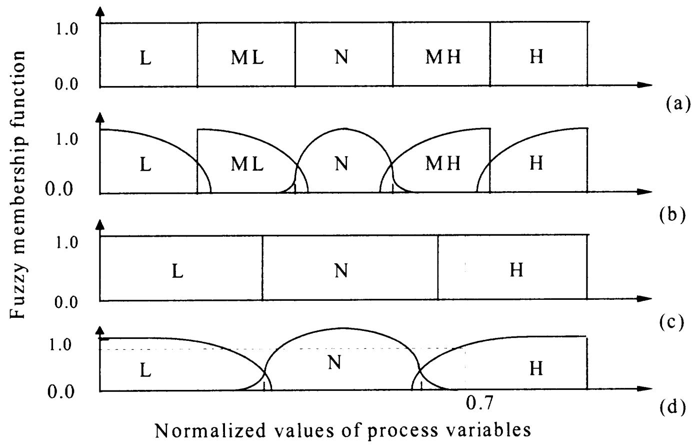
(2) All input and output data patterns are processed using the membership functions in step (1). For example, value of 600 ^{0}C for a temperature may be regarded as High ($\mu_H = 0.7$) when expressed as a fuzzy variable.

(3) A fuzzy NN is constructed. To illustrate the fuzzy NN structure, consider a fuzzy NN with two input variables (T_{C1} and F_{C1}) and a single output variable (T_{H11}), as shown in Figure 8.5. The box contains a normal NN which has one input and one output layer together with a simple hidden layer. The activation functions of the neurons for both the hidden and output layers are sigmoidal functions. Thus, if the normalised value for T_{C1} is 0.7 and it has the fuzzy membership function of the type shown in Figure 8.4(c), then the outputs of the three μ neurons corresponding to T_{C1} would be

$$\mu_{L,T_{C1}} = 0 \quad \mu_{N,T_{C1}} = 0 \quad \mu_{H,T_{C1}} = 1$$

If the normalised value for T_{C1} is 0.7 and it has the fuzzy membership function of Figure 8.4(d), then the outputs of the three μ neurons corresponding to T_{C1} would be

$$\mu_{L,T_{C1}} = 0 \quad \mu_{N,T_{C1}} = 0 \quad \mu_{H,T_{C1}} = 0.8$$



Fuzzy membership function
1.0
0.0
L
ML
N
MH
H
(a)
(b)
(c)
(d)
0.7
Normalized values of process variables

Figure 8.4 Divisions of input and output variables into fuzzy regions and the corresponding membership functions.

(4) Use the data produced in Steps 2 and 3 to train the fuzzy neural network shown in Figure 8.5. The parts outside the dashed-line box are the fuzzy representation of input and output data. The part inside the dashed-line box is a normal three layer back propagation neural network. The training procedure uses error backpropagation which adjusts the weights between the input and output layers.

The fuzzy rules obtained by training the network are of two types. One is of the following form:

IF T_{C1} is High AND F_{C1} is Low
THEN T_{H11} is high (8.7)

This means that if the value of a variable belongs to a fuzzy class it belongs to it unambiguously. Another type of rule which is obtained from training the network is of the following form:

$$\begin{aligned}&\text{IF } T_{C1} \text{ is High } (\mu_{H,\,T_{C1}}) \text{ AND } F_{C1} \text{ is Low } (\mu_{L,\,F_{C1}})\\ &\text{THEN } T_{H11} \text{ is high } (\mu_{H,\,T_{H11}})\end{aligned} \tag{8.8}$$

The next section examines each of these rules.

8.4.1 Generating Rules Without Fuzzy Membership Values

If the membership functions of the variables of a fuzzy NN have the form of Figure 8.4(a) or (c), then the rules generated take the form of Expression 8.7, i.e., rules without fuzzy membership values. The maximum number of rules is determined by the network topology. If there are NOV input variables and each variable takes NFV fuzzy values (i.e., each input variable has NFV corresponding nodes, assuming all input variables take the same number of fuzzy values), then the maximum number of rules will be equal to NOV^{NFV}. This represents the maximum number of rules although it is likely that in practise it will be less since it depends on the range of data used to generate rules.

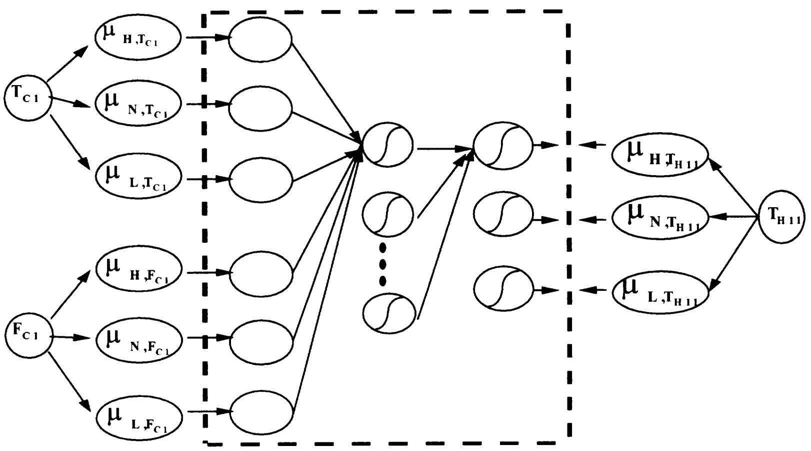


Figure 8.5 The architecture of the fuzzy neural network.

For the fuzzy NN structure shown in Figure 8.5 having two input variables and one output variable with each variable taking three fuzzy values, the maximum number of rules will be 3^2=9. If each variable takes five values (High, Medium High, Normal, Medium Low, and Low), then the maximum number of rules is 5^2=25;

In the case of a shell-and-tube heat exchanger of the type illustrated in Figure 8.6(a) [224]. The steady state values are shown in Figure 8.6(a). Assuming no leaks, then the cause-effect diagram for variables is shown in Figure 8.6(b). For simplicity, attention is restricted to fixed values of T_{HI} and F_{HI} so only F_{CI} and T_{CI} change. Then if each input variable takes three fuzzy values, i.e., Normal, High and Low, the situation is as depicted by Figure 8.4(c). A fuzzy NN structure for the heat exchanger is shown in Figure 8.5 and there will be a maximum number of 3^2=9 rules.

The boundary values of the variables for low, normal and high are indicated in Table 8.2. 87 input-output patterns are obtained by changing F_{CI} and/or T_{CI} based on Table 8.3. Details of simulation results are not given to save space but the resulting data patterns (as shown in Table 8.4) are given which are obtained after fuzzification of simulation results using the fuzzy membership function of Figure 8.4(c) and the boundary values of Table 8.2. Each pattern in Table 8.4 is a different form of a production rule, for example, data pattern 1 represents the following rule:

IF F_{CI} is Normal and T_{CI} is Normal
THEN T_{HII} is Normal

If there are no conflicts in any of the data patterns , such as in the case of data patterns 1, 2, and 3 which give the same rule then Table 8.4 is the final result. However, more often than not, there are conflicts in data patterns, i.e., rules have the same IF part but different THEN part. Thus data pattern 7 and 7* are examples which take the following form:

7: IF F_{CI} is High and T_{CI} is Normal
THEN T_{HII} is Low
7*: IF F_{CI} is High and T_{CI} is Normal
THEN T_{HII} is Normal

Here, 7 and 7* have the same IF part but different THEN part, the conflict has to be resolved. The approach being proposed here is to use the neural networks to train the data patterns in Table 8.4. Since neural networks are capable of dealing with noise, the noise data patterns can be identified and neglected. To demonstrate this, Table 8.4 contains 10 data patterns having noise which illustrate the conflicts. The 10 data patterns are shown with a "*" e.g., 7*. The result is an increase in the number of data patterns from 87 to 97.

The 97 data patterns are fed to a back propagation neural network (Figure 8.5) having 6 inputs (F_{C1} Low, F_{C1} Normal, F_{C1} High; T_{C1} Low, T_{C1} Normal and T_{C1} High) and 3 outputs (T_{H11} Low, T_{H11} Normal and T_{H11} High) with 6 hidden neurons. The activation functions of output and hidden layers are sigmoidal. The trained results after 5000 iterations are shown in Table 8.5.

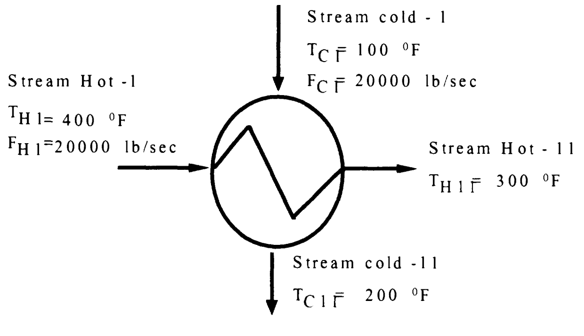
Table 8.5 gives the targets and predictions of T_{H11} for all the 97 data patterns of Table 8.4. For example, for data pattern 1, the target values for T_{H11} are L = 0, N = 1 and H = 0 and the predictions for T_{H11} are L = 0.1, N = 0.85 and H = 0.06.

It was found that for all 87 data patterns obtained from simulation, very good predictions are obtained, but for all 10 of the noisy data the errors are very large. The conflict can be resolved by comparing errors between the target values and predictions, keeping the rules with small errors and abandoning those with large errors.

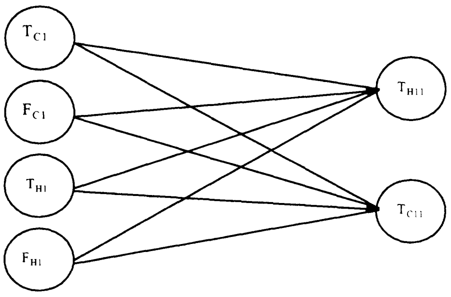
The conflicting data patterns 7 and 7* can be used as an example. By comparing the results in Table 8.5, it can be seen that the error for 7* is very large and for 7 is small. The square root of the sum of error squares for data patterns 7 and 7* are 0.26 and 1.15 respectively. So 7* should be rejected. The same conclusion can be reached by looking at the other 9 noise data patterns. The rules generated have been summarised in Table 8.6.

Table 8.2 The boundary values of low, normal and high for T_{H11}, F_{C1} and T_{C1}.

	L (Low)	N (Normal)	H(High)
T_{H11} ^{0}F	< 285	[285, 320]	>320
F_{C1} lb/sec	$<0.627*F_{C1,steady}$	$[0.627*F_{C1,steady}, 1.42*F_{C1,steady}]$	$>1.42*F_{C1,steady}$
T_{C1} ^{0}F	$<0.55*T_{C1,steady}$	$[0.55*T_{C1,steady}, 1.6*T_{C1,steady}]$	$>1.6*T_{C1,steady}$



(a)



(b)

Figure 8.6 The shell-and-tube heat exchanger and its cause-effect diagram.

Table 8.3 87 data patterns are obtained by changing F_{CI} and/or T_{CI}.

Data patterns	Coefficient ($F_{CI} = F_{CI}$* Coefficient)	Coefficient ($T_{CI} = T_{CI}$* Coefficient)
(1) ~ (5)	1.0, 1.1, 1.2, 1.3, 1.4	1.0
(6) ~ (9)	1.45, 1.50, 1.60, 1.65	1.0
(10) ~ (13)	0.9, 0.8, 0.7, 0.65	1.0
(14) ~ (16)	0.6, 0.5, 0.45	1.0
(17) ~ (20)	1.0	1.2, 1.3, 1.4, 1.5
(21) ~ (24)	1.0	1.65, 1.7, 1.8, 1.9
(25) ~ (27)	1.0	0.8, 0.7, 0.6
(28) ~ (31)	1.0	0.5, 0.4, 0.35, 0.3
(32) ~ (47)	1.45, 1.5, 1.6, 1.65	1.65, 1.7, 1.8, 1.9
(48) ~ (59)	0.6, 0.5, 0.45	1.65, 1.7, 1.8, 1.9
(60) ~ (75)	1.45, 1.5, 1.6, 1.65	0.5, 0.4, 0.35, 0.3
(76) ~ (87)	0.6, 0.5, 0.45	0.5, 0.4, 0.35, 0.3

Table 8.4 Fuzzification results of data patterns obtained through simulation using the membership function of Figure 8.4(c).

	F_{C1}			T_{C1}			T_{H11}		
No.	L	N	H	L	N	H	L	N	H
1	0	1	0	0	1	0	0	1	0
2	0	1	0	0	1	0	0	1	0
3	0	1	0	0	1	0	0	1	0
3*	0	1	0	0	1	0	1	0	0
7	0	0	1	0	1	0	1	0	0
7*	0	0	1	0	1	0	0	1	0
11	0	1	0	0	1	0	0	1	0
11*	0	1	0	0	1	0	0	0	1
18	0	1	0	0	1	0	0	1	0
18*	0	1	0	0	1	0	1	0	0
22	0	1	0	0	0	1	0	0	1
22*	0	1	0	0	0	1	1	0	0
29	0	1	0	1	0	0	1	0	0
29*	0	1	0	1	0	0	0	0	1
40	0	0	1	0	0	1	0	1	0
40*	0	0	1	0	0	1	1	0	0
50	1	0	0	0	0	1	0	0	1
50*	1	0	0	0	0	1	1	0	0
70	0	0	1	1	0	0	1	0	0
70*	0	0	1	1	0	0	0	1	0
80	1	0	0	1	0	0	0	1	0
80*	1	0	0	1	0	0	0	0	1
81	1	0	0	1	0	0	0	1	0

Table 8.5 Target and predicted fuzzy membership values for T_{H11}.

	F_{C1}		T_{C1}		T_{H11}	
No.	(T,	P)	(T,	P)	(T,	P)
1	0	.10	1	.85	0	.06
2	0	.10	1	.85	0	.06
3	0	.10	1	.85	0	.06
3*	1	.10	0	.85	0	.06
7	1	.82	0	.19	0	.01
7*	0	.82	1	.19	0	.01
11	0	.10	1	.85	0	.06
11*	0	.10	0	.85	1	.06
18	0	.10	1	.85	0	.06
18*	1	.10	0	.85	0	.06
22	0	.20	0	.02	1	.81
22*	1	.20	0	.02	0	.81
29	1	.80	0	.03	0	.06
29*	0	.81	0	.03	1	.06
40	0	.06	1	.95	0	.03
40*	1	.06	0	.95	0	.03
50	0	.07	0	.01	1	.94
50*	1	.07	0	.01	0	.94
70	1	.94	0	.05	0	.01
70*	0	.94	1	.05	0	.01
80	.0	.01	1	.78	0	.24
80*	.0	.01	0	.78	1	.24
81	.0	.01	1	.77	0	.24

Table 8.6 Rules generated.

IF	FC1 **and**	TC1 **THEN**	TH11
	Normal	Normal	Normal
	High	Normal	Low
	Low	Normal	High
	Normal	High	High
	Normal	Low	Low
	High	High	Normal
	Low	High	High
	High	Low	Low
	Low	Low	Normal

8.4.2 Generating Rules With Fuzzy Membership Values

If fuzzy membership functions take the form shown in Figure 8.4(b) or (d), then rules of the form of Expression 8.8 are obtained. Translating all the data into rules of the form of Expression 8.8, there are as many rules as the number of data patterns. Therefore an important step is to deal with number of rules. Two approaches are proposed and described next.

8.4.2.1 λ -Cut approach

One way to reduce the number of rules is to reject the rules that are less reliable. There are two situations which make rules less reliable. One is that in Expression 8.8, $\mu_{H,T_{C1}}$, $\mu_{L,F_{C1}}$, and $\mu_{H,T_{H11}}$ are small because small membership values mean it is ambiguous in identifying which fuzzy values should be used. The other situation is that the error, ER, for the data pattern is large when training is completed. It is therefore necessary to define a confidence factor, CF, representing reliability:

$$CF = \mu_{H,T_{C1}} \mu_{L,F_{C1}} \mu_{H,T_{CH11}} / ER \tag{8.9}$$

A λ -Cut value can then be defined as the threshold for the rule to be worth keeping. If the confidence factor of the rule is smaller than the λ -Cut value then the rule should be abandoned. Adjusting the λ *-Cut* value can change the number of rules generated.

Referring to the heat exchanger the simulation results of the heat exchanger shown in Table 8.3 are first converted to fuzzy patterns using corresponding fuzzy membership functions. The function used in this case is represented in Figure 8.4(b) and (d) and described by the following expression:

$$y = e^{-|x|^l} \tag{8.10}$$

Here, x changes in the range [-2, 2] and l changes from 0.51 to 5.0. In this example, $l = 1.0$. The boundary values for Low, Normal and High for the three variables (F_{C1}, T_{C1} and T_{H11}) are shown in Table 8.7. There is a little difference from Table 8.2 but it does not affect the principle to be demonstrated.

By applying the fuzzy membership functions in Equation 8.10 to the 87 simulation data patterns corresponding to Table 8.3, data patterns can be converted to fuzzy rules of the following form,

IF F_{C1} is Normal (0.61) and T_{C1} is Normal (1.0)
THEN T_{H11} is Normal (0.67).

All the 87 data patterns after fuzzification are input to the neural network having 6 hidden neurons, as shown in Figure 8.5. The network has six input neurons (F_{C1} Low, Normal and High as well as T_{C1} Low, Normal and High) and three outputs (T_{H11} Low, Normal and High).

After training, the confidence factors CF are shown in Table 8.8. A large value of CF means that it is more reliable. So a λ -cut value is used to select the desired number of rules. By changing the λ -cut values it is possible to change the number of rules, as is demonstrated in Table 8.9. For example, with a λ -cut value x10 is 6.0, there are 17 rules and for 5.0 are 21 rules. The rules for the former case are shown in Table 8.10.

The λ -cut approach can effectively select the desired number of rules which then can be used by a fuzzy expert system shell to develop an expert system. In applications, when the input data does not exactly match the IF parts, interpolation and extrapolation are required.

Table 8.7 Boundary values the three variables for Low, Normal and High.

	L	N	H
T_{H11}	<280	[280, 320]	>320
F_{C1}	12000	[12000, 28000]	>28000
T_{C1}	<40	[40, 160]	>160

Table 8.8 Confidence factors for some example data patterns.

Data patterns	1	2	3	4	5	...	48	49	50
Confidence Factor x 10	24.01	32.23	3.07	.89	.41	...	.21	.35	.79
51	52	53	54	55	56	57	58	59	...
3.28	.59	1.54	7.99	9.82	.34	4.20	5.03		...

Table 8.9 Number of rules generated can change with changing λ - cut values.

λ -Cut value x 10	1.0	1.5	2.0	3.0	5.0	6.0	8.0	9.0	10.
Number of rules	48	42	32	25	21	17	11	8	6

Table 8.10 Fuzzy rules generated (λ -cut value x 10 = 6.).

No.	**IF** FC1	**and** TC1	**THEN** TH11
(1)	(N, 1.00)	(N, 1.00)	(N, 0.99)
(2) (10)	(N, 0.61)	(N, 1.00)	(N, 0.67)
(17)(25)	(N, 1.00)	(N, 0.51)	(N, 0.51)
(18)(26)	(N, 1.00)	(N, 0.37)	(N, 0.37)
(40)	(H, 0.67)	(H, 0.19)	(N, 0.55)
(47)	(H, 1.00)	(H, 1.00)	(N, 0.22)
(54)	(L, 0.51)	(H, 0.51)	(H, 0.82)
(55)	(L, 0.51)	(H, 1.00)	(H, 0.96)
(57)	(L, 1.00)	(H, 0.26)	(H, 0.87)
(70)	(H, 0.67)	(L, 0.37)	(L, 0.79)
(71)	(H, 0.67)	(L, 1.00)	(L, 0.90)
(74)	(H, 1.00)	(L, 0.37)	(L, 0.87)
(75)	(H, 1.00)	(L, 1.00)	(L, 1.00)
(83)	(L, 0.51)	(L, 1.00)	(N, 0.28)

8.4.2.2 Neuro-expert approach

Although the λ -cut can effectively select the desired number of rules, the rules obtained do not keep all the information inherent in the original data. Some information is lost. The neuro-expert system approach is different in that it can make use of all the information in the data. It requires writing the basic number of rules (i.e., NOV^{NFV}). For two inputs, where each variable takes three values, the basic number of rules are $3^2=9$. These rules are of the form of expression (8.11) in which $\mu_{H,\ x_1}$, $\mu_{L,\ x_2}$ and $\mu_{MH,\ y}$ are variables that can be any values in the range of [0, 1].

$$\text{IF } x_1 \text{ is High } (\mu_{H,\ x_1}) \text{ AND } x_2 \text{ is Low } (\mu_{L,\ x_2})$$
$$\text{THEN } y \text{ is medium high } (\mu_{MH,\ y}) \qquad (8.11)$$

When an expert system is used, it receives fuzzy values of x_1 and x_2 and the corresponding μ values. The expert system gives the fuzzy values for y and uses a trained neural network algorithm (based on the weights) to calculate the μ for y.

In conclusion, it is essentially the same as the neural network but it has the following important features. Firstly, it is able to give explanations about how conclusions are reached. In the case of the heat exchanger, it can also give the following explanations:

Conclusion: T_{H11} is (Normal, 0.24), CF x10 is 2.95.
HOW (user input asking how conclusion was reached.)
IF F_{C1} is (Low, 0.51) and T_{C1} is (Low, 0.37)
THEN T_{H11} is (Normal, 0.24)
Confidence factor x10 = 2.95

Secondly, the basic rules are represented as cause-effect knowledge in contrast to the black box feature of a neural network.

8.5 Discussion

Both neural networks and expert systems are now starting to realise their potential in developing intelligent operational decision support systems. Knowledge used by an expert system to reach conclusions is transparent and can be displayed through the HOW and WHY explanation facilities. However, a critical problem with expert systems is knowledge acquisition. The methods introduced in this Chapter are able to extract production rules automatically from data therefore are potentially very useful.

The *fuzzy neural network* method is similar to the method of *rule generation by fuzzy set operation* [205, 207] in the first two steps. The major difference is in dealing with conflicts. The *fuzzy set operation* method calculates the extent to which a rule depends on a factor derived from human experts. This is not practical in many applications and so efficiency can be a serious problem when large amounts of data are involved. The fuzzy NN described in this chapter is not an incremental approach. Frayman and Wang [225] developed a recurrent fuzzy NN which is able to learn incrementally.

The *rough set method* [219] initially was not able to generate fuzzy rules. Quafafou and Chan [209] have improved the method so as to be able to generate fuzzy rules but there are still limitations, as indicated by Quafafou and Chan [209].

The *neural network method* by Fu [217] depends on the explanation of a properly trained structured neural network. The nodes represent concepts and the branches describe cause-effect relations. However, since hidden neurons are used, the knowledge obtained includes concepts which do not have physical significance. Another problem is that the number of rules increases dramatically with increase in the size of the problem.

Chan [219] has classified the various methods into two categories: incremental and non-incremental learning. The former works on one example at one time and the latter on a mass of training examples. The methods of *the fuzzy set operation* and *the rough set* belong to the incremental method group. The *neural network method* by Fu [217] is non-incremental for generating the trained neural network but becomes incremental when the weights are explained. The method proposed in this paper belongs to incremental. A disadvantage is that some information contained in the numerical data may be lost during conflict filtering, which is an essential step in incremental methods. This is unlike the NN which takes account of all data used for training to obtain the weight. However, incremental learning has its advantage that it is a recursive method, so that as new data becomes available, it updates the system and changes the existing knowledge base. This is particularly helpful in on-line application, as new data may be continuously made available for updating.

Another very effective technique for automatic generation of rules from data is inductive learning which has been introduced in Chapter 7 therefore is not repeated here.

CHAPTER 9

INFERENTIAL MODELS AND SOFTWARE SENSORS

9.1 Feedforward Neural Networks as Software Sensors

Process control contributes to process operational performance in safety, environmental protection and product quality through proper monitoring and control of process variables. Some of the variables including flowrate, temperatures, pressures and levels can be easily monitored on-line with cost effective and reliable measuring devices. Some variables, however, are often analysed off-line in laboratories because they are either too expensive or technically unreliable to install on-line instrument. These variables often relate to product quality and environmentally significant parameters such as composition and physical properties. A significant delay is normally incurred in laboratory testing, which is in the range of half an hour to a few hours. Such a delay makes it too late to make timely adjustments. An inferential or software sensor can predict the primary output variable from process inputs and other measured variables without incurring the measurement delay. Such prediction is necessary for implementation of inferential control.

Software sensors assume that there is a relationship between some easily measured and continuously available variables and the variable to be inferred. Traditionally mathematical models have been used which are obtained from fundamental domain knowledge. However, for many chemical and biochemical processes such fundamental models are not available due to lack of knowledge about these processes. A promising approach is to develop systems that can learn to develop inferential models from operational data. A major advantage of machine learning based approaches is that the model is able to be continuously improved as more and more data become available. Among the various approaches studied, feedforward neural networks (FFNN) have attracted wide attention and shown great potential as a basis for reliable software sensor design.

This chapter describes inferential models and software sensor design using FFNNs, with emphasis being put on addressing a number of critical issues. Firstly, FFNN is recognised to have the danger of extrapolation beyond the parameter space used for the training data. It is therefore important to select the data for model development and test with some care. This also means that there is a need to know when the FFNN model needs to be retrained with new data during use. In Section 9.2, a method for addressing the extrapolation issue is introduced. An industrial case study describing the development of a software sensor using the approach is presented in Section 9.3. Secondly, how to select input variables is also an important issue. Because of inadequate knowledge of domain problem, people tend to use more variables than necessary while some may be redundant. Redundant variables deteriorate the performance. Methods are introduced in Section 9.4 to select input variables. Finally, Section 9.5 gives a brief introduction to dynamic neural networks for software sensor development.

9.2 A Method for Selection of Training / Test Data and Model Retraining

A limitation of FFNN models is that they are data intensive because they are trained and not programmed [145]. Therefore when such a model is being developed, it is important to divide the data into training and test sets so that the model developed can be validated. In addition, the trained model needs to be periodically updated to maintain the accuracy of the model. The traditional way of dividing a database into training and test sets is through random sampling. Random sampling of test data has major shortcomings. If more sampled patterns belong to the sparse region of the high dimensional parameter space, the results will not be reliable, because the training patterns from that region are not adequate. On the other hand if more sampled patterns for testing come from the dense space, the model will give over optimistic estimation. Another problem is to decide when the model needs to be retrained. It is generally assumed that if the values of all input variables are within the boundaries of the training data, the model is reliable, but this assumption is questionable. Given a new data pattern, with input attributes within the boundaries of the training data, it is still difficult to tell if it is covered by the training data since the new data represents a point in a very high dimensional non-linear parameter space.

To address this issue, we have proposed a method which combines unsupervised clustering approach and FFNN, using unsupervised clustering to automatically cluster the data into classes so that data patterns within a class can be distinguished from those in other classes and so form a basis for sampling test data patterns [227]. Application of the method to the development of a software sensor for the main fractionator of a refinery fluid catalytic cracking process is described to illustrate the approach.

Figure 9.1 schematically depicts the method. It is divided into two stages, model development and model maintenance. In model development, data are first grouped into classes using the unsupervised clustering system AutoClass, a Bayesian clustering approach that has been introduced in Chapter 6 and can automatically group multivariate data patterns into clusters. Test patterns for FFNN are sampled from each class according to the size of the class. During model maintenance, it is necessary to decide when the model requires to be retrained. Because of the high dimensionality and large volume of data, it is difficult to carry out the analysis manually. The approach used here mixes new data with older data and then analyses it using AutoClass. If any of the following three situations arises, the model needs to be retrained.

- New classes are formed that cover mainly new data. This means that the new data are located in new parameter spaces.
- Some new data are assigned to small sized existing classes. This implies that the parameter spaces covered by small sized classes are insufficiently trained. When new data are available, the model should be retrained to improve its performance.
- New data are assigned to large classes and the degree of confidence estimated using the old model for the new patterns is low.

Other unsupervised machine learning approaches introduced in Chapter 6 could also be used. AutoClass is used here because its degree of autonomy is high. For example, users are not required to give a threshold for the similarity measure. The algorithm was discussed in detail in Chapter 6 so will not be repeated here.

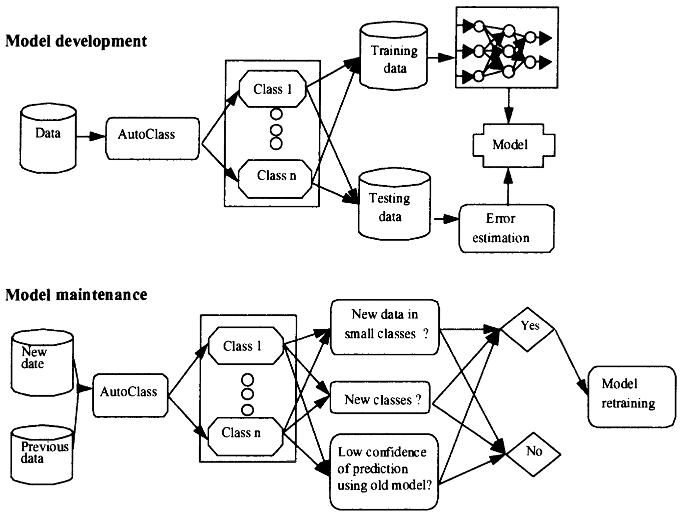


Figure 9.1 A combined framework for software sensor design using automatic classification and FFNN.

9.3 An Industrial Case Study

This section describes the application of the above method to an industrial process for developing a software sensor predicting product quality.

9.3.1 The Main Fractionator and Product Quality Prediction

The fluid catalytic cracking process of the refinery has been described in Chapter 4 (Figure 4.12). One of the products is light diesel which is typically characterised by temperature of condensation. Previously the temperature of condensation was monitored by off-line laboratory analysis. The sampling interval is between four to six hours and it is not practical to sample it more frequently since the procedure is time consuming. This deficiency of off-line analysis is obvious and the time delay is a cause of concern because control action is delayed. Moreover, laboratory analysis requires sophisticated equipment facilities and analytical technicians. Clearly there is great scope for a software sensor to carry out such on-line monitoring.

Fourteen variables are used as inputs to the FFNN model as shown in Figure 9.2. These variables are also summarised in Table 4.1.

Initially 146 data patterns (Dataset-1) from the refinery have been used to develop the model (Model-1). Later, three different sets of data became available having data patterns of 84 (Dataset-2), 48 (Dataset-3) and 25 (Dataset-4) respectively. Model-1 has therefore been modified three times with the versions being denoted by Model-2, Model-3 and Model-4. There are interesting and different results for every model improvement, as described below. In all cases, the accuracy of the model is required to be within $\pm 2^0$C.

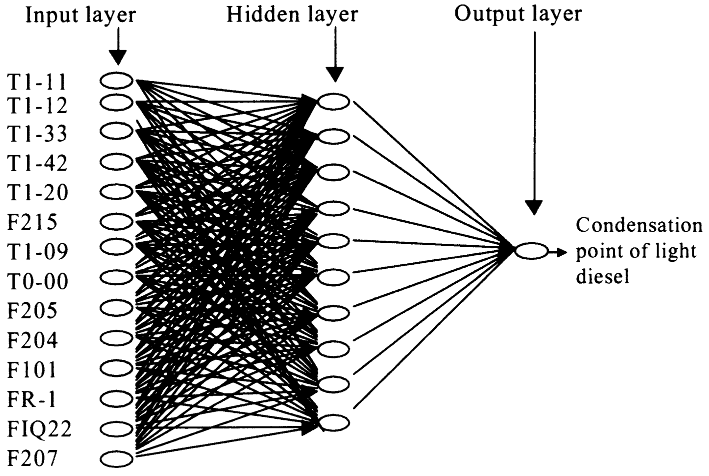


Figure 9.2 The feedforward neural network structure.

9.3.2 Model Development (Model-1) Using Dataset-1

The data set (Dataset-1) used for model development comprises 146 patterns. As illustrated in Figure 9.1, the first step is to process the data using AutoClass. It requires selecting a density distribution model for all attributes (i.e., all the inputs to the FFNN, Figure 9.2). All variables have real values and no data are missing so a Gaussian model is used.

AutoClass predicts seven classes (numbered 0, 1, ..., 6) as shown in Table 9.1. For example, class 0 has 32 members (i.e., 32 data patterns). Test data are sampled from each class and they are indicated in bold and underlined in Table 9.1. More data patterns are sampled from larger classes and fewer from smaller classes.

Altogether there are 30 data patterns used for testing and 116 for training. The procedure is summarised in Table 9.2.

The assignment of a data pattern to a class is fuzzy in the sense that there is a membership probability. Examples of membership probabilities are shown in Table 9.3. For instance, data pattern 17 (the third row in Table 9.3) has a membership probability of 0.914 to class 0, 0.055 to class 1 and 0.030 to class 4. It therefore is assigned to class 0.

The FFNN software sensor model (Model -1) is obtained when the training error reaches 0.424. With normalized [0, 1] training data, the error is calculated using $\frac{1}{2}\sum_{i=1}^{n}\left(y_i' - y_i\right)^2$, where y_i' is the prediction by the model for the ith training pattern and y_i the target value. There are three training patterns and two test patterns with absolute errors exceeding the required $\pm 2^0$C. Therefore the degree of confidence for training data is 100% - (3/116)% = 97.4%, and that for test data is 100% - (2/30)% = 93.3%. A degree of confidence of 90% is considered acceptable by the refinery.

Table 9.1 Classification of Dataset-1 and test data selection for FFNN[a].

Class 0 Weight 32 5 **6** 16 17 31 32 34 35 **36** 37-39 **40** 41-43 81 103 **104** 105 108 **109** 110 112 114 **115** 116 118 136 **137** 138 139	*Class 1 Weight 31* 1 10 25 **26** 27 44 45 **46** 47-50 **51** 52 76 79 **80** 82 120 **121** 122-24 **125** 126 128 129 132 **133** 134 135	*Class 2 Weight 29* 2 **3** 4 15 18 19 **64** 65 69 70 72 **73** 74 75 83 84 **85** 86-88 **89** 90 91 93 94 **95** 96 97 146	
Class 3 Weight 26 11 **12** 13 14 53 54 **55** 56-59 **60** 61-63 66 67 **68** 71 77 78 92 98 **99** 100 143	*Class 4 Weight 11* 7 9 28 **29** 33 111 117 127 **130** 131 145	*Class 5 Weight 9* **20** 21 30 101 102 106 140 **141** 142	*Class 6 Weight 8* 8 22 **23** 24 107 113 119 144

[a] data in bold and underlined are test data; weight – number of members

Table 9.2 Test and training data from the 146 patterns in Dataset-1.

Classes	0	1	2	3	4	5	6
Class weight	32	31	29	26	11	9	8
Number of patterns for training	25	24	23	21	9	7	7
Number of patterns for test	7	7	6	5	2	2	1

Table 9.3 Examples of class membership probability.

Data Pattern	(Class, Membership probability)	(Class, Membership probability)	(Class, Membership probability)
16	(0, 0.803)	(1, 0.197)	
17	(0, 0.914)	(1, 0.055)	(4, 0.030)
31	(0, 0.984)	(1, 0.014)	(6, 0.002)
32	(0, 1.000)		
34	(0, 0.995)	(6, 0.005)	
35	(0, 1.000)		
36	(0, 0.967)	(1, 0.003)	(6, 0.001)
37	(0, 0.996)	(1, 0.002)	

9.3.3 Model-1 Improvement Using Dataset-2

The production strategy changes according to season. The accuracy of Model-1 originally developed was later found to be inadequate and 84 more data patterns (Dataset-2) were provided in order to improve model performance. Initial application of Model-1 to the 84 data patterns indicated that the degree of confidence is only 100% - (63/84)% = 25.0%.

Table 9.4 Classification result of dataset-1 plus dataset-2[a].

Class 0 ***Weight 38*** 2 **3** 4 14 15 **18** 19 21 64 **65** 66-69 **70** 71-75 82-84 86 **87** 88-90 **91** 93 94 95 96-98 **99** 100 146	***Class 1*** *Weight 31* 5 6 17 29 32 **35** 36-39 **40** 41-43 81 103 **104** 105 108 109 **110** 111 112 114 **115** 116 117 136 **137** 138 139	***Class 2*** *Weight 26* *160* ***161*** *175 177-179 182 186 187* ***192*** *193-195* ***196*** *197 209 210* ***211*** *212 215* ***216*** *217-219* ***220*** *221*	***Class 3*** *Weight 19* 1 10 16 **44** 45-48 **49** 50-52 **53** 54 55 79 **80** 121 122	***Class 4*** *Weight 17* 11 **12** 13 25 56 **57** 58-60 **61** 62 63 77 **78** 85 92 143
Class 5 *Weight 15* *157 176* ***180*** *181 183* ***184*** *185 188* ***189*** *190 191 205* ***206*** *207 208*	***Class 6*** *Weight 14* 7 8 22 **23** 24 31 33 **34** 76 107 113 **118** 119 144	***Class 7*** *Weight 12* *162* ***163*** *164 222 223* ***224*** *225-227* ***228*** *229 230*	***Class 8*** *Weight 11* *158 159 198* ***199*** *200-202* ***203*** *204 213 214*	***Class 9*** *Weight 10* *147 148* ***149*** *150-152* ***153*** *154-156*
Class 10 *Weight 10* 9 26 28 120 **123** 124 127 130 **131** 145	***Class 11*** *Weight 9* 27 125 126 **128** 129 132 133 **134** 135	***Class 12*** *Weight 8* 20 30 **101** 102 106 140 **141** 142	***Class 13*** *Weight 7* *166* ***167*** *168-170* ***171*** *172*	***Class 14*** *Weight 3* *165 173 174*

[a] In italic refers to data patters from dataset-2; In bold and underlined are used for test.

The 84 new data patterns were combined with dataset-1 and processed by AutoClass. It was found that the 230 data patterns (84 + 146) were classified into 15 classes. Interestingly, all the 84 new data patterns in dataset-2 were classified into seven new classes (classes 2, 5, 7, 8, 9, 13, 14), while all the 146 patterns in dataset-1 are in eight classes (classes 0, 1, 3, 4, 6, 10, 11, 12). This is consistent with the poor predictions for the dataset-2 using Model-1. A summary of the classification is given in Table 9.4 where the data in italic are from dataset-2. The data patterns in bold and underlined are chosen for testing and the rest for retraining to generate model-2. All together 49 data patterns are chosen for test, of which 19 are from Dataset-2 and 30 from Dataset-1. The degree of confidence for the training data is 100% - (11/181)% = 92.8% and for test data 100%-(3/49)% = 93.9%.

9.3.4 Improvement Using Dataset-3

Later further 48 data patterns (Dataset-3) were provided. Model-2 was applied to dataset-3 and the degree of confidence was found to be 100% - (27/48)% = 43.8%. This is low but better than the prediction to dataset-2 using Model-1 (25.0%). The 48 new data patterns are then mixed with dataset-1 and dataset-2 and processed by AutoClass to give sixteen classes (a detailed classification is given in Table 9.5). It is found that twenty-five of the forty-eight data patterns form a new class (class 1 in Table 9.5, data patterns 251 - 275) but the rest are assigned to the classes combined with data from dataset-1 and dataset-2. The degree of confidence using model-2 to predict the classes 1, 2, 3, 8 and 11 are summarised in Table 9.6. It can be seen that class 1 contains only new data patterns and has the lowest degree of confidence for prediction using model-2. Nineteen of the twenty-five patterns in dataset-1 have deviations bigger than $\pm 2^0$C. Classes with fewer new data patterns have a higher degree of confidence in the predictions. This result further demonstrates the advantage of using AutoClass for clustering data before training a FFNN model. It is also found that a confidence in a model for predicting new data is lower if more data are grouped into new classes. For example, all patterns in dataset-2 are grouped into new classes and the confidence of predicting dataset-2 using Model-1 is 25.0%; while some patterns in dataset-3 are classified into old classes, with a confidence in predicting dataset-3 using Model-2 of 43.8%.

Table 9.5 Classification result of dataset-1, 2 and 3[a].

Class 0 *Weight 50*	***Class 1*** *Weight 25*	***Class 2*** *Weight 22*	***Class 3*** *Weight 22*	***Class 4*** *Weight 9*
1 6 7 **9** 10 16 **17** 22 **23** 26 **28** 29 **31** 33 **34** 36-39 **40** 41 47 48 **49** 50 51 **52** 76 81 **82** 111 **113** 117 **118** 120 **21** 122 **123** 124 **126** 127 **128** 129 130 **131** 133 134 **135** 138 145	*251* ***252*** *253-* *255* ***256*** *257-* *59* ***260*** *261-* *264* ***265*** *266-* *268* ***269*** *270-* *273* ***274*** *275*	160 **161** 175 177 **178** 179 186 187 **192** 193 209 212 216 **217** 218 219 220 **221** *234* ***235*** *236* ***237***	162 176 182 **183** 184 185 222 **223** 224-227 **228** 229-*231* ***232*** *233 238* ***239*** *240* ***241***	157 158 180 181 **188** 189 190 191 198 199 **200** 201-203 **204** 205 206 **207** 208

Class 5 *Weight 18*	***Class 6*** *Weight 18*	***Class 7*** *Weight 18*	***Class 8*** *Weight 13*	***Class 9*** *Weight 12*
5 32 35 42 **43** 103 104 105 108 109 **110** 112 114 115 116 **136** 137 139	11 **12** 13 53 54 **55** 56-69 **60** 61-63 77 **78** 79 85	8 20 24 27 30 **44** 45-46 80 **101** 102 119 125 132 140 **141** 142 144	147 **148** 149-151 **152** 153 154 **155** 156 ***276*** *277* ***278***	21 **25** 73 83 87 **92** 93 94 100 **106** 107 143

Class 10 *Weight 12*	***Class 11*** *Weight 11*	***Class 12*** *Weight 10*	***Class 13*** *Weight 10*	***Class 14*** *Weight 10*	***Class 15*** *Weight 8*
2 14 15 64 **65** 66 67 68 70 71 **98** 99	163 164 ***242*** *243-245* ***246*** *247-249* ***250***	3 18 19 72 84 86 **88** 89 90 91	165 166 167 168-171 **172** 173 174	159 194 195 196 197 210 **211** 213 214 215	4 69 74 **75** 95 96 97 146

[a] Italic – data from dataset-3; in bold and underlined – test data

The 48 new data patterns in dataset-3 have been combined with datasets-1 and -2 to develop model-3. The sampling of test data patterns follows the same procedure as before. A total of 68 data patterns has been used for testing with the remaining 210 patterns for training. Model-3 has a degree of confidence of 92.6% (=100% - (5/68)%) for testing data and 93.8% (=100%-(13/210)%) for the training data.

Table 9.6 Degree of confidence to using model-2 to predict dataset-3.

Classes	Number of data patterns from dataset-3	Number of data patterns from datasets-1 and 2	Total data patterns in the class	Number of predictions with an absolute error greater than ±2^{0}C	Degree of confidence using model-2 to predict the class
1	25	0	25	19	24.0%(=100%-(19/25)%)
2	4	18	22	0	100.0%
3	7	15	22	4	57.1%
8	3	10	13	0	100.0%
11	9	2	11	4	44.4%

9.3.5 Improvement Using Dataset-4

Dataset-4 was subsequently obtained from the refinery and has 25 new data patterns. The confidence of applying model-3 to dataset-4 is 100%-(8/25)% = 68.0. Dataset-4 was then combined with previous data sets to give a database of 303 patterns. These are classified by AutoClass into fifteen classes. 77 data patterns have been selected as test data and the rest for training to develop the FFNN model-4. The model has a confidence of 92.5% (=100%-(17/226)%) for training data and 90.9% (=100%-(7/77)%) for test data. In order to make a comparison with the conventional sampling approach, 77 data patterns have selected using random sampling as test patterns. The training is terminated using the same criterion, i.e., a training error of $5.75e^{-1}$. On this basis, 11 test data patterns have errors exceeding $\pm 2^0$C. So the confidence for the test data is 100%-(11/77)% = 85.7%. This is lower than that of the test data used in this study, which is 90.9% (100%-(7/77)%). The confidence of the training data for both approaches is the same, 92.5%.

Table 9.7 The changing ranges of input /output variables of model-4.

Attribute	Unit	Input	Output	Max.	Min.	Range
T1-11	^{0}C	√		205.40	181.50	23.90
T1-12	^{0}C	√		262.30	215.10	47.20
T1-33	^{0}C	√		228.00	186.10	41.90
T1-42	^{0}C	√		285.00	246.80	38.20
T1-20	^{0}C	√		187.30	129.20	58.10
F215	Ton/hr	√		170.00	106.00	64.00
T1-09	^{0}C	√		113.00	102.30	10.70
T1-00	^{0}C	√		511.00	493.00	18.00
F205	Ton/hr	√		118.40	80.00	38.40
F204	Ton/hr	√		39.50	10.00	29.50
F101	Ton/hr	√		3.46	2.39	1.07
FR-1	Ton/hr	√		10.00	6.40	3.60
FIQ22	Ton/hr	√		7.63	3.30	4.33
F207	Ton/hr	√		14.00	6.80	7.20
TS3	^{0}C		√	7.00	-13.00	20.00

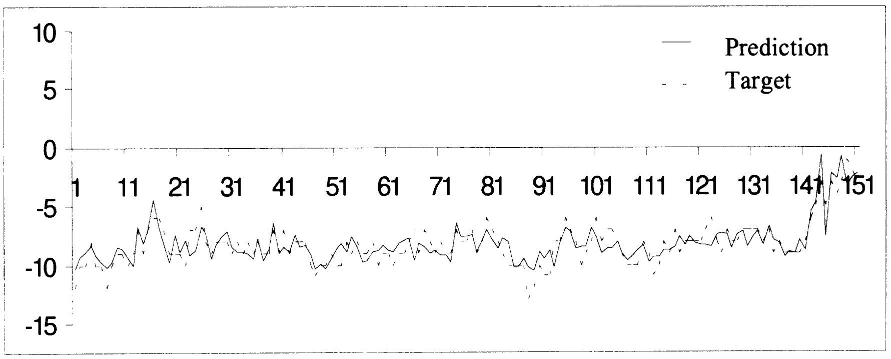


Figure 9.3 Comparison between predictions using Model-4 and the target values for the first 151 data patterns.

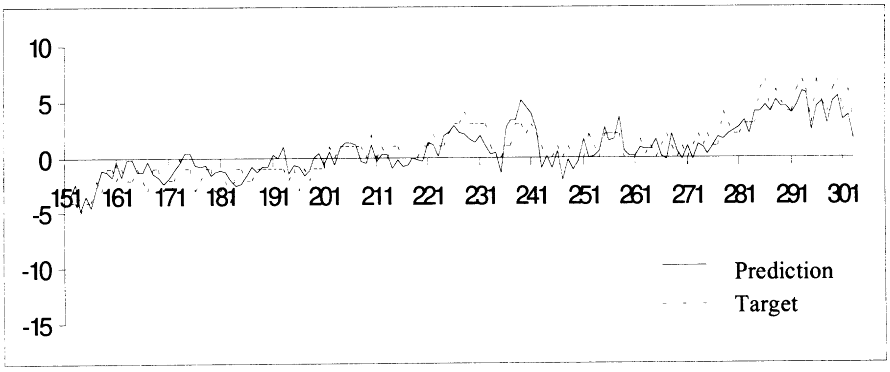


Figure 9.4 Comparison between predictions using Model-4 and the target values for the last 152 data patterns.

Model-4 (i.e., all datasets) covers all the operational seasons and is being used very satisfactorily. It has proved to be robust and reliable over a wide range of operational conditions. Figures 9.3 and 9.4 show the differences between predictions using Mode-4 and the target values of the 303 patterns. Variations in the input and output variables are summarised in Table 9.7. The plant has reduced the sampling frequency to twice a day compared with four to six times used previously and the intention is to reduce it to once a day in the future.

9.4 Dimension Reduction of Input Variables

In applying neural networks for software sensor design, researchers tend to include as many variables as possible to ensure that no relevant variables are omitted. This may sometimes result in a very large dimension of input. An unnecessary large dimension of input variables may have adverse effects. For a fixed number of training data patterns, with the increase of input variables it becomes more sparse in the multi-dimensional space, and therefore degrades the learning performance. The generality of the learned model may also be reduced due to inclusion of irrelevant or not important input variables. Increased number of inputs also means more connection weights to be determined and therefore leads to longer training time. Apart from irrelevant and not important variables that cause large dimension of input variables, there may be correlations between input variables. Correlated inputs make the model more sensitive to the statistical peculiarities of the particular sample; they accentuate the overfitting problem and limit generalisation - a common occurrence in linear regression.

Use of prior knowledge, if available to identify irrelevant or correlated variables is clearly useful. Otherwise mathematical techniques are required to solve the problem. In the following, we will be looking at several techniques.

9.4.1 Use of PCA for Input Dimension Reduction

Since the first few PCs are able to capture most of the variance of the original explanatory variables, it is possible to use only the first few PCs to replace the explanatory variables as the inputs to a FFNN model. Since normally not all PCs are required, the dimension of inputs is reduced. In addition, these PCs are linearly uncorrelated and orthogonal.

McAvoy et al. [228] studied the use of FFNNs to fluorescent spectral analysis data for predicting composition of a binary mixture. Thirty data points were sampled from each fluorescent spectrum which were used as inputs of a FFNN mode whose output is the composition of one of the components. Baughman and Liu [229] used the same data and developed a 30-20-1 FFNN for this purpose. Because there are only 46 data patterns available, the size of the network is relatively large.

Table 9.8 Principal components for the fluorescent spectral analysis data.

Original variables	PC_1	PC_2	PC_3	...	PC_{30}
x_1	8.74E-03	0.975	0.223	...	-2.42E-05
x_2	6.20E-02	0.995	7.37E-02	...	1.13E-04
x_3	9.50E-02	0.995	-7.16E-03	...	-1.42E-04
x_4	0.123	0.991	-4.39E-02	...	-8.74E-05
x_5	0.172	0.984	-4.90E-02	...	1.06E-04
x_6	0.28	0.958	-5.70E-02	...	-9.44E-05
x_7	0.472	0.88	-5.77E-02	...	2.92E-04
x_8	0.715	0.697	-5.06E-02	...	-5.28E-05
x_9	0.898	0.437	-3.94E-02	...	-2.27E-04
x_0	0.975	0.219	-2.81E-02	...	-1.30E-05
x_{11}	0.996	7.98E-02	-1.98E-02	...	1.26E-04
x_{12}	1.000	-6.04E-03	-1.32E-02	...	-3.17E-04
x_{13}	0.998	-5.80E-02	-7.49E-03	...	7.07E-04
x_{14}	0.996	-8.84E-02	-5.41E-03	...	-2.87E-04
x_{15}	0.994	-0.109	-9.89E-04	...	7.49E-05
x_{16}	0.993	-0.122	1.34E-04	...	-1.82E-04
x_{17}	0.991	-0.131	2.36E-03	...	1.01E-05
x_{18}	0.991	-0.137	3.75E-03	...	1.65E-05
x_{19}	0.99	-0.141	6.07E-03	...	5.93E-05
x_{20}	0.989	-0.144	7.51E-03	...	-1.72E-05
x_{21}	0.989	-0.145	8.06E-03	...	5.70E-05
x_{22}	0.989	-0.146	1.01E-02	...	-8.84E-05
x_{23}	0.989	-0.145	1.38E-02	...	7.60E-05
x_{24}	0.989	-0.145	1.62E-02	...	5.66E-06
x_{25}	0.989	-0.145	1.87E-02	...	-9.03E-05
x_{26}	0.99	-0.142	1.81E-02	...	9.29E-06
x_{27}	0.99	-0.139	1.69E-02	...	2.64E-05
x_{28}	0.99	-0.137	2.21E-02	...	-6.22E-05
x_{29}	0.99	-0.135	2.72E-02	...	-1.62E-05
X_{30}	0.991	-0.13	2.79E-02	...	3.38E-05
% of variance	**74.331%**	**25.398%**	**0.252%**	...	**3.241E-06%**
% of cummulative variance	**74.331%**	**99.729%**	**99.980%**	...	**100%**

We have carried out PCA analysis of the data and the result is shown in Table 9.8. Two observations are made. Firstly, it was found that the first two PCs can capture 99.729% of the variance. Secondly, $x_1 \sim x_7$ are mainly associated with PC_2 and $x_9 \sim x_{30}$ with PC_1. x_8 is nearly equally associated both PC_s.

Because the first two PCs can capture 99.729% of all the variance, it is possible to use the two PCs rather than the 30 x_s as the inputs to develop a FFNN model. As a result we have developed a 2-4-1 model. Comparison was made between the 30-20-1 model developed by Baughman and Liu [229] and the 2-4-1 model. For both models, 32 data patterns were used for training and 9 for test. The training and test errors for the 30-20-1 model are 3.02% and 3.87%, while for the 2-4-1 model are

2.29% and 3.55%. Because the difference in configuration of the two networks it is not appropriate to give a definite conclusion on which model is more accurate, only based on the above errors. Nevertheless it is obvious that the 2-4-1 model can give more accurate or equivalent result.

An alternative way to use PCA to reduce the input variable dimension is to analyse the contribution of individual explanatory variables to the first few PCs. Those which are not important can be excluded from the FFNN model.

9.4.2 Sensitivity Analysis

Sensitivity analysis is a simple method of finding the effect of an input on the output of the network. A FFNN can be developed first using all possible input variables and then be used to carry out sensitivity studies. Irrelevant variables can be excluded as a new FFNN is developed. Suppose there are a number of inputs x_i and only one output z, then the relationship of an input variable x_i and the output z is found by determining the impact of a small change in x_i on z. If large change occurs on z, then x_i is considered to be one of the key factors in producing the current activation value of z. The sensitivity can be deduced given a FFNN structure with multiple inputs [231], a number of hidden neurons and only single output. The input to a hidden neuron is

$$\mathrm{u}_j = \mathrm{a}_{0j} + \sum \mathrm{a}_{ij} x_i \tag{9.1}$$

where a_{0j} is the bias weight and x_i denotes an input with associated weight a_{ij}. The output of a hidden neuron is

$$y_j = \frac{1}{1 + e^{-\mathrm{u}_j}} \tag{9.2}$$

Then the output of the output node is

$$v = \mathrm{b}_0 + \sum \mathrm{b}_j y_j \tag{9.3}$$

where b_0 is the bias weight and b_j are the respective weights for the hidden nodes. Clearly the output $z = \dfrac{1}{1 + e^{-v}}$.

The sensitivity of the output z to the input variable x_i is $\mathrm{d}z/\mathrm{d}x_i$. Using the chain rule, the sensitivity may be related to the output z, the output from each of the hidden nodes and the different set of weights a_{ij} and b_j as follows,

$$\begin{aligned} \mathrm{d}z/\mathrm{d}x_i &= (\mathrm{d}z/\mathrm{d}v)(\mathrm{d}v/\mathrm{d}y_j)(\mathrm{d}y_j/\mathrm{du}_j)(\mathrm{du}_j/\mathrm{d}x_i) \\ &= z(1-z) \sum \mathrm{b}_j y_j (1 - y_j) \mathrm{a}_{ij} \end{aligned}$$

For a FFNN with multiple outputs, the same procedure can be applied to study the impact of each input on each output. Table 9.9 is an example of sensitivity study for a waste water treatment process [230]. It shows sensitivity of the standard deviation of suspended solid to a number of operational variables. It indicates that the classifier interface level is the most important variable and inlet oil concentration and inlet cyanide act have positive impact, while an increase in ammonia flow has a positive effect in reducing the standard deviation of suspended solids.

Table 9.9 The sensitivity of the standard deviation of suspended solids to various operational variables.

Variable	Average sensitivity
Interface lev0el	0.52
Sludge age	0.48
Inlet oil concentration	0.46
Inlet cyanide concentration	0.44
Aerated tank temperature	-0.42
Inlet ammonia flow	-0.33
Treated effluent pH	-0.33

Sensitivity can also be carried out by plotting the output variable against an input variable as shown in Figure 9.5 [175]. Figure 9.5 shows the relationships of variables of a waste water processing unit. PH-P is the measured PH value of the effluent from the unit and PH-E, SS-E and Q-E are the PH value, suspended solid and flowrate of the influent to the unit.

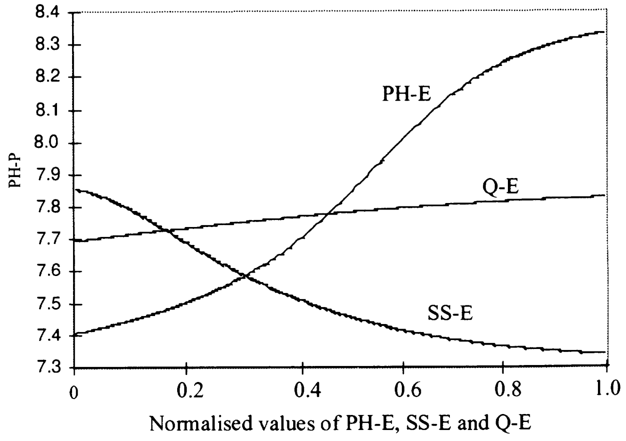


Figure 9.5 The sensitivity of the effect of PH-P to influent SS-E, Q-E and PH-E.

However the above sensitivity studies should be conducted with great care. If the inputs are strongly correlated then the sensitivity of the output to an input variable may be dependent on what values the other input variables are fixed at. If PCs rather than the explanatory variables are used, this problem can be resolved since all PCs are linearly uncorrelated. Although it may be able to estimate the impact of each PC on the output, to relate to the explanatory variables is not always possible. For example, Table 9.10 shows the coefficients of the first four PCs and the six explanatory variables $x_1 \sim x_6$. Since PC1 is nearly equally weighted on four (i.e., x_1, x_2, x_4, x_5) of the six variables and the PC1 only accounts about half of the total variance, even we can find the impact of PC1 on the output, it is difficult to determine the impact on the output by $x_1 \sim x_6$ [231]. A satisfactory situation would be that the first PC accounts for most of the total variance and/or is heavily weighted on a few explanatory variables.

Table 9.10 The coefficients of the first four PCs and the explanatory variables.

Variable	PC1	PC2	PC3	PC4
x_1	0.449	0.249	0.349	-0.447
x_2	0.530	-0.058	0.041	0.197
x_3	0.280	-0.537	-0.575	-0.542
x_4	0.457	-0.253	-0.149	0.669
x_5	0.467	0.223	0.267	-0.134
x_6	0.110	0.729	-0.673	0.042

9.4.3 Weight Matrix Analysis

Another approach to analyse the input impact on an output is to analyse the weight matrix. Since the nodes between two adjacent layers are fully connected, such an analysis is not straightforward. Garson [232] proposed the following measure for the proportional contribution of an input to a particular output,

$$G_{ik} = \frac{\sum_{j=1}^{n_h} \frac{w_{ij}}{\sum_{p=1}^{n_i} w_{pj}} w_{jk}}{\sum_{q=1}^{n_i} \left(\sum_{j=1}^{n_h} \frac{w_{qj}}{\sum_{p=1}^{n_i} w_{pj}} w_{qj} \right)} \tag{9.5}$$

In the above equation, p and q refer to the input layer, $p = 1$, n_i, j refers to hidden layer, $j = 1$, n_h, i, j, k are the ith, jth and kth input, hidden and output node. A disadvantage of the approach is that during the summation process, positive and negative weights can cancel their contribution, which leads to inconsistent results.

Milne [233] commented that the sign of the contribution is lost, and proposed the following measure,

$$G_{ik} = \frac{\sum_{j=1}^{n_h} \frac{w_{ij}}{\sum_{p=1}^{n_i} |w_{pj}|} w_{jk}}{\sum_{q=1}^{n_i} \left(\sum_{j=1}^{n_h} \left| \frac{w_{qj}}{\sum_{p=1}^{n_i} w_{pj}} w_{qj} \right| \right)} \tag{9.6}$$

Wong et al. [234] and Gedeon [235] defined measures for the contribution of an input to a neuron in the hidden layer, and a hidden layer neuron to an output layer neuron,

$$P_{ij} = \frac{|w_{ij}|}{\sum_{p=1}^{n_i} |w_{pj}|} \tag{9.7}$$

$$P_{jk} = \frac{|w_{jk}|}{\sum_{r=1}^{n_h} |w_{rk}|} \tag{9.8}$$

Then the contribution of an input neuron to an output neuron is,

$$Q_{ik} = \sum_{r=1}^{n_h} (P_{ir} \times P_{rk}) \tag{9.9}$$

The approaches using weight matrix to examine the impact of input nodes on output nodes are interesting but they have not been widely tested.

9.5 Dynamic Neural Networks as Inferential Models

The data used in the industrial case study described in Section 9.3 was collected on steady state basis. Consequently the model developed will only be accurate if the operation is around steady states. A feedforward neural network is static in nature and is not designed to deal with dynamics, e.g., time delays. Recently several approaches have been proposed to introduce dynamics to FFNNs, in order to capture the dynamic nonlinear feature of processes. Such neural networks can essentially be classified according to implementation of the dynamic character as: (1) NNs with lumped dynamics; (2) recurrent NNs; and (3) dynamic multilayer percetron [236].

NNs with lumped dynamics are static networks with delayed measurements of the process inputs and outputs. Accordingly, the network approximate the current process output $y(k)$ as a function of the delayed measurements:

$$\begin{aligned} y(k) = f(&u_1(kT), u_1((k-1)T), \ldots u_1((k-m)T), \\ &u_2(kT), u_2((k-1)T), \ldots u_2((k-m)T), \\ &\ldots \\ &u_p(kT), u_p((k-1)T), \ldots u_p((k-m)T) \\ &\quad y((k-1)T), \ldots y((k-n)T)) \end{aligned} \tag{9.10}$$

where y - output variable

$u_1, u_2, \ldots u_p$ - in total there are p input variables

T - period or time step for sampling training data

k - the current sampling point

m - the $(k\text{-}m)$th sampling point for inputs

n - the $(k\text{-}n)$th sampling point for the output

It is clear that NNs with lumped dynamics do not change the internal structure of a static NN. Dynamics is captured by sampling data from input-output dynamic trends. Consequently, the dimension of the NN input space increases according to the number of delayed measurements used.

Bhat and McAvoy [237] described such a case study using a stirred tank reactor as shown in Figure 9.6. The dynamics of the system can be described by two linear differential equations and three nonlinear algebraic equations. Training data were generated using dynamic simulation by forcing the F_2 stream with a 2% PRBS signal super imposed upon its steady state value. The F_2 and pH responses used for training are shown in Figure 9.7. A FFNN neural network was developed to use the past five pH and F_2 values as well as five future F_2 values to predict the future pH values. The time step T is 0.4 min. Since the data is actually historical data generated through simulation, the future F_2 values are available. The size of the network is therefore fifteen inputs and five outputs. The number of hidden neurons was five.

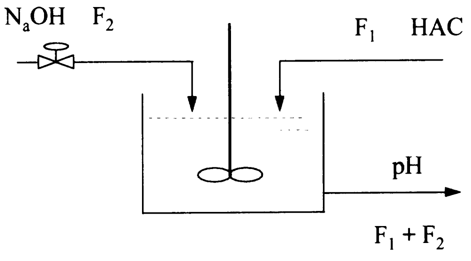


Figure 9.6 pH CSTR.

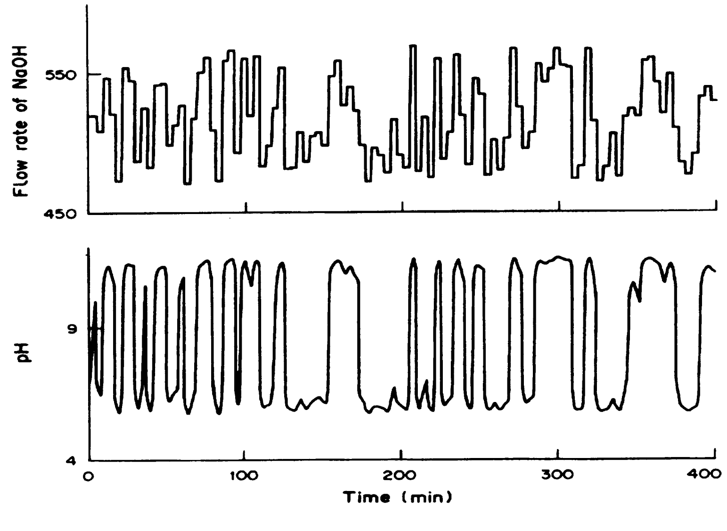


Figure 9.7 Data used for training the CSTR neural network.

A moving window approach is used. At the beginning of the training process the window is placed at the beginning of the database. The first five pH and ten F_2 are used as inputs and the next five pH are used as the outputs. After the first presentation of the data, the window moves T down the database. Again the first five pH and ten F_2 in the window are inputs to the net and the next five pH in the window are outputs. This process is continued until the end of the database is reached and, the process is repeated by starting at the beginning of the database, until the training is terminated through either reaching the required error or the maximum number of iteration.

Recurrent neural networks (RNN) (e.g., Shaw et al. [238]) dispense with the explicit use of lagged measurements at the network input. Instead, RNN posses a long-term memory through recurrent connections within the network. However, according to Ayoubi [236], the application of RNN was minimised to sequence processing, yet there are no significant applications to model identification of real-world systems.

An alternative method is the multilayer dynamic percetron [236, 239] which modifies the neuron processing and interconnections in order to incorporate dynamics inherently within the network.

9.6 Summary

It is important to have an appropriate mechanism to test the reliability of software sensor models developed using FFNN in both model development and maintenance stages. This is complicated by the high dimensionality and large volume of data. The integral framework described in Sections 9.2 and 9.3 uses Bayesian automatic classification to cluster the multivariate data into classes having similar features. Test data for FFNN model development are then selected from each of the classes. When new data become available, it is possible to tell if they represent new information which makes it possible to determine when the neural network needs to be retrained. Through the progressive improvement of the model, it is found that the confidence of the model for predicting new data is lower if more data are grouped into new classes. For example, all patterns in dataset-2 have been grouped into new classes, so the confidence in predicting dataset-2 using Model-1 is 25.0%; while some patterns in dataset-3 are classified into old classes, where the confidence of predicting dataset-3 using Model-2 is 43.8%.

Due to inadequate knowledge of the domain problem, people tend to use many input variables. Inclusion of irrelevant and redundant variables may deteriorate the model performance because it increases the size of the net, leading to longer training time, and requires more training data, and reduces model generalisation capability. The several approaches introduced in this chapter, including PCA, sensitivity analysis and weight matrix analysis can be used to combat the problem. However there is still the need to carry out further investigation in these approaches, especially in solving practical problems.

Only a very brief introduction has been given to dynamic FFNNs for software sensor design. Due to its great application potential, there has been a growing interest in dynamic neural networks. Apart from the need to develop the most appropriate network structure and learning approaches, there is clearly a practical difficulty in applying dynamic neural networks, that is, how to get the large number of data patterns needed. Experimenting on industrial processes is time consuming and expensive because it may cause the process to deviate from desired conditions. There is another problem associated with doing experiment on industrial processes, which is that the processes are often under closed loop control using linear models. Kim et al. [240] addressed the need to consider the reality that most processes are under linearized control scheme.

There are many other issues associated with determination of FFNN network topology and training. A detailed discussion on these has been given in Chapter 5 therefore they are not repeated in this chapter.

CHAPTER 10
CONCLUDING REMARKS

In this monograph we have sought to provide an introduction to automatic analysis and interpretation of process operational data both in real time and over the operating history. Methods are developed for designing intelligent, state space based systems for process monitoring, control and diagnosis. Such a system is able to identify known and new operational states, either normal or abnormal and project the operation of the process to a single point of the operational state space by simultaneously considering all measurements and giving causal explanations to operators and plant managers. The techniques have also proved useful in discovering operational states for product design. In developing the methods, we have attempted to address the point that plant operators and supervisors are part of the overall control system responsible for data interpretation and critical decision making, and therefore should be integrated into control systems in a way to provide them with necessary computer based, automatic processing tools.

It is believed that the problems addressed in this book which have not been fully studied before are important in process monitoring and control, and we hope readers find the methods of approaching the problems both interesting and practical. Many of the methods introduced in this book are new, such as feature extraction using wavelets and conceptual clustering using inductive learning. Some techniques are applications of traditional methods, such as principal component analysis. Even for well developed methods such as feedforward neural networks we have tried not to repeat the algorithms. Instead we present the methods in a novel way that summarises experience and addresses the various necessary considerations, such as training and test data selection and sensitivity study in using neural networks.

Examples and industrial case studies in varying degrees of complexity have been used to illustrate these methods. They include a continuous stirred tank reactor, the reaction regeneration system of a fluid catalytic cracking process, a methyl tertiary butyl ether unit, the reaction fractionation system of fluid catalytic cracking and a waste water treatment plant. Many more examples could have been added, but we feel these should be sufficient to demonstrate the effectiveness of the methods and aid in clarifying the ideas and concepts.

We have intended to provide a new way of thinking in the approach to designing and operating process control systems and introduce practical methodologies for implementation. We also hope that the book will provide a stimulus to researchers since the field is still in its infancy. Many more challenges than those addressed in this book need to be considered in order to develop effective, robust and practical systems for knowledge discovery and for designing intelligent and state space based monitoring and control environment.

Suggestions for future work have been summarised at the end of each chapter and will therefore not be repeated here. Most of the case studies in the book are with respect to continuos processes save a case study described in Chapter 4. Batch processes are expected to provide more challenges due to the distinctive features of batch operation mode. Batch processes do not have steady state operations and are more flexible. They also provide great opportunities for application of the techniques discussed in this book. For example, wavelets are potentially a powerful technique for dealing with signals of batch operations because they often change at higher frequencies than in continuous processes. In addition, feedstock disturbances occur frequently and on-line measurements of product quality variables are not available. As a result, most batch processes have not been able to achieve tight quality control. Knowledge discovery approaches are attractive for dealing with these kind of problems because of difficulties associated with developing accurate process models from first principles [241]. At present, industrialised countries are shifting the focus from large-scale commodity production to smaller scale fine chemicals, pharmaceutical and food production, therefore, there is a clear need for advances in this field.

APPENDIX A
THE CONTINUOUS STIRRED TANK REACTOR (CSTR)

A single, non-isothermal continuous stirred-tank chemical reactor (CSTR) is shown in Figure A1.

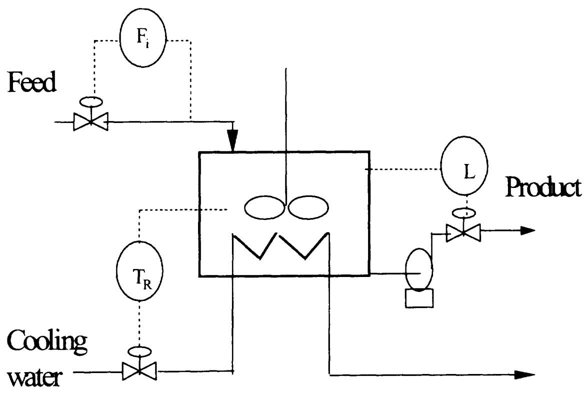


Figure A1 The CSTR reactor.

A single reaction $A \rightarrow B$ takes place in the reactor. It is assumed that the tank is very well mixed therefore the concentration of C_0 of component A and the temperature of the product stream leaving the tank are equal to that in the tank. The reaction is exothermic and reaction temperature T_R is automatically controlled by a PID controller through manipulating the cooling water flowrate. There are also feed flowrate and liquid level controllers as shown in Figure A1. Apart from the three controllers, the dynamic behaviour of the process is described by the following equations:

Component mass balance

$$V\frac{dC_o}{dt} = F_i C_i - F_o C_o - V K_o e^{-E/RT_R} C_o \tag{A1}$$

Energy balance

$$V\rho C_p \frac{dT_R}{dt} = \rho C_p F_i T_i - \rho C_p F_o T_R - \frac{a F_c^{b+1}}{F_C + \dfrac{a F_c^b}{2\rho_c C_{pc}}}(T_R - T_{ci}) + (-\Delta H_{rxn}) V K_o e^{-E/RT_R} C_o \tag{A2}$$

Total mass balance

$$\frac{dV}{dt} = F_i - F_o \tag{A3}$$

V - holdup of the reaction mixture, m^3. The volume of the tank is 1 m^3.

L - Liquid level in the tank, 0 ~ 100.

C_o - Concentration of component A in the product stream leaving the tank, kmol/m^3, which is equal to that in the tank.

C_i - Concentration of component A in the inlet feed stream, kmol/m^3.

F_i - Feed flowrate, m^3/min.

F_o - Product stream flowrate, m^3/min.

F_c - Cooling water flowrate, m^3/min.

K_o - Reaction coefficient, K_o = 1.0 x 10^{10} min^{-1}.

E/R - 8330.1 K^{-1}.

ρ - Density of reaction mixture, ρ = 10^6 g/m^3.

ρ_c - Density of cooling water, ρ_c = 10^6 g/m^3.

C_p - Specific heat capacity of the reaction mixture, C_p = 1 cal/m^3.

C_{pc} - Specific heat capacity of cooling water, C_p = 1 cal/m^3.

$-\Delta H_{rxn}$ = 130 x 10^6 (cal/kmol) a - Constant, a = 1.678 x 10^6 (cal/min)/K.

b - Constant, b = 0.5

T_i - Temperature of the inlet feed, K.

T_R - Temperature of the reaction mixture, K.

T_{cin} - Inlet temperature of cooling water, K.

More detailed description of the process can be found in [8]. A dynamic simulator was developed for the CSTR which has included three controllers as shown in Figure A1. The simulator has a MS Windows based graphical interface which allows users to make any changes and faults and disturbances can be easily introduced. To generate a data set or data case, run the simulator at steady state and introduce a disturbance or fault and at the same time start to record the dynamic responses. Eighty five data sets were generated which are summarised in Table A1. For each data set, eight variables were recorded, including F_i, T_i, C_i, T_{wi}, F_w, TR, C_O and L. In each data set, each variable was recorded as a dynamic trend consisting of 150 sampling points. Therefore for each variable the data size is a matrix 85 (the number of data sets) × 150 (the number data points representing a dynamic trend).

Table A1 Data cases generated.

Data sets	Data detail
1~11	All control loops are at AUTO mode and S.P. of T_R = 350K. Change Ti (K) from 343 → 333, 343 → 323, 323 → 343, 343 → 353, 353 → 343, 343 → 363, 363 → 343, 310 → 303, 303 → 313, 310 → 293, 293 → 313
12~15	All control loops are at AUTO mode and S.P. of T_R = 350K. Change Ci (kmol/m^3) from 2.0 → 1.6, 1.6 → 2.0, 2.0 → 1.2, 1.2 → 2.0
16~20	All control loops are at AUTO mode and S.P. of T_R = 350K. Change Fi (m^3 /min) from 1.00 → 1.06, 1.06 → 1.02, 1.02 → 0.94, 1.00 → 0.62, 0.62 → 1.00
21~24	All control loops are at AUTO mode and S.P. of T_R = 350K. Change L (%) from 50.0 → 60.0, 60.0 → 50.0, 50.0 → 40.0, 40.0 → 50.0
25~31	All control loops are at AUTO mode and S.P. of T_R = 405 K. Change Ti (K) from 343 → 333, 333 → 323, 323 → 333, 343 → 353, 353 → 343, 343 → 363, 363 → 343
32~36	All control loops are at AUTO mode and S.P. of T_R = 405 K. Change Twi (K) from 310 → 303, 310 → 293, 293 → 313, 313 → 323, 323 → 313

Table A1 Data cases generated (continued).

Data sets	**Data detail**
37~39	All control loops are at AUTO mode and S.P. of T_R = 405 K. Change Ci (kmol/m^3) from 2.0 → 1.6, 1.6 → 2.0, 2.0 → 1.2
40~43	All control loops are at AUTO mode and S.P. of T_R = 405 K. Change Fi (m^3 /min) from 1.00→ 1.06, 1.06→ 0.98, 0.98→ 0.86, 0.86→ 1.02
44~46	All control loops are at AUTO mode and S.P. of T_R = 405 K. Change L (%) from 50.0→ 60.0, 60.0→ 50.0, 50.0→ 40.0
46~52	All control loops are at AUTO mode and S.P. of T_R = 380 K. Change Ti (K) from 343→333, 333 → 323, 323 → 333, 333 → 343, 343 → 353, 353 → 343
53~56	All control loops are at AUTO mode and S.P. of T_R = 380 K. Change Twi (K) from 310 → 303, 303 → 313, 310 → 323, 323 → 313
57~60	All control loops are at AUTO mode and S.P. of T_R = 380 K. Change Ci (kmol/m^3) from 2.0 → 1.6, 1.6 → 2.0, 2.0 → 1.2, 1.2 → 2.0
61~66	All control loops are at AUTO mode and S.P. of T_R = 380 K. Change Fi (m^3 /min) from 1.00→ 1.06, 1.06→ 1.00, 1.00→ 1.10, 1.10→ 1.00, 1.00→ 0.90, 0.90→ 1.00
67~70	All control loops are at AUTO mode and S.P. of T_R = 380 K. Change the S.P. of L (%) from 50.0→ 40.0, 40.0→ 50.0, 50.0→ 60.0, 60.0→ 50.0
71~80	All control loops are at AUTO mode and S.P. of T_R = 380 K. Change the output of the CSTR level controller from (5) 50.0→ 10.0, 50.0→ 8.0, 50.0→ 6.0, 50.0→ 12.0, 50.0→ 14.0, 50.0→15.0, 50.0→ 5.0, 50.0→ 2.0, 50.0→ 3.0, 50.0→ 0.0
81~85	All control loops are at AUTO mode. Change the output of the controller T_R from (%) 71.4→ 20.0, 71.4→ 21.0, 71.4→ 22.0, 71.4→ 19.0, 71.4→18.0

APPENDIX B
THE RESDIUE FLUID CATALYTIC CRACKING (R-FCC) PROCESS

Fluid catalytic cracking process (FCC) is a dominant feature of most refinery operations, representing the order of thirty percent in product value. FCC is also a very complicated process featured by highly non-linear dynamics due to the strong interactions between the riser tube reactor and fluidised bed regenerator through heat, mass and pressure balances.

The residual fluid catalytic cracking (R-FCC), as shown in Figure B1 is used as case studies in Chapters 5 and 6. It converts heavy bottoms of the crude and vacuum distillation columns into more valuable gasoline and lighter products. Preheated feed is mixed with high temperature slurry recycle, which comes from the bottom of the main fractionator, and injected into the riser tube reactor, where it is mixed with high temperature regenerated catalyst and totally vaporises. The high temperature regenerated catalyst provides the sensible heat, heat of vaporisation and heat of reaction necessary for the endothermic cracking reactions which complete in a few seconds. As a result of the cracking reactions, a carbonaceous material – coke is deposited on the surface of the catalyst. As a result continuous regeneration is required.

Separation of catalyst and gas occurs in the disengaging zone of the reactor. Entrained catalyst is stopped by the cyclones. Catalyst is returned to the stripping section of the reactor where steam is injected to remove entrained hydrocarbons. Reactor product gas is channelled to the main fractionator for heat recovery and separation into various product streams. Wet gas from the overheads of the main fractionator (C_6 and lighter) is compressed for further separation in downstream fractionators.

Spent catalyst is transported from the reactor to the regenerator through the spent catalyst pipe. Air is injected into the bottom of the regenerator lift pipe to assist the circulation of catalyst.

Catalyst in the regenerator is fluidised with air which is provided by the lift and combustion air blowers. Carbon and hydrogen on the catalyst react with oxygen to produce carbon monoxide, carbon dioxide and water. While most of the reactions occur in the fluidised bed, some reaction does occur in the disengaging section above the bed, where some catalyst is still present. Gas travels up the regenerator into the cyclones where entrained catalyst is removed and returned to the bed.

The regenerator is run at a high temperature and an excess of oxygen to ensure that virtually all carbon monoxide produced in the bed is converted to carbon dioxide before entering the cyclones. Because this is a residual FCC process processing heavy feed, the amount of heat generated through burning coke in the regenerator is more than that required by the endorthemic cracking reactions. Therefore it has internal and external heat exchangers to remove the excess heat.

Regenerated catalyst flows over a weir into the regenerator standpipe. The head produced by catalyst in the standpipe provides the driving force for catalyst through the regenerated catalyst pipe to the riser tube reactor.

The major control loops for the process are summarised in Table B1. They include reaction temperature, pressures of the two reactor vessels as well as flowrates of feed, air and steam as well as catalyst hold-up. A very important feature of the process is the very complicated and dynamic interactions between the heat, mass and pressure balances between the two reactor vessels, and fluidisation conditions. On top of the control loops, safety guard systems play a critical role in preventing disastrous situations. There are four major safety guard systems as shown in Table B1. They all need the authorisation of operators by pressing one or two buttons to prevent system over reactions. When a safety guard system is activated, in total fourteen valves will act to close, open or maintain at the original position according to predefined logic.

A dynamic simulator for training operators was developed in 1992 by one of the author, Wang for the refinery. The simulator has a number of features compared with some other FCC simulators. First, it is able to simulate continuously and smoothly the start-up and shutdown procedures and normal operation. In addition, it has a very good operability. All the operating variables and equipment units are operable within a wide range, including conditions regarded as abnormal. Furthermore apart from faults that can be initiated randomly, there are also extra twenty faults that are common to FCC processes. More importantly, it has high fidelity: the simulator has been tested in the refinery for operator training for several years and during the process the simulator has been continuously improved. There are other features including intelligent on-line support, scoring and snapshot.

In this study, sixty four data patterns generated from the simulator was used which are summarised in Table B2. We limit our discussion to 64 data patterns in order to simplify the discussion and result presentation. The data sets include faults or disturbances:

- fresh flow rate increased and decreased
- preheat temperature of mixed feed increased and decreased
- recycle slurry flow rate increased and decreased
- opening rate of hand valve V20 increased and decreased
- air flow rate increased and decreased
- valve 401-ST opening from 100% decreased
- cooling water pump failure
- compressor failure
- double faults

The sixty four data patterns were first generated by the customised dynamic training simulator, then random noises were added to the data using a noise generator of MATLAB®, before it is fed to the system.

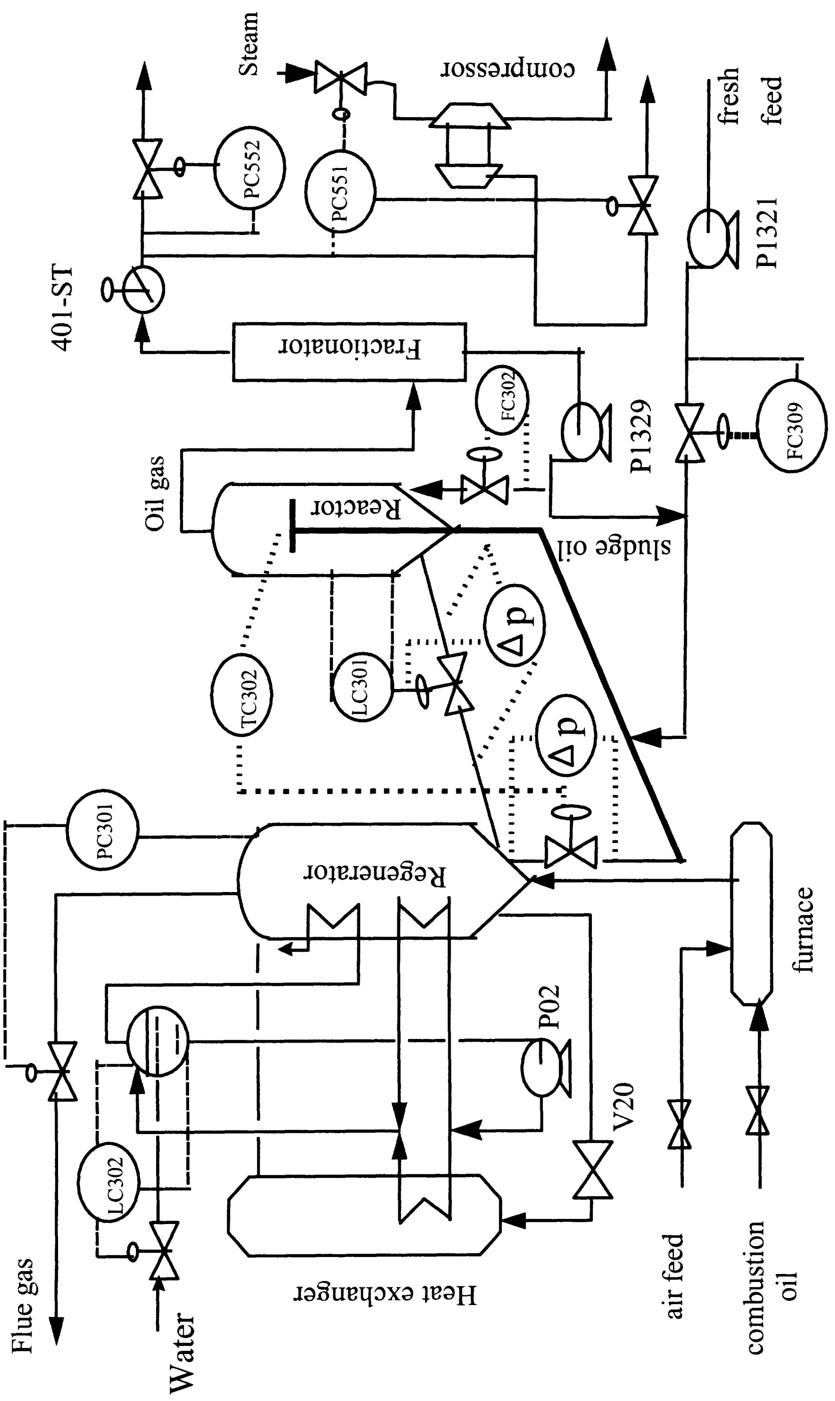


Figure B1 The simplified flowsheet of the R-FCC process.

Table B1 The major control loops and safety guard systems of the R-FCC process.

Major control loops		
Reaction temperature	TC302	Temperature control at the exit of the riser tube reactor. Set pint, 513^{0}C. Manipulated variable is regenerated catalyst flow to the riser tube reactor
Regenerator Pressure	PC301	Pressure control of the regenerator at 1.6 kg/cm^2. Manipulated variable is the flue gas flowrate
Inlet pressure of the compressor	PC551	Pressure control. This is to maintain a pressure difference of > 0.3 kg/ cm^2 between the regenerator and the reactor. The manipulated variable is normally the compressor speed, but can also be switched to value on top of the fractionator, especially during start-up
Other controllers		Feed flowrates FC309 and FC305, Steam flowrate F302, water level LC302, catalyst holdup controller LC301
Safeguards		
Low feed flowrate		At low feed flowrate, reactions will deteriorate with increased reaction depth and secondary reactions.
Low flow of main air supply		Low flow of main air supply causes increase of regenerator temperature, poor fluidisation and even cause reverse flow of catalyst or catalyst flowing to the air compressor
Low pressure difference between the regenerator and the reactor		A pressure difference of 0.3 kg/ cm^2 between the regenerator and the reactor is normally required to prevent gas oil flowing into the regenerator because this may cause possible very high temperature in the regenerator and damage. This safety guard will induce the low feed safety guard
External heat exchanging system		When the low main air supply safety guard is activated the external heat exchanging system is also needed to be cut off

Table B2 Summary of the data patterns studied.

Data patterns	**Fault mode**	**when time t < 0 operation is at steady state, at t = 0 make the following step change**
1~9	**1**	Fresh feed increased by 10 20 30 40 50 60 70 80 90%
10~18	**2**	Fresh feed decreased by 10 20 30 40 50 60 70 80 90%
19~22	**3**	Preheat Temperature T of mixed feed increased by 5 10 15 20 °C
23~24	**4**	Preheat Temperature of mixed feed decreased by 15 10°C
25~26	**5**	Recycle oil Flow rate F increased 70 90%
27~28	**6**	Recycle oil Flow rate F decreased 70 90%
29~32	**7**	Opening ratio of hand-valve V20 increased by 5 10 15 24%
33~37	**8**	Opening ratio of hand-valve V20 decreased by 15 25 35 55 10%
38	**9**	Cooling water pump P-02 failure
39~43	**10**	Air flow rate increased by 6.5 11.5 15 31.5 40.5%
44~49	**11**	Air flow rate decreased by 3.5 8.5 28.5 38.5 48.5 53.5%
50	**12**	Compressor failure
51~57	**13**	Valve 401-ST opening from 100% decreased by 10 20 40 45 60 80 90%
58	**1**	Fresh feed Flow rate F increased by 65%
59	**2**	Fresh feed Flow rate F decreased by 85%
60	**1&8**	Fresh feed Flow rate F increased 65% and V20 opening rate decreased 55%
61	**1&9**	Air flow rate increased 9.5% and compressor failure
62	**2&9**	Air flow rate increased 9.5% and valve 401-ST opening decreased 35%
63	**10&12**	Pump P-02 failure and compressor failure
64	**10&13**	Fresh feed decreased 70% and valve 401-ST opening decreased 20%

APPENDIX C
THE METHYL TERTIARY BUTYL ETHER (MTBE) PROCESS

The refinery methyl tertiary butyl ether (MTBE) process, shown in Figure C1 is used in Chapter 7 as a case study of applying conceptual clustering to developing process monitoring systems. MTBE is an important industrial chemical because large quantities of MTBE are now required as an octane booster in gasoline to replace tetra ethyl lead. MTBE is produced from the reaction between isobutene and methanol in a reactor and a catalytic distillation column packed with catalysts. The reaction takes place at a temperature of 35~75^{0}C and a pressure of 0.7~0.9 MPa. The main chemical reaction is:

methanol + isobutene === methy tertiary butyl ether

$CH_3OH + (CH_3)_2C{=}CH_2 === (CH_3)_3COCH_3$

The reaction occurs in the liquid phase and is reversible and exothermic. Proper selection of the reaction pressure allows part of the reaction product to be vaporised, absorbing part of the heat of reaction. Thus the temperature in the reactor could be controlled.

An unavoidable side reaction is:

isobutene + isobutene === diisobutene

$(CH_3)_2C{=}CH_2 + (CH_3)_2C{=}CH_2 === \{(CH_3)_2C{=}CH_2\}_2$

However, the selectivity of the primary reaction is about 98~99%. Since the product of the side of reaction is very limited it can be ignored.

MTBE is mostly produced in the reactor, and a small amount is produced in the catalytic distillation column which combines reaction and distillation. The rest of the isobutene and methanol from the reactor passes to the catalytic distillation column and is reacted. The MTBE produced is then separated simultaneously, which ensures a high conversion level of isobutene in the catalytic distillation column.

The rest of methanol after the reaction is recovered from the distillate of the top of the column by water extraction and a conventional column. The solvability of methanol in C_4 hydrocarbon and in water is quite different so C_4 hydrocarbon can be easily separated from methanol water solution. The recovered methanol and C_4 hydrocarbon are then recycled.

The process used in this study comprises a catalytic reactor R201, catalytic distillation column C201, water extraction column C202, methanol distillation column C203 and several other vessels together with pumps. The overview of the flowsheet of the process is shown in Figure C1. Methanol from D202 and C_4 from D201 are mixed in M201 and then pass to the reactor R201. An on-line analyser measures the ratio of isobutene and methanol and the inlet temperature of the reactor is controlled by means of the heat exchanger E201. Maintaining the pressure in the reactor is important so as to control the amount of product vaporised to absorb the heat of reaction. To achieve a high conversion level of isobutene an additional methanol feed stream is introduced into the catalytic distillation column at the top of the column. The vessel D211 is filled with catalysts and is used to filter methanol before it enters the column. The MTBE is taken from the bottom of the column and the distillate is the mixture of methanol and the C_4 hydrocarbon. The pressure controller at the top of the column is also used to control the pressure of the upstream reactor. Part of the methanol and C_4 hydrocarbon is returned to he catalytic distillation column as reflux. The rest of the mixture passes to the water extraction column C202 to be separated into unreacted C_4 hydrocarbon which is taken off from the top and the methanol water solution from the bottom.

The case study is based on data generated from a customised dynamic simulator used for training operators in start-up and shutdown procedures, emergency management as well as normal operation monitoring and control of a commercial MTBE process. This simulator has also been used to study the start-up procedures and has been validated over several years of usage for various normal and abnormal conditions. Disturbances as well as faults concerned with various equipment items can be easily introduced to the simulator and corresponding responses of variables displayed in trend groups. This study comprises 100 tests, 64 of which are normal or with small disturbances and 36 have significant upsets or faults, as summarised in Table C1. For each test, 21 variables were recorded as indicated in Table C2. For each variable, 256 data points are recorded following a disturbance or fault. So the data has a dimension of 21 x 256. Random noise was added to the data. Figure C2 shows such an example.

Only the sections of feed, reactor and reactive distillation column of the process have been considered, which means that columns C202 and C203 have not been included in the study apart from the fact that there is a small recycle stream of the recovered methanol which affects the feed tank D202. This recycle stream was fixed at the design flowrate.

Table C2 The variables recorded as dynamic responses.

No.	Variable	Description
1	F_D201_in	Inlet Flow of D201
2	F_D201_out	Outlet Flow of D201
3	L_D201	Liquid Level of D201
4	F_D202_in1	Adding methanol Flow of D202
5	F_D202_out1	Methanol flow from D202 to R201
6	F_D202_out2	Methanol flow from D202 to D211
7	L_D202	Liquid Level of D202
8	T_E201_out	Outlet Temperature of E201
9	F_E201_steam	Steam flow through E201
10	T_R201_top	Top Temperature of reactor R201
11	T_R201_mid	Middle Temperature of reactor R201
12	T_R201_bot	Bottom Temperature of reactor R201
13	F_C201_ref	Reflex flow of C201
14	F_C201_out	Product Flow of MTBE
15	T_C201_top	Top Temperature of C201
16	T_C201_mid	Middle Temperature of C201
17	T_C201_bot	Bottom Temperature of C201
18	T_MTBE	MTBE Temperature
19	L_C201	Liquid Level of C201
20	F_D203_out	Liquid Flow from D203 to C202
21	L_D203	Liquid Level of D203

Table C1 Data cases used for the study.

Cases	Case Detail		Case Description
1	Control valve of LC201D	50% → 0%	Liquid level control of C4 hydrocarbons tank D201
2	Pump P201	Failure	Pump after the D201
3	Switch valve SW3	On → Off	Switch valve after P201, on the feed stream of C4 hydrocarbons
4	Pump P202	Failure	Pump after D202, on the feed stream of methanol
5	Switch valve SW2	On → Off	Switch valve after P202, on the feed stream of methanol
6	Manual valve HC202D	20%→ 0%.	Manual valve on the feed of methanol to tank D202
7	Control valve of FC202D	33%→ 13%	Feed flowrate controller of methanol
8	Control valve of FC202D	33%→ 26%	… …
9	Control valve of FC202D	33% → 46%	… …
10	Control valve of FC202D	33% → 59%	… …
11	Control valve of FC202D	33% → 3%	… …
12	Pump P208	Failure	
13	Switch valve SW1	On → Off	Switch valve on the feed stream of methanol directly to column
14	Manual valve HC211D	18% → 40%	Manual valve on the feed stream of methanol directly to column
15	Manual valve HC211D	18% → 60%	… …
16	Manual valve HC211D	18% → 100%	… …
17	Manual valve HC211D	18% → 0%	… …
18	Manual valve HC211D	18% → 10%	… …
19	Control valve of TC201R	39% → 30%	Bottom T controller of the reactive distillation column C201
20	Control valve of TC201R	39% → 20%	… …
21	Control valve of TC201R	39% → 10%	… …
22	Control valve of TC201R	39% → 0%	… …
23	Control valve of FC203E	37% → 33%	Steam flowrate controller at the bottom of the column C201
24	Control valve of FC203E	37% → 29%	… …
25	Control valve of FC203E	37% → 16%	… …
26	Control valve of FC203E	37% → 6%	… …
27	Control valve of FC203E	37% → 0%	… …
28	Control valve of FC203E	37% → 57%	… …
29	Control valve of FC201C	40% → 30%	Reflux F controller of the reactive distillation column C201
30	Control valve of FC201C	40% → 20%	… …
31	Control valve of FC201C	40% → 10%	… …
32	Control valve of FC201C	40% → 0%	… …
33	Control valve of FC202D	33% → 37%	Feed flowrate controller of methanol
34	Control valve of FC202D	33% → 40%	Feed flowrate controller of methanol
35	Control valve of FC202D	33% → 50%	Feed flowrate controller of methanol
36	Control valve of FC202D	33% → 55%	Feed flowrate controller of methanol
37 - 100		Normal	Normal Operation

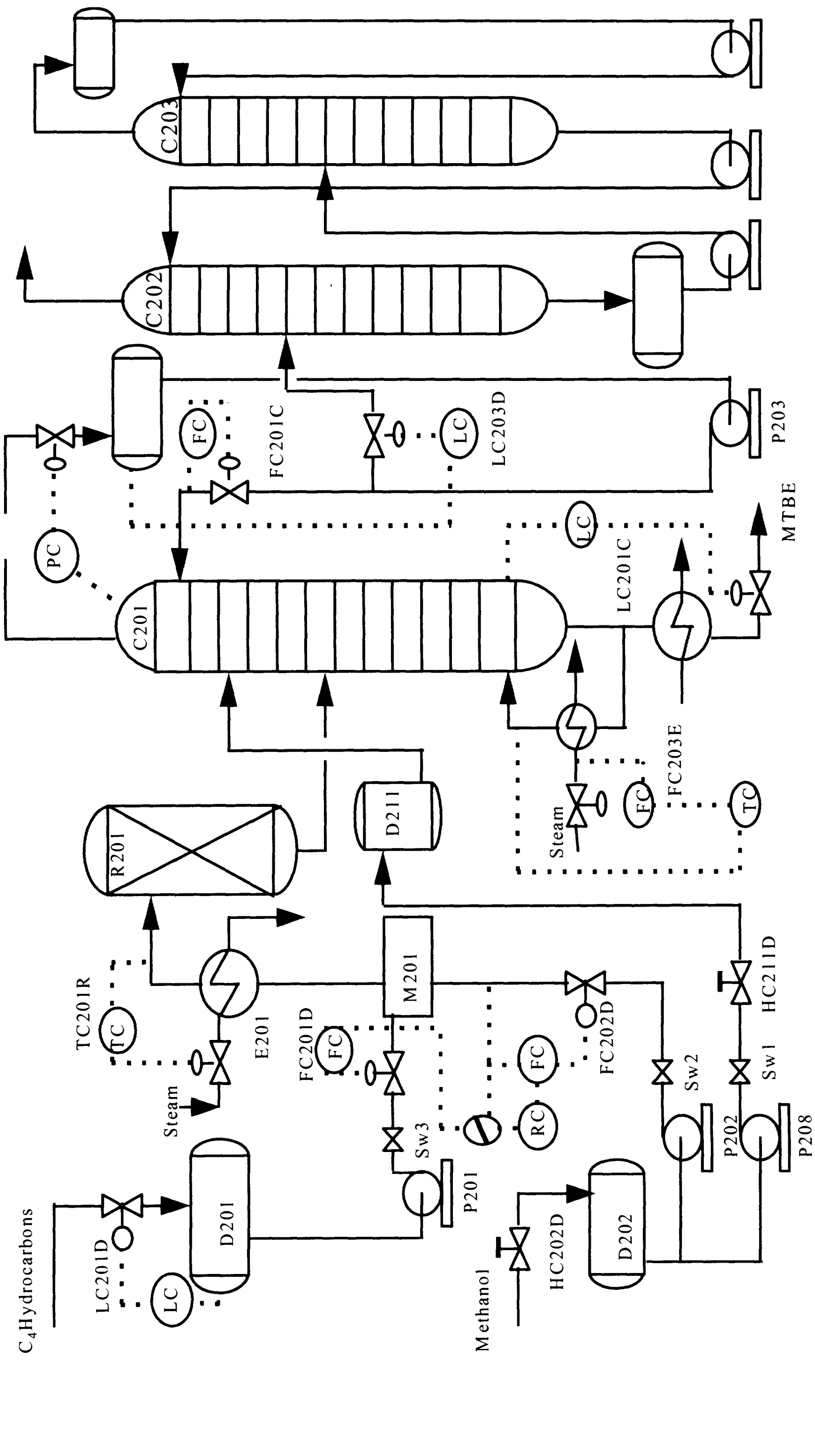


Figure C1 The simplified flowsheet of the MTBE process.

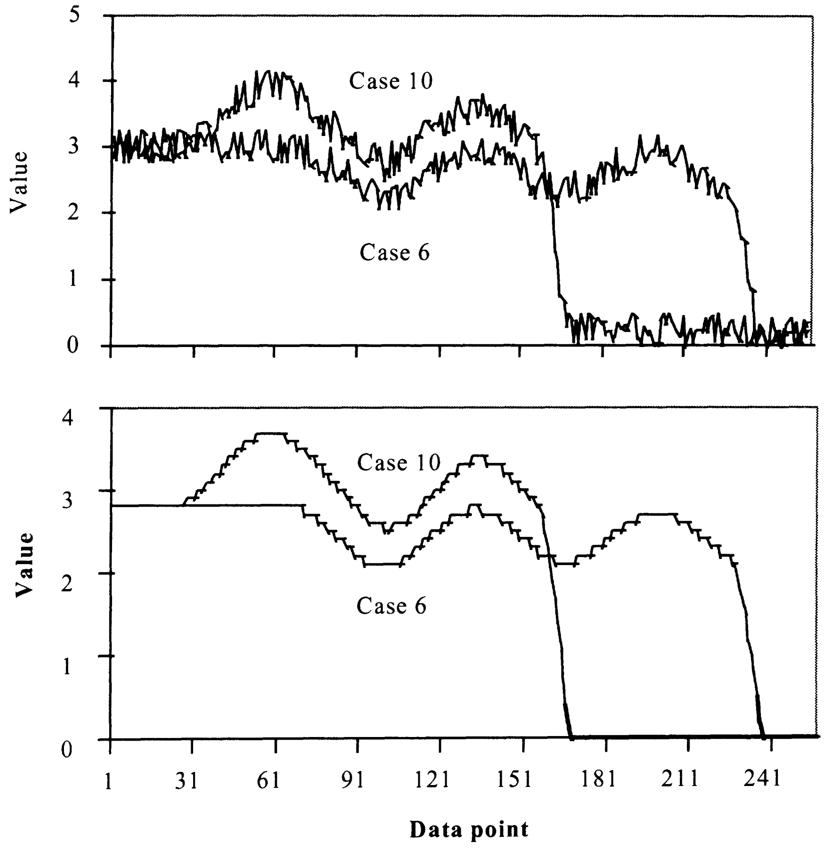


Figure C2 Two dynamic trends with and without noise.

REFERENCES

1 Lukas MP. Distributed control systems: their evaluation and design. Van Nostrand Reinhold Company, New York Wokingham, 1986.

2 Yamanaka F and Nishiya T. Application of the intelligent alarm system for the plant operation. Comput Chem Engng 1997; 21: s625-s630.

3 Howat CS. Analysis of plant performance. In: Perry RH, Green DW (eds.) Perry's chemical engineers' handbook, Section 30. McGraw-Hill, 1997, 30-1 - 30-36.

4 Saraiva PM and Stephanopoulos G. Continuous process improvement through inductive and analogical learning. AIChE J. 1992; 38: 161-183.

5 Saraiva, PM. Inductive and analogical learning: data-driven improvement of process operations. In: Stephanopoulos G and Han C (Eds) Intelligent Systems in Process Engineering: Paradigms from Design and Operations. Academic Press, Inc., San Diego, California, 1996, 377-435.

6 Taylor W. What every engineer should know about artificial intelligence. MIT Press, Cambridge, MA, 1989.

7 Shewhart WA. Economic control of quality of manufactured product. Van Nostrand, Princeton, N.J., 1931.

8 Marlin TE. Process control - designing processes and control systems for dynamic performance. McGraw-Hill, 1995.

9 Oakland JS. Statistical process control: a practical guide. Heinemann, London, 1986.

10 Woodward RH, Goldsmith PL. Cumulative sum techniques. Oliver and Boyd,, London, 1964.

11 Roberts SW. Control charts tests based on geometric moving averages. Technometrics 1959; 1: 239-250.

12 Hunter JS. The exponentially weighted moving average. J of Quality Technology 1986; 18:203-210.

13 Stephanopoulos G. Chemical process control: an introduction to theory and practice. Prentice-Hall, Inc., New Jersey, 1984.

14 Bekiaris N, Morari M. Multiple steady states in distillation: infinity/infinity predictions, extensions, and implications for design, synthesis, and simulation. Ind. Eng. Chem. Res. 1996; 35: 4264-4280.

15 Schrans S, DeWolf S, Baur R. Dynamic simulation of reactive distillation: an MTBE case study. Comput. Chem Engng, 1996; 20: S1619- S1624.

16 Arandes JM, Delasa HI. Simulation and multiplicity of steady-states in fluidised FCCUs. Chem. Eng. Sci., 1992; 47: 2535- 2540.

17 Hangos KM, Perkins JD. Structural stability of chemical process plants. AIChE J., 1997; 43: 1511-1518.

18 Quinlan JR. C4.5: programs for machine learning. Morgan Kauffman, 1993.

19 Quinlan JR. Induction of decision trees. Machine learning 1986; 1: 81-106.

20 Quinlan JR. http://www.rulequest.com/, 1999

21 Chen FZ, Wang XZ. An integrated data mining system and its application to process operational data analysis. Comput. Chem. Engng 1999; 23(suppl.): s777-s780.

22 Clarke-Pringle T, MacGregor JF. Product quality control in reduced dimensional spaces. Ind. Eng. Chem. Res. 1998; 37:3992-4002.

23 Fayyad UM, Simoudis E. Data mining and knowledge discovery. Tutorial Notes at PADD'97 - 1st Int. Conf. Prac. App. KDD & Data Mining, London, 1997.

24 Massey J, Newing R. Trouble in mind. Computing 1994; May: 44-45.

25 Fayyad UM, Piatetsky-Shapiro G, Smyth P. From data mining to knowledge discovery: an overview. In: Fayyad UM, Piatetsky-Shapiro G, Smyth P, Uthurusamy R (eds.) Advances in Knowledge Discovery and Data Mining. AAAI Press/MIT Press, 1996, 1-36.

26 Glymour C, Madigan D, Pregibon D, Smyth P. Statistical themes and lessons for data mining. Data Mining and Knowledge Discovery 1997; 1: 11-28.

27 Elder IV J, Pregibon D, A statistical perspective on knowledge discovery in databases. In: Fayyad UM, Piatetsky-Shapiro G, Smyth P, Uthurusamy R (eds.) Advances in Knowledge Discovery and Data Mining. AAAI Press/MIT Press, 1996, 83-116.

28 Chen MS, Han JW, Yu PS. Data mining: an overview from a database perspective. IEEE Trans. Know. Data Eng. 1996, 8: 866-883.

29 Imielinski T, Mannila H. A database perspective on knowledge discovery. Comm. of ACM 1996; 39(11): 58-64.

30 Inmon WH. The data warehouse and data mining. Comm. of the ACM 1996; 39(11): 49-50.

31 Shortland R, Scarfe R. Data mining applications in BT. BT Technology J. 1994; 12(4): 17-22.

32 Brachman RJ, Khabaza T, Kloesgen W, Piatesky-shapiro G, Simoudis E. Mining business databases. Comm. of ACM 1996; 39(11):42-48.

33 Fayyad U, Piatetsky-Shapiro G, Smyth P. The KDD process for extracting useful knowledge from volumes of data. Comm. of the ACM 1996; 39(11): 27-34.

34 Meggs G, Fagan M. Knowledge discovery – practical methodology and case studies. Tutorial Notes in *PADD98* - 2nd Int. Conf. Prac. Appl. Knowl. Disc. Data Mining. London, 1998.

35 Fayyad U, Haussler D, Stolorz P. Mining scientific data. Comm. of the ACM 1996; 39(11): 51-57.

36 Kohonen T. Self-organisation and associative memory. Springer-Verlag, Berlin, 1988.

37 Apte, C. Data Mining: an industrial research perspective. IEEE Computational Sci. & Eng. 1997; 4(2): 6-9.

38 Kohonen T. Self-organising maps. Springer-Verlag, Berlin, 1995.

39 Cheeseman P and Stutz J. Bayesian classification (AutoClass): theory and results. In: Fayyad UM, Piatetsky-Shapiro G, Smyth P, Uthurusamy R (eds.) Advances in Knowledge Discovery and Data Mining. AAAI Press/MIT Press, 1996, 153-180.

http://fi-www.arc.nasa.gov/fi/projects/bayes-group/group/autoclass/autoclass-c-program.html

40 Cheeseman P, Freeman D, Kelly J, Self M, Stutz, J, Taylor W. AutoClass: a Bayesian classification system. In: Proc Fifth Inter Conf Machine Learning, 1988.

41 Cheeseman P, Stutz J, Self M, Taylor W, Goebel J, Volk K, Walker H. Automatic classification of spectrum from the infrared astronomical satellite (IRAS), NASA reference publication #1217, National technical Information Service, Springfield, Virginia, 1989.

42 Hanson R, Stutz J, Cheeseman P. Bayesian classification theory, http://fi-www.arc.nasa.gov/fi/projects/bayes-group/group/autoclass/autoclass-c-program.html, 1997.

43 Carpenter GA and Grossberg S. ART2: self-organisation of stable category recognition codes for analogue input patterns. Appl Opt 1987b; 26: 4919-4930.

44 Duran BS and Odell PL. Cluster analysis: a survey. Springer-Verlag, 1974.

45 Everitt BS and Dunn G. Applied multivariate data analysis. Edward Arnold. 1991.

46 Everitt BS. Cluster analysis. 3rd edition, Edward Arnold , London, 1993.

47 Grossberg S. Adaptive pattern classification and universal recording: I. parallel development and coding of neural feature detector. Bio Cybernet 1976; 23: 121-134.

48 Grossberg S. Adaptive pattern classification and universal recording: II feedback, expectation, olfaction, illusions. Cybernet 1976; 23: 187-202.

49 Carpenter GA and Grossberg S. A massive parallel architecture for a self-organising neural pattern recognition machine. Computer Vision Graphics and Image Processing 1987; 37: 54-115.

50 Carpenter GA and Grossberg S. ART3: hierarchical search using chemical transmitters in self-organising pattern recognition architectures. Neural Networks 1990; 3: 129-152.

51 Carpenter GA and Grossberg S. The ART of adaptive pattern recognition by a self-organising neural network. Computer 1988; 21:77-88.

52 Buntine W. A guide to the literature on learning probabilistic networks from data. IEEE Trans. Knowl. Data Engng 1996; 8(2):195-210.

53 Buntine W. Graphical models for discovering knowledge. In: Fayyad UM, Piatetsky-Shapiro G, Smyth P, Uthurusamy R (eds.) Advances in Knowledge Discovery and Data Mining. AAAI Press/MIT Press, 1996, 59-82.

54 Cooper GF and Herskovits E. A Bayesian method for the induction of probabilistic networks from data. Machine Learning 1992; 9: 309-347.

55 Bouckaert RB. Properties of Bayesian network learning algorithms. In: Lopez R, Poole, D (Eds.), Uncertainty in Artificial Intelligence: Proc. Ninth Conf., de Mantaras, 1993, 102.

56 Bouckaert RB. Belief network construction using the minimum description length principle. Proc. ECSQARU 1993; 41.

57 Wang XZ, Chen BH, McGreavy C. Data mining for failure diagnosis of process units by learning probabilistic networks. Trans. IChemE 1997; 75B: 210-216.

58 Jensen FV. An introduction to Bayesian networks. UCL Press, 1996.

59 Wang XZ, Yang SA, Veloso E, Lu ML, McGreavy C. Qualitative process modelling - a fuzzy signed directed graph method. Comput. & Chem. Engng 1995; 19(suppl.): s735-s740.

60 Wang XZ, Yang SA, Yang SH, McGreavy C. Fuzzy qualitative simulation in safety and operability studies of process plants. Comput. & Chem. Engng 1996; 20: s671-s676.

61 Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases. In: Proc. ACM SIG-MOD Conf. on Management of Data, Washington, D.C., 1993, 207-216.

62 Agrawal R, Mannila H, Srikant R, Toivonen H, Verkamo AI. Fast discovery of association rules. In: Fayyad UM, Piatetsky-Shapiro G, Smyth P, Uthurusamy R (eds.) Advances in Knowledge Discovery and Data Mining. AAAI Press/MIT Press, 1996, 307-328.

63 Kolodner JL. An introduction to case-based reasoning. Artificial Intelligence 1992; 6: 3-34.

64 Watson I, Marir F. Case-based reasoning: a review. The Knowl. Engng Review 1994; 9: 327-354.

65 Caruana CM. Case-based reasoning gets CPI foothold. Chem. Eng. Progr 1999; 95(4): 9-10.

66 Jefferson M, Chung PW, Kletz TA. Learning the lessons from past accidents. Hazards XIII process safety - the future, symposium series No. 141, The Institution of Chemical Engineers, Manchester, 1997, 217-226.

67 Kraslawski A, Koiranen T, Nystrom L. Case-based reasoning system for mixing equipment selection. Comput. Chem. Engng 1995; 19(suppl.): s821-s826.

68 Graco W, Cooksey RW. Feature selection with data mining. Proc. PADD98 - The second int. conf. on the practical application of knowl. discovery and data mining, London, March 1998, 111-130.

69 Gnanadesikian R, Kettenring JR, Tsao SL. Weighting and selection of variables for cluster analysis. J. of Classification 1995; 12: 113-136.

70 Milligan GW. Clustering validation: results and implications for applied analysis. In: Arabie P, Hubert LJ, De Sote G (eds.), clustering and classification. River Edge NJ: World Scientific, 1996, 341-375.

71 Fayyad UM, Piatetsky-Shapiro G, Smyth P, Uthurusamy R (eds.) Advances in Knowledge Discovery and Data Mining. AAAI Press/MIT Press, 1996.

72 Wu, X. Knowledge acquisition from databases. Ablex, USA, 1995.

73 Simoudis E, Han, J, Fayyad UM (eds.) KDD-96: proceedings-second international conference on knowledge discovery & data mining. AAAI Press, 1996.

74 Wu X, Kotagiri R, Korb K (eds.) Research and development in knowledge discovery and data mining: proceedings of the second Pacific-Asia conference, PAKDD-98. Springer, 1998.

75 Pyle D. Data preparation for data mining. Morgan Kaufmand Publishers, 1999.

76 Moore RL, Kramer MA. Expert systems in on-line process control. Proc. third int. conf. chemical process control, in Chemical Process Control - CPCIII (Edited by Morari M and McAvoy TJ), Asilomar, California, January 1986, Elsevier Amsterdam. p.839.

77 Janusz ME and Venkatasubramanian V. Automatic generation of qualitative

descriptions of process trends for fault detection and diagnosis. Engng Applic. Artif. Intell., 1991; 4: 329-339.

78 Cheung JTY and Stephanopoulos G. Representation of process trends - 1. a formal representation framework. Comput. & Chem Engng 1990; 14: 495-510.

79 Cheung JTY and Stephanopoulos G. Representation of process trends - 2. the problem of scale and qualitative scaling. Comput. & Chem Engng 1990; 14: 511-539.

80 Whiteley JR and Davis JF. Knowledge-based interpretation of sensor patterns. Comput. Chem. Engng 1992; 16: 329-346.

81 Bakshi BR and Stephanopoulos G. Representation of process trends. Part III. Multi-scale extraction of trends from process data. Comput. & Chem Engng 1994; 18: 267-320.

82 Chen BH, Wang XZ, Yang SH, McGreavy C. Application of wavelets and neural networks to diagnostic system development - 1. feature extraction. Comput. Chem. Engng. 1999; 23: 899-906.

83 Wang XZ, Li RF. Combining conceptual clustering and principal component analysis for state space based process monitoring. Internal report, University of Leeds, 1999.

84 Pearson K. On lines and planes of closest fit to systems of points in space. Phi.l Mag. 1901; 2: 559-572.

85 Hotelling H. Analysis of a complex of statistical variables into principal components. J Educ. Psychol. 1933; 24: 417-441, 498-520.

86 Misiti M, Misiti Y, Oppenheim G, Poggi, JM. Wavelet toolbox User's Guide, Version 1. The Math Works Inc.1997.

87 Chan YT. Wavelet basics. Kluwer Academic Publishers, Boston London, 1995.

88 Daubechies I. Ten lectures on wavelets. Society of Industrial and Applied mathematics, Philadelphia, Pennsylvania, 1992.

89 Mallat S. A theory for multiresolution signal decomposition: the wavelet representation. IEEE Pattern Anal. and Machine Intell. 1989; 11: 674-693.

90 Bakshi BR and Stephanopoulos G. Reasoning in time: modelling, analysis, and pattern recognition of temporal process trends. In: Stephanopoulos G, Han C (eds.) Intelligent systems in process engineering - paradigms from design to operations, Academic Press, 1996, 487-549.

91 Bakshi BR and Stephanopoulos G. Representation of process trends. Part IV. Induction of real-time patterns from operating data for diagnosis and supervisory control. Comput. & Chem Engng 1994; 18: 303-332.

92 Mallat S, Hwang WL. Singularity detection and processing with wavelets. IEEE Trans. on Inf. Theory 1992; 38: 617-643.

93 Mallat S, Zhong S. Characterisation of signals from multi-scale edges. IEEE Trans. Pattern Analysis and Machine Intelligence 1992; 14, 710-732.

94 Berman Z, Baras JS. Properties of the multi-scale maxima and zero-crossing representations. IEEE Trans. Signal Processing 1993; 41: 3216-3231.

95 Cvetkovic Z, Vetterli M. Discrete-time wavelet extrema representation: design and consistent reconstruction. IEEE Trans. Signal Processing 1995; 43: 681-693.

96 William BC. Doing time: putting qualitative reasoning on firmer ground. National conf. on artif. intell., Philadelphia, 1986.

97 Mallat S. Zero-crossing of a wavelet transform. IEEE Trans. on Information Theory 1991; 37: 1019-1033.

98 Carrier JF, Stephanopoulos G. Wavelet-based modulation in control-relevant process identification. AIChE J. 1998; 44: 341-360.

99 Dai X, Joseph B, Motard RL. Introduction to wavelet transformation and time frequency analysis. In: Motard RL, Joseph B (eds.) Wavelet applications in chemical engineering. Kluwer Academic Publishers, 1994, 1-32.

100 Joshi A, Kumar A, Motard RL. Trend analysis using the Frazier-Jawerth transform. In: Motard RL, Joseph B (eds.) Wavelet applications in chemical engineering. Kluwer Academic Publishers, 1994, 85-113.

101 Dai X, Joseph B, Motard RL. Process signal feature analysis. In: Motard RL, Joseph B (eds.) Wavelet applications in chemical engineering. Kluwer Academic Publishers, 1994, 115-137.

102 Ramesh TS, Shum SK, Davis JF. A structured framework for efficient problem-solving in diagnostic expert systems. Comput. Chem. Engng 1988; 12: 891-902

103 MacGregor JF, Kourti T. Multivariate statistical treatment of historical data for productivity and quality improvements. In: Pekny J, Blau G. (eds.) Foundation of computer aided process operations, University of Michigan, Ann Arbor Press , 1998, in print.

104 Kresta JV, MacGregor JF, Marlin TE. Multivariate statistical monitoring of process operating performance. Cana. J. Chem. Eng. 1991; 69: 35-47.

105 Kourti T, MacGregor JF. Process analysis, monitoring and diagnosis, using projection methods - a tutorial. Chemometrics and Intell. Lab. Systems 1995; 28:3-21.

106 Jobson JD. Applied multivariate data analysis. Vol II: categorical and multivariate methods. Springer-Verlag New York, Inc., 1992.

107 Hotelling H. Multivariate quality control, illustrated by the air testing of sample bombsights. In: Eisenhart C, Hastay MW, Wallis WA (eds.) Techniques of Statistical Analysis, McGraw-Hill, New York, 1947, 113-184..

108 Alt FB, Smith ND. Multivariate process control. In: Krishnaiah PR, Rao CR (eds.) Handbook of statistics, Vol. 7, North-Holland, Amsterdam, 1988, 333-351.

109 Ryan TP. Statistical methods for quality improvement. Wiley, New York, 1989.

110 Jackson JE. A user's guide to principal components. John Wiley and Sons, Inc., New York, 1991.

111 Tracy ND, Young JC, Mason RL. Multivariate control charts for individual observations. J. Quality Tech. 1992; 24: 88-95.

112 MacGregor JF, Kourti T. Statistical process control of multivariable processes. Control Engng Practice 1995; 3: 403-414.

113 MacGregor JF, Jaeckle C, Kiparissides C, Koutoudi M. Process monitoring and diagnosis by multiblock PLS methods. AIChE J. 1994; 40: 826-838.

114 Martin EB, Morris J, Papazoglou MC. Confidence bounds for multivariate process performance monitoring charts. Preprints of the IFAC workshop on on-line fault detection and supervision in the chemical process industries, Newcastle , June 1995; 33-42.

115 Wold S, Geladi P, Esbensen K, Ohman J. Multi-way principal components- and PLS-analysis. J. Chemometrics 1987; 1:41-56.

116 Nomikos P, MacGregor JF. Monitoring batch process using multiway principal component analysis. AIChE J. 1994; 40 1361-1375.

117 Palus M, Dvorak I. Singular-value decomposition in attractor reconstruction: pitfalls and precautions. Physica D 1992; 55: 221-234.

118 Xu L, Oja E, Suen CY. Modified hebbian learning for curve and surface fitting. Neural Networks 1992; 5: 441-457.

119 Dong D, McAvoy TJ. Nonlinear principal component analysis - based on principal curves and neural networks. Comput Chem Engng 1996; 20: 65-78.

120 Gnanadesikian R. Methods for statistical data analysis of multivariate observations. Wiley, New York, 1977.

121 Etezadi-Amoli J, McDonald RP. A second generation nonlinear factor analysis. Psychometrika 1983; 48(3): 315-327.

122 Kramer MA. Nonlinear principal component analysis using autoassociative neural networks. AIChE J 1991; 37: 233-243.

123 Sammon JW. A nonlinear mapping for data structure analysis. IEEE Trans. Comput. 1969; C18: 401.

124 Yuan B, Wang XZ, Chen FZ, Morris T. Software analyser design using data mining technology for toxicity prediction of aqueous effluents. Proc. 2nd Conf. on Process Integration, Modelling and Optimisation for Energy Saving and Pollution Reduction

(Pres'99). Budapest, Hungary, May 31 - June 2, 1999.

125 Bakshi BR. Multiscale PCA with application to multivariate statistical process monitoring. AIChE J. 1998; 44:1596-1610.

126 Tabe H, Chow KC, Tan KJ, Zhang J, Thornhill N. Dynamic principal component analysis using integral transforms. AIChE Annual Meeting, November 1998.

127 Biehl M, Schlosser E. The dynamics of on-line principal component analysis. J Phys. A: Math. Gen. 1998; 31: L97-L103.

128 Neogi D, Schlags CE. Multivariate statistical analysis of an emulsion batch process. Ind. Eng. Chem. Res. 1998; 37: 3971-3979.

129 Jaeckle CM, MacGregor JF. Product design through multivariate statistical analysis of process data. AIChE J. 1998; 44: 1105-1118.

130 Kresta JV, Marlin TE, MacGregor JF. Development of inferential process models using PLS. Comput. Chem. Engng. 1994; 18: 597-611.

131 Santen A, Koot GLM and Zullo LC. Statistical data analysis of a chemical plant. Comput Chem Engng 1997; 21(suppl.): s1123-s1129.

132 Zhang J, Martin EB, Morris AJ. Fault detection and diagnosis using multivariate statistical techniques. Trans. IChemE 1996; 74A: 89-96.

133 Dunia R, Qin SJ, Edgar TF, McAvoy TJ. Identification of faulty sensors using principal component analysis. AIChE J. 1996; 42: 2797-2812.

134 Qin SJ, Yue HY, Dunia R. Self-validating inferential sensors with application to air emission monitoring. Ind. Eng. Chem. Res. 1997; 36: 1675-1685.

135 Chen FZ, Wang XZ. Knowledge discovery using PCA for operational strategy development and product design. Submitted to Trans. IChemE Part A, 1999.

136 Chen JG, Bandoni JA, Romagnoli JA. Robust PCA and normal region in multivariate statistical process monitoring. AIChE J. 1996; 42: 3563-3566.

137 Negiz A, Cinar A. Statistical monitoring of multivariable dynamic processes with state space models. AIChE J. 1997; 43: 2002-2020.

138 Leonard J, Kramer MA. Improvement of the backpropagation algorithm for training neural networks. Comput Chem Engng 1990; 14: 337-341.

139 Brent RP. Fast training algorithms for multilayer neural nets. IEEE Trans. Neural Nets 1991; 2:346-354.

140 Chen S, Billings SA. Neural networks for nonlinear dynamic system modelling and identification. Int. J. Control 1992; 56: 319-346.

141 Peel C, Willis MJ, Tham MT. A fast procedure for the training of neural networks. J. Proc. Cont. 1992; 2: 205-211.

142 Powell MJD. Some global convergence properties of a variable metric algorithm for minimisation without exact line searches. In: Cottle R, Lemke CE (eds.) SIAM-AMS Proc. Symposium on non-linear programming, 1975, IX: 53-72.

143 Press WH, Flannery, BP, Teukolsky, SA, Vetterling, WT. Numerical recipes, the art of scientific computing (Fortran version). Cambridge University Press, Cambridge, 1989.

144 Lorentz, GG. The 13th problem of Hilbert. In: Browder FE (Ed.) Mathematical developments from Hilbert problems. America Mathematical Society, Providence, R.I., 1976.

145 Crowe ER, Vassiliadis CA. Artificial intelligence: staring to realise its practical promise. Chem. Eng. Progr. 1995; 91(1): 22-31.

146 Knight K. Connectionist ideas and algorithms. Comm. of the ACM 1990; 33 (11): 59-74.

147 Chitra SP. Use of neural networks for problem solving. Chem. Eng. Progr. 1993; 89(4): 44-52.

148 Kramer MA, Leonard, JA. Diagnosis using backpropagation neural networks - analysis and criticism. Comput. Chem. Engng 1990; 14: 1323-1338.

149 Shao R, Martin EB, Zhang J, Morris AJ. Confidence bounds for neural network representations. Comput. Chem Engng 1997; 21: s1173-s1178.

150 Zhang J, Martin EB, Morris AJ, Kiparissides C. Inferential estimation of polymer quality using stacked neural networks. Comput. Chem. Engng 1997; 21: s1025-s1030.

151 Wang XZ, Lu ML, McGreavy C. Learning dynamic fault models based on a fuzzy set covering method. Comput. Chem Engng 1997; 21: 621-630.

152 Kramer MA. Malfunction diagnosis using quantitative models with non-Boolean reasoning in expert systems. AIChE J. 1987; 33: 130-140.

153 Venkatasubramanian V, Vaidyanathan R, Yamamoto Y. Process fault-detection and diagnosis using neural networks, 1. steady-state processes. Comput. Chem. Engng 1990, 14: 699-712.

154 Zhang J, Morris J. Process modelling and fault diagnosis using fuzzy neural networks. Fuzzy Sets and Systems 1996; 79: 127-140.

155 Wang XZ, Chen BH, Yang SH, McGreavy C. Neural nets, fuzzy sets and digraphs in safety and operability studies of refinery reaction processes. Chem Engng Sci 1996; 51: 2169-2178.

156 Buckley JJ, Hayashi Y. Fuzzy neural networks: a survey. Fuzzy Sets and Systems 1994; 66: 1-13.

157 Reggia JA, Nau DS, Wang PY. Diagnostic expert system based on a set covering

model. Int J Man-machine Studies 1983; 19: 437-460.

158 Penalva JM, Coudouneau L, Leyval L, Montmain J. A supervision support system for industrial processes. IEEE Expert 1993; 8(5): 57-65.

159 Iri M, Aoki E, O'Shima E, Matsuyama H. An algorithm for diagnosis of system failures in the chemical process. Comput. Chem. Engng 1979; 3: 489-493.

160 Oyeleye OO, Kramer MA. Qualitative simulation of chemical process systems: steady-state analysis. AIChE J. 1988; 34: 1441-1454.

161 Yu CC, Lee C. Fault diagnosis based on qualitative/quantitative process knowledge. AIChE J. 1991; 37: 617-628.

162 Gujima F, Shibata B, Tsuge Y, Shiozaki J, Matsuyama H, O'shima E. Improvements of the accuracy of fault diagnosis systems, using signed directed graphs. Int. Chem. Engng. 1993; 33: 671-679.

163 Mohindra S, Clark PA. A distributed fault diagnosis method based on graph models: steady-state analysis. Comput. Chem. Engng 1993; 17: 193-209.

164 Wilcox NA, Himmelblau DM. The possible cause-effect graph (PCEG) model for fault diagnosis. Comput. Chem. Engng 1994; 18: 103-127.

165 Ouassir M, Melin C. Causal graphs and rule generation: application to fault diagnosis of dynamic processes. Proceedings of the 3rd IFAC symposium on fault detection, supervision and safety for technical processes 1997 (SAFEPROCESS 97), Kingston Hull, England, Aug. 1997, 1087-1092.

167 Vedam H, Venkatasubramanian V. Signed digraph based multiple fault diagnosis. Comput. Chem. Engng 1997; 21: s655-s660.

168 Umeda T, Kuriyama T, O'shima E. Matsuyama H. A graphical approach to cause and effect analysis of. chemical processing systems. Chem. Eng. Sci. 1980; 35: 2379-2388.

169 Vaidhyanathan R, Venkatasubramanian V. Digraph-based models for automated HAZOP analysis. Reliability Engineering and System Safety 1995; 50: 33-49.

170 Srinivasan R, Venkatasubramanian V. Petri Net-Digraph models for automating HAZOP analysis of batch process plants. Comput. Chem. Engng 1996; 20: s719-s725.

171 Kuo DH, Hsu DS, Chang CT, Chen DH. Prototype for integrated hazard analysis. AIChE J. 1997; 43: 1494-1510.

172 Han CC, Shih RF, Lee LS. Quantifying signed directed-graphs with the fuzzy set for fault diagnosis resolution improvement. Ind. Eng. Chem. Res. 1994; 33: 1943-1954.

173 Shih RF, Lee LS. Use of fuzzy cause-effect digraph for resolution fault diagnosis for process plants. Ind. Eng. Chem. Res. 1995; 34: 1688-1717.

174 Wang XZ, Chen BH, Yang SH, McGreavy C. Fuzzy rule generation from data for process operational decision support. Comput. Chem. Engng 1997; 21: s661-s666.

175 Huang YC, Wang XZ. Application of fuzzy causal networks to wastewater treatment plants. Chem. Eng. Sci. 1999; 54: 2731-2738.

176 Chen CL, Jong MJ. Fuzzy predictive control for the time-delay system. Proc. 2nd IEEE int. conf. fuzzy systems 1993; 236-240.

177 Becraft WR, Lee PL. An integrated neural network/expert system approach for fault diagnosis. Comput. Chem. Engng 1993; 17: 1001-1014.

178 Dubois D, Prade H. Fuzzy sets and systems: theory and applications. Academic Press, 1980

179 Rosenfield A. Fuzzy graphs. In: Zadeh LA, Fu KS, Tanaka K, Shimura M (eds.) Fuzzy sets and their applications to cognitive and decision processes. Academic Press, New York, 1975, 77-95.

180 Zhang J, Morris AJ, Martin EB, Kiparissides C. Prediction of polymer quality in batch polymerisation reactors using robust neural networks. Chem. Eng. J. 1998; 69: 135-143.

181 Caudil M. Neural network primer, Part VIII, AI Expert 1989.

182 Wasserman PD. Neural computing: theory and practice. Van Nostrand Reinhold, New York, 1989.

183 Whiteley JR, Davis JF. A similarity-based approach to interpretation of sensor data using adaptive resonance theory. Comput. & Chem. Engng 1994; 18: 637-661.

184 Whiteley JR, Davis JF Mehrotra A and Ahalt SC. Observations and problems applying ART2 for dynamic sensor pattern interpretation. IEEE Trans. on Sys. Man Cybernetics Part A: Sys. and Humans 1996; 26: 423-437.

185 Wang XZ, Chen BH, Yang SH, McGreavy C. Application of wavelets and neural networks to diagnostic system development - 2. an integrated framework and its application. Comput. Chem. Engng. 1999; 23: 945-954.

186 Pao YH. Adaptive pattern recognition and neural networks. Addison-Wesley, 1989.

187 Looney CG. Pattern recognition using neural networks: theory and algorithms for engineers and scientists. Oxford University Press, 1997.

188 Everitt BS, Hand DJ. Finite mixture distributions. London Chapman & Hall, 1981.

189 Titterington DM, Smith AFM, Makov UE. Statistical analysis of finite mixture distributions. John Wiley & Sons, New York, 1985.

190 Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society 1977; 39 (series B), No. 1: 1.

191 Ayoubi M, Leonhardt S. Methods of fault diagnosis. Control Eng. Practice 1977; 5: 683.

192 Wolfe JH. Pattern clustering by multivariate mixture analysis. Multivariate behavioural research 1970; 5: 329-350.

193 Wang XZ, McGreavy C. Automatic classification for mining process operational data. Ind. Eng. Chem. Res. 1998; 37: 2215-2222.

194 Nam DS, Han C, Jeong CW, Yoon ES. Automatic construction of extended symptom-fault associations from the signed digraph. Comput. Chem. Engng. 1996; 20(suppl.): s605-s610.

195 Wang XZ, Chen BH. Clustering of infrared spectra of lubricating base oils using adaptive resonance theory. J. Chem. Inf. Comput. Sci. 1998; 38: 457-462.

196 Michalski, R.S. and Larson, J.B. Selection of most representative training examples and incremental generation of VLI hypotheses: the underlying method and desc. of prog. ESEL and AQ11, 867, Computer Science Department, University of Illinois, 1978.

197 Mitchell, T.M. Version Spaces: a candidate elimination approach to rule learning, in Proc. of IJCAI-87, Cambridge, Mass., 1977

198 Quinlan JR. Improved use of continuous attributes in C4.5. J. Artif. Intell. Res. 1996; 4: 77-90.

199 Wu, X. A Bayesian discretiser for real-valued attributes. The Comput J., 1996; 39: 688-691.

200 Daniel C, Wood FS. Fitting equations to data, 2nd Edition. John Wiley & Sons, Inc., 1980.

201 Raich A, Cinar A. Statistical process monitoring and disturbance diagnosis in multivariable continuous processes. AIChE J. 1996; 42: 995-1009.

202 Shi ZZ. Principles of machine learning. International Academic Publishers, Beijing, 1992.

203 Dutton DM, Conroy GV. A review of machine learning. The Knowl. Engng Review 1996; 12: 341-367.

204 Kocabas S. A review of learning. The Knowl. Engng Review 1991; 6: 195-222.

205 Wang LX, Mendel JM. Generating fuzzy rules by learning from examples. IEEE Trans. on Systems, Man, and Cybernetics 1992; 22: 1414-1427.

206 Wang LX, Mendel JM. Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. IEEE Trans. on Neural Networks 1992; 3: 807-814.

207 Wang LX. Adaptive fuzzy systems and control: design and stability analysis.

Prentice-Hall, Inc., 1994.

208 Wang LX. Fuzzy systems are universal approximators. Proc. IEEE Int. Conf. on Fuzzy Systems, San Diego, 1992, 1163-1170.

209 Quafafou M. and Chan CC. An incremental approach for learning fuzzy rules from examples. EUFIT' 95, Aachen, Germany, 1995, 520-523.

210 Srinivasan ACB, Chan CC. Using inductive learning to determine fuzzy rules for dynamic system. Engng Applic. of Artif. Intell. 1993; 6: 257-264.

211 Abe S, Lan MS. A method for fuzzy rules extraction directly from numerical data and its application to pattern classification. IEEE Trans. on Fuzzy Systems 1995; 3: 18-28.

212 Foulloy L, Galichet S, Nakoula Y. Learning a fuzzy symbolic rulebase. EUFIT'95, Aachen, Germany, 1995, 594-598.

213 Schretter N, Hollatz J. Consistency and completeness in fuzzy rulebases generated from numerical data. EUFIT '95, Aachen, Germany, 1995, 599-604.

214 Levrat E, Rondeau L, Ruelas R, Lamotte M. Fuzzy rules learning method. EUFIT '95, Aachen, Germany, 1995, 515-519.

215 Sudkamp T, Hammell RJ. Interpolation, completion, and learning fuzzy rules. IEEE Trans. on Systems, Man, and Cybernetics 1994; 24: 332-342.

216 Kosko, B. Neural networks and fuzzy systems: a dynamic systems approach to machine intelligence. Prentice Hall, Englewood Cliffts, NJ, 1992.

217 Fu LM. Rule generation from neural networks. IEEE Trans. on Systems, Man, and Cybernetics 1994; 24: 1114-1124.

218 Gallant SI. Connectionist expert systems. Comm. of The ACM 1988; 31(2): 152-169.

219 Chan CC. Incremental learning of production rules from examples under uncertainty: a rough set approach. Int. J. of Software Engng and Knowl. Engng 1991; 1: 439-461.

220 Chmielewski MR, Grzymala-Busse JW, Peterson NW, Than S. The rule induction system LERS - a version for personal computers. Foundations of Computing and Decision Sciences 1993; 18: 181-212.

221 Pawlak Z. Rough sets. Int. J. Computer and Information Sci. 1982; 11: 341-356.

222 Pawlak Z. Rough classification. Int. J. man-machine Studies 1984; 20: 469-483.

223 Pawlak Z. Rough sets and fuzzy sets. Fuzzy Sets and Systems 1985; 17: 99-102.

224 Fusillo RH, Powers GJ. Operating procedure synthesis using local models and distributed goals. Comput. Chem Engng 1988; 12: 1023-1034.

225 Frayman Y, Wang L. Data mining using dynamically constructed recurrent fuzzy neural networks. In: Wu X, Kotagiri R, Korb KB (eds.) Research and development in

knowledge discovery and data mining, Springer, 1998, 122-131.

226 Ziarko WP (ed.) Rough sets, fuzzy sets and knowledge discovery. Springer-Verlag, 1994.

227 Chen FZ, Wang XZ. Software sensor design using Bayesian automatic classification and backpropagation neural networks. Ind. Eng. Chem. Res. 1998; 37: 3985-3991.

228 McAvoy TJ, Su HT, Wang NS, He M, Horvath J, Semerjian H. A comparison of neural networks and partial least squares for deconvoluting fluorescence spectra. Biotech. and Bioeng 1992; 40: 53-62.

229 Baughman DR, Liu YA. Neural networks in bioprocessing and chemical engineering. Academic Press, 1995.

230 Miller RM, Itoyama K, Uda A, Takada H, Bhat N. Modeling and control of a chemical waste water treatment plant. Comput. Chem. Engng 1997; 21(suppl.): s947-s952.

231 Gillespie ES, Wilson RN. Application of sensitivity analysis to neural network determination of financial variable relationships. Applied Stochastic Models and Data Analysis 1998; 13: 409-414.

232 Garson GD. Interpreting neural-network connection weights. AI Expert 1991; April: 47-51.

233 Milne LK, Gedeon TD, Skidmore AK. Classifying dry sclerophyll forest from augmented satellite data: comparing neural networks, decision tree & maximum likelihood. Proc. Australian Conf. Neural Networks, Sydney, 1995, 160-163.

234 Wong PM, Gedeon TD, Taggart IJ. An improved technique in porosity prediction: a neural network approach. IEEE Trans. Geoscience and Remote Sensing 1995; 33: 971-980.

235 Gedeon TD. Data mining of inputs: analysing magnitude and functional measures. Int. J. Neural Systems 1997; 8: 209-218.

236 Ayoubi M. Comparison between the dynamic multi-layered percetron and the generalised Hammerstein model for experimental identification of the loading process in diesel engines. Control Engng Practice 1998, 6: 271-279.

237 Bhat N, McAvoy TJ. Use of neural nets for dynamic modelling and control of chemical process systems. Comput. Chem. Engng 1990; 14: 573-583.

238 Shaw AM, Doyle FJ, Schwaber JS. A dynamic neural network approach to nonlinear process modelling. Comput. Chem. Engng 1997; 21: 371-385.

239 Morris AJ, Montague GA, Willis MJ. Artificial neural networks: studies in process modelling and control. Trans. IChemE. 1994; 72A: 3-19.

240 Kim SJ, Lee MH, Park SW, Lee SY, Park CH. A neural linearizing control scheme for nonlinear chemical processes. Comput. Chem. Engng 1997; 21: 187-200

241 Martinez EC, Pulley RA, Wilson JA. Learning to control the performance of batch processes. Trans IChemE 1998; 76A: 711-722.

INDEX