DATA DRIVEN PROCESS MONITORING BASED ON NEURAL NETWORKS AND CLASSIFICATION TREES

A Dissertation

by

YIFENG ZHOU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2004

Major Subject: Chemical Engineering

DATA DRIVEN PROCESS MONITORING BASED ON NEURAL NETWORKS AND CLASSIFICATION TREES

A Dissertation

by

YIFENG ZHOU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved as to style and content by:

_____	_____
M. Sam Mannan (Co-Chair of Committee)	Juergen Hahn (Co-Chair of Committee)
_____	_____
Harry H. West (Member)	Shankar P. Bhattacharyya (Member)

_____
Kenneth R. Hall
(Head of Department)

August 2004

Major Subject: Chemical Engineering

ABSTRACT

Data Driven Process Monitoring Based on Neural Networks and Classification Trees. (August 2004)

Yifeng Zhou, B.S., Xian Jiao-Tong University, China;

M.S., Research Institute of Petroleum Processing, China

Chair of Advisory Committee: Dr. M. Sam Mannan

Process monitoring in the chemical and other process industries has been of great practical importance. Early detection of faults is critical in avoiding product quality deterioration, equipment damage, and personal injury. The goal of this dissertation is to develop process monitoring schemes that can be applied to complex process systems.

Neural networks have been a popular tool for modeling and pattern classification for monitoring of process systems. However, due to the prohibitive computational cost caused by high dimensionality and frequently changing operating conditions in batch processes, their applications have been difficult. The first part of this work tackles this problem by employing a polynomial-based data preprocessing step that greatly reduces the dimensionality of the neural network process model. The process measurements and manipulated variables go through a polynomial regression step and the polynomial coefficients, which are usually of far lower dimensionality than the original data, are used to build a neural network model to produce residuals for fault

classification. Case studies show a significant reduction in neural model construction time and sometimes better classification results as well.

The second part of this research investigates classification trees as a promising approach to fault detection and classification. It is found that the underlying principles of classification trees often result in complicated trees even for rather simple problems, and construction time can excessive for high dimensional problems. Fisher Discriminant Analysis (FDA), which features an optimal linear discrimination between different faults and projects original data on to perpendicular scores, is used as a dimensionality reduction tool. Classification trees use the scores to separate observations into different fault classes. A procedure identifies the order of FDA scores that results in a minimum tree cost as the optimal order. Comparisons to other popular multivariate statistical analysis based methods indicate that the new scheme exhibits better performance on a benchmarking problem.

To my parents

ACKNOWLEDGEMENTS

I have to express my sincere gratitude toward my advisor and mentor, Dr. Sam Mannan, for his stimulating guidance, patience, encouragement, and unfaltering support throughout these last five years. I believe that selecting him as my advisor was one of the best decisions that I have ever made. In addition to the enormous guidance on my research, he has had a lasting impact on my career.

I would also like to thank Dr. Juergen Hahn, Dr. Harry H. West, and Dr. Shankar P. Bhattacharyya for serving on my advisory committee and for their guidance on my research. Special thanks to friends and mentors Dr. Jiaxiang Zhou, Dr. Hong P. Gao of Applied Materials, Inc. and Dr. Harold Wade of Wade Associates, Inc., for technical advice and career guidance. Gratitude is expressed to the faculty, staff and graduate students at the Mary Kay O'Connor Process Safety Center and the Chemical Engineering Department at Texas A&M University for their service and support.

I would like to acknowledge my family members and friends for their love and friendship.

TABLE OF CONTENTS

Page

Page

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

Page

I. INTRODUCTION

1. Fault detection and diagnosis

In chemical, petrochemical, food processing, papermaking, steel, power and other process industries, there has been a continuing demand for higher quality products, lower product rejection rates, and satisfying increasingly stringent safety and environmental regulations. Implementation and improvement of digital control scheme has been essential over the last three decades in order to meet these ever increasing standards especially since modern process plants are often large scale, highly complex, and operate with a huge number of process variables under closed loop control. Common distributed control systems are designed to replace human operators for tasks such as opening and closing valves, and perform those actions in an automated manner with greater accuracy. Application of advanced control systems, including supervisory control and model predictive control, has been of enormous benefits to various process industries.

The primary purpose of process control systems is to maintain satisfactory operations by compensating for the disturbances and changes occurring in the process. While control systems are capable of compensating most disturbances and changes, there are changes in the process that the control system cannot adequately handle, which

---

This dissertation follows the style and format of the *ISA Transactions.*

we call abnormal conditions, or by a more precise definition, undesired deviation of at least one variable or characteristic property of the system [1].

Despite the fact that automatic control has relieved the operators from most regulatory control tasks, responding to abnormal conditions has largely remained a manual activity. This includes timely detection of abnormal conditions, locating the cause and origin of the abnormal conditions, and taking appropriate measures to bring the process back to normal operating conditions. This entire activity is often referred to as Abnormal Conditions Management (ACM), Abnormal Situation Management (ASM)[§], or Abnormal Events Management (AEM).

Chemical plants experience numerous faults, which include sensor faults, process faults, actuator faults, and unexpected disturbances. A dead sensor is a common type of sensor faults while catalyst poisoning is an example of process faults. A sticking control valve represents actuator faults and an unexpected disturbance can be an extreme change in the feed concentration of a reactor.

To minimize product rejection rate and plant down time, the faults need to be detected and diagnosed. A popular definition of fault detection is determining whether a fault has occurred, and fault diagnosis determines which fault has occurred, or determining the causes of out-of-control status [2]. Application of fault detection and diagnosis systems can minimize manufacturing cost, preventing damage to equipments, and improve the safety of plant operations. As chemical plants are increasingly

[§] Abnormal Situation Management, ASM are registered service marks of Honeywell International Inc.

complex, highly integrated and heavily instrumented, the problem of detecting and diagnosing faults is becoming strategically important.

A typical fault detection and diagnosis (FDD) scheme contains one or more measures, which are calculated from the process data and somehow represent the state of the process. By developing measures that accurately characterize the process behavior, faults can be detected and diagnosed by comparing the values of the measures with their values of the normal operating condition.

FDD is closely related with Abnormal Situation Management (ASM), which deals with events that are out of the desired operating conditions. While FDD is a collection of technical procedures, ASM solutions contain many interactive parts, including management issues, hazard identification and risk analysis, process monitoring, and abnormal situation responding procedures. FDD constitutes an integrated part of process monitoring, as shown in Figure 1.1.

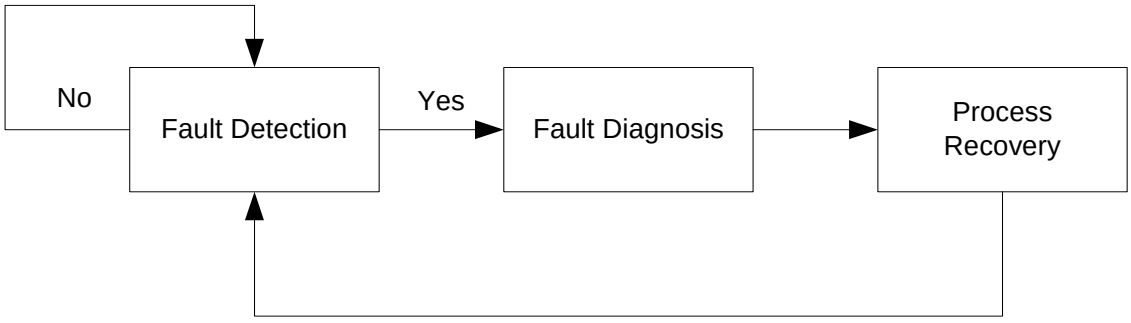


Fig. 1.1 A schematic diagram of the process monitoring loop

2. Overview

FDD measures are mainly derived from three approaches: analytical model-based, data driven, and knowledge-based. Analytical model-based approaches require accurate process models and a sufficient number of sensors for their implementation; data driven methods directly derive measures from process data and knowledge based approach uses qualitative models such as fault tree analysis.

2.1 Analytical model based approach

When the faults are connected with changes in unmeasurable state variables, a state estimation approach is appropriate. These unmeasurable states are reconstructed from the measurable input and output variables of the process using a Kalman Filter [3] [4], an unknown input observer [5], or a non-linear state observer[6]. The parameter estimation approach is appropriate if the process faults are associated with changes in parameters in the process model. The model parameters can be calculated with a least-squares method, which can be implemented recursively to save computational costs. This approach is very intuitive if the process model is constructed from first principles and the model parameters have physical meaning in the process. When both unmeasurable states and parameters are needed for fault diagnosis, there exist observers that are capable of joint estimation of states and parameters [7].

A more advanced analytical model-based approach is analytical redundancy. Approaches that use analytical redundancy incorporate an explicit process model to generate and evaluate residuals [8]. In some cases the residual is generated between the

model prediction and the plant measurement. Observer-based approaches can be used to reconstruct the output of the system from the measurements with the aid of observers, in which case the output estimation error is used as the residual [9][10]. In the case of parameter estimation, the residuals can be taken as the difference between the nominal model parameters and the estimated model parameters. The second step of analytical redundancy approaches is residual evaluation, in which the resulting residual is used as feature input to fault detection and diagnosis through logical [11], causal [12], or pattern recognition techniques [13].

Analytical model based approaches require accurate models to be effective [14]. It is shown that when uncertain models are used, control performance must be traded off against diagnostic performance [15] and the fault diagnosis and control schemes should be designed together [16]. Most applications of the analytical model-based methods have been to relatively small-scale systems [17] [18], i.e., systems with small number of inputs, outputs, and states. Accurate models for large-scale systems like the current chemical plants are difficult to obtain because of all the cross-couplings associated with a multivariable system.

2.2 Knowledge-based approach

For large-scale systems, accurate models may not be available or may be too costly and time-consuming to obtain. An alternative method is to use knowledge-based methods such as causal analysis and expert systems, which are based on qualitative models. Approaches based on causal analysis use the concept of causal modeling of

fault-symptom relationships. An example of causal analysis is the signed directed graph (SDG) [19] [20]. An SDG is a representation of the process causal information, in which the process variables and parameters are represented as graph nodes and causal relations are represented by directed arcs. Nodes in the SDG assume values of (0), (+) and (-) representing the nominal steady state value, higher and lower than steady state values. Directed arcs point from a cause node to its effect node. Arc signs associated with each directed arc could take values of (+) and (-) representing whether the cause and effect change in the same direction or opposite direction, respectively. The advantage of SDG is that all possible nodes can be located to explain an abnormal operating condition. However, constructing an SDG for a large chemical plant may be tedious.

A traditional approach for building a knowledge expert system is to develop IF-THEN rules through process knowledge [21]. Knowledge expert systems are flexible, fast to implement, and the conclusions are easy to verify. A new approach for building knowledge expert systems is to use machine-learning techniques. This system allows knowledge extraction and background knowledge encoding by integrating symbolic information into a neural network learning algorithm [22].

A real-time version of a fault tree model, a symptom tree model (STM), relates the faults and symptoms [22] [23]. In an STM, the root cause of a fault is determined by taking the intersection of causes attached to the observed symptoms. Often this procedure will result in more than one candidate faults, and there is no mechanism

available to decide the most probable fault. This problem is solved by a weighted symptom tree model (WSTM) [24], which contains a weight for each symptom-root cause pair. WSTM allows ranking all possible faults by their probabilities, and a subsequent pattern-matching step matches the observed fault propagation trends with standard trends obtained from training set. The best matching fault is selected as the diagnosed result.

Knowledge-based approaches are also difficult to apply to large scale systems because constructing the fault models demands a large amount of effort and requires extensive process experience [25].

2.3 Data driven approach

Because accurate models are difficult to develop for large-scale systems, most of the fault detection and diagnosis systems applied to industrial processes are based on data-driven approaches. Process data collected from the normal and abnormal operating conditions are used to develop measures for detecting and diagnosing faults. This class of methods includes multivariate statistical analysis, neural networks, classification and regression trees (CART), and hybrid methods using a combination of various techniques. Since these methods are data-driven, the effectiveness of these methods is highly dependent on the quality and the quantity of the process data. Although modern control systems allow acquiring huge amounts of process data, only a small portion is usable as it is often not certain these data are not corrupted and no unknown faults are occurring.

Multivariate statistical analysis-based approaches formulate the fault diagnosis problem as a statistical pattern recognition problem. Since process data are always subject to random disturbances, it is natural to analyze the system in a probabilistic setting. Multivariate statistical techniques, including principal component analysis (PCA), partial least squares (PLS), and Fisher discriminant analysis (FDA), are capable of reducing the dimensionality of the original data such that essential information is retained; they are also able to classify data points to pre-determined fault classes.

Neural networks-based approaches assume a functional form for the decision rules thus describing the decision boundaries with a set of network parameters. The network parameters are obtained either by supervised training or unsupervised training using historical process data.

3. Statement of the problem

Although the applicability of data-driven methods for process fault detection and diagnosis has been well demonstrated, difficulties still arise when the process experiences slow deviation or frequent changes of operating conditions. The models for the data driven methods thus must be retrained to reflect the current process behavior. This is not a problem for simple statistical models such as PCA, but for computationally intensive models like neural networks or CART, excessive retraining time often becomes the bottleneck for their applications [26]. It is especially unacceptable for batch processes where online retraining is often required.

The objectives of this research are to:

- Investigate appropriate dimensionality reduction techniques for neural network and CART approaches for process fault detection and diagnosis
- Determine the optimal degree of dimensionality reduction by balancing loss of information and system complexity
- Validate the developed algorithms on both simulated and real chemical processes and compare the results to that of the existing techniques

II. FAULT DETECTION AND CLASSIFICATION BASED ON NEURAL NETWORKS WITH FEATURE EXTRACTION*

1. Introduction

Batch processes are routinely used for manufacturing high value added chemicals such as pharmaceuticals, polymers, and fine chemicals. The batch mode of operation is preferred when the production volume is low and when the materials involved are hard to handle. The operating procedures often follow a "golden recipe" consisting of a series of operating conditions and processing times. Unlike continuous processes, batch processes are characterized by time-varying variables and process parameters. As a result, normal states are not described by the ranges of variables but by a set of acceptable "trajectories".

Monitoring tasks in batch processes are very difficult due to many reasons. For example, the absence of steady states operations, high nonlinearities of the processes, lack of online sensors for measuring product composition and the finite duration of the operation are the major problems for developing process supervision systems. Moreover, batch processes are generally operated as multi-product plants where many products share common equipment. Thus the production scheduling must be taken into account for development of supervision systems, because abnormal situations are

* Reprinted with permission from "Fault detection and classification in chemical processes based on neural networks with feature extraction", Yifeng Zhou, Juergen Hahn, and M. Sam Mannan, 2003, *ISA Transactions*, Volume 42, Number 4, October 2003, 651-664.

strongly dependent on the product manufacturing processes. Finally, the variability of batch-to-batch operations due to different operators and initial conditions implies substantial system complexity and uncertainty.

Due to the inherent complexity and flexibility of batch processes, fault diagnosis systems that provide fault information to operating and scheduling levels allow improvement of product quality, facilitate active scheduling, and reduce risk of accidents. Fault detection and diagnosis problems have been intensively studied in recent years. Most of the approaches presented so far are applicable to steady state processes. These approaches can be divided into three groups: data driven methods [27][28][29][30], model-based methods [31][32] and combinations of both. However, due to the nonlinearity, complexity and absence of steady state operating conditions, the application of these methods to batch processes is usually very difficult.

More recently, the potential of neural networks for fault diagnosis has been demonstrated [33][34][35]. A neural network-based approach is especially suitable for processes for which accurate mathematical models are too difficult or too expensive to obtain. Neural networks attempt to mimic the computational structures of the mammal brains by nonlinear mapping between input and output that consists of interconnected nodes arranged in layers. The layers are connected such that the signal on the input side can propagate through the network and reach the output side. Neural network behaviors are determined by the transfer functions of the units, network topology, and connection pattern of layers. Among all forms of neural networks, the two layer feed forward

network has been the most popular. This class of networks consists of two layers of nodes, namely the hidden layer and the output layer, and two layers of weights serving as the connections between the input and the hidden layer, as well as between the hidden layer and the output layer. No connection is allowed within its own layer and the information flow is one directional. Sigmoid functions are usually selected as the transfer function for hidden layer nodes and linear functions for the nodes of the output layer.

It has been shown that this class of neural network can approximate any functional (one-one or many-one) continuous mapping from one finite-dimensional space to another arbitrarily well, provided the number of hidden nodes is sufficiently large [36]. It has also been proven that, in the context of pattern recognition, this class of networks with sigmoidal non-linearity and two layers of weights can approximate any decision boundary to within arbitrary accuracy [37][38][39]. These properties of two layer feed forward neural networks lay the theoretical foundation for applying neural networks to process modeling and fault diagnosis by pattern recognition.

Neural networks can be applied to fault detection and diagnosis in two ways, performing as a process model or a pattern classifier. Numerous papers applying neural networks to fault detection and diagnosis can be summarized into three categories [40][41][42][43], as shown in Fig. 2.1. The first (Fig. 2.1(a)) is the use of a neural network to differentiate various fault patterns from normal operating conditions, according to different measured process output data. Training of the neural network can

be performed offline or online. In the second category, neural networks are used as classifiers to isolate faults represented by process model-generated residuals. The process model can be a mathematical model (Fig. 2.1(b)), based on which the fault diagnosis structure utilizes some process mechanism provided by the quantitative model, and therefore facilitates the implementation and training of the neural classifier. In cases where mathematical process models are not available, a neural network process model can be employed to generate residuals (Fig. 2.1(c)); another network is then used to isolate faults.

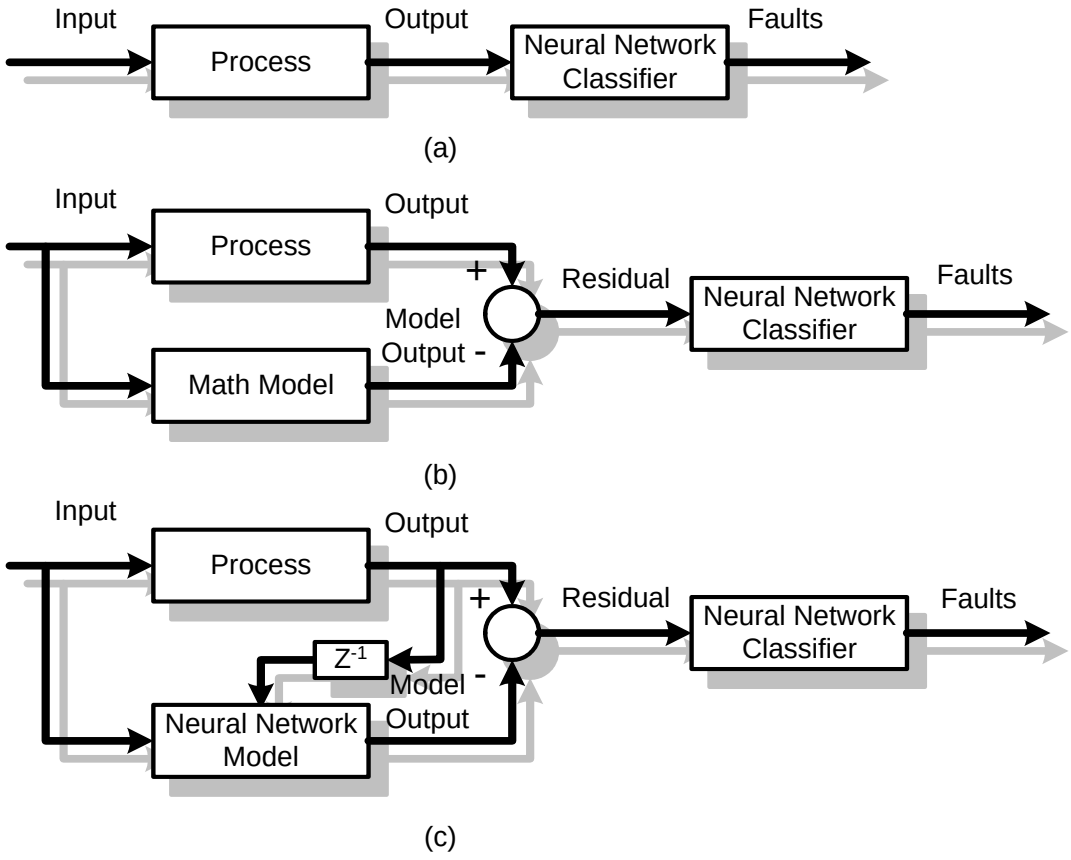


Fig. 2.1 Different approaches of applying neural networks to fault detection and diagnosis

The training session of the neural network uses the error in the output values to update the weights connecting layers, until the accuracy is within the tolerance level. The training time for a feed forward neural network using one of the variations of backpropagation can be substantial. For a simple 2-input 2-output system with 50 training samples, 100,000 training iterations are not uncommon [44]. For large-scale systems, memory and computation time required for training a neural network can exceed hardware limits. This has been a bottleneck in developing fault diagnosis algorithms for industrial applications. Like other data-driven methods, the performance of neural networks is determined by the available data. It is highly possible that neural networks will generate unpredictable outputs when presented with an input out of the range of the training data. This suggests that the neural networks need to be retrained when there is a slight change of the normal operation conditions, e.g., a molecular weight specification change in a polymerization reactor. This is not a big problem if the neural networks are trained offline then used online in fault diagnosis systems. However, in batch processes, due to the share of common equipment by many products and variability through batch-to-batch operations, operating regions are seldom constant. To maximize the utilization of equipment, batch processes are often tightly scheduled and this requires considerable amount of online retraining of the neural network process model and fault classifier. This reduces the potential application of neural network based fault diagnosis systems to batch processes.

In this section, a new neural network based fault diagnosis approach for batch processes is proposed to reduce the time required for training neural networks. A feed

forward neural network is employed to predict future process outputs by a series of past process inputs and outputs. Past process input/output data are fed into a preprocessor, which fits a polynomial to each series of input. Then the coefficients of the polynomials are fed into the network as inputs. A second neural network realized by a radial basis function (RBF) network is used as the classifier to classify faults. Compared with published approaches, this approach improves neural model training time without reducing accuracy, thus making on-line training possible.

2. Process fault detection and classification based on neural networks with feature extraction

2.1 Polynomial fitting as the feature extraction technique

Neural networks can be used as a process model to generate residuals for process monitoring. In this case the neural network is usually a feed forward recurrent neural network, which uses past process inputs and past process outputs to generate one-step-ahead prediction of process outputs, as shown in Fig. 2.2 and the following formulation.

$$\mathbf{y}(t+1) = \text{neural network } [\mathbf{y}(t), \mathbf{y}(t-1), ...\mathbf{y}(t-d), \mathbf{u}(t), \mathbf{u}(t-1), ...\mathbf{u}(t-d)] \quad (2.1)$$

where

$\mathbf{y}(t+1)$: predicted process output vector

$\mathbf{y}(t), \mathbf{y}(t-1), ...\mathbf{y}(t-d)$: the past process output vector sequence

$\mathbf{u}(t), \mathbf{u}(t-1), ... \mathbf{u}(t-d)$: the past process input vector sequence

d : the length of the sample sequence

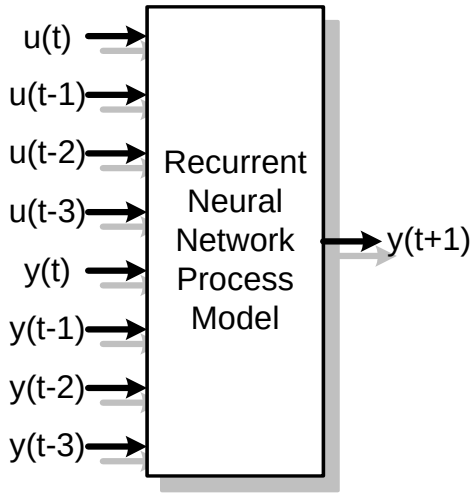


Fig. 2.2 Recurrent neural network process model

Since variables in chemical processes are changing continuously, the past process input and output sequences are not independent. Therefore the possibility exists for using a lower dimensional representation of the sequences and to realize dimensionality reduction. Due to the low frequency and large time constant nature of chemical process dynamics, least squares polynomial fitting is proposed as the dimensionality reduction technique, as shown in Figure 2.3. As one can see in Figure 2.3, a process input sequence consisting of seven numbers is represented by a second order polynomial, which can be described with three coefficients. Therefore the dimension reduction from seven to three is realized. An added benefit of this feature extraction process is that it smoothes the original data and provides some degree of noise reduction.

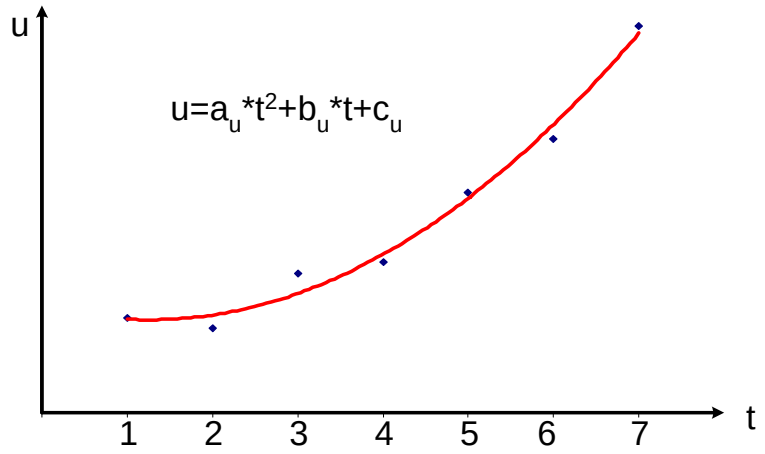


Fig. 2.3 Polynomial fitting as a dimensionality reduction technique

The new structure of the neural network process model is shown in the following formulation:

$$\begin{aligned} \mathbf{a}_y &= \text{polynomial fitting } [\mathbf{y}(t), \mathbf{y}(t-1), ... \mathbf{y}(t-d)] \\ \mathbf{a}_u &= \text{polynomial fitting } [\mathbf{u}(t), \mathbf{u}(t-1), ... \mathbf{u}(t-d)] \\ \mathbf{y}(t+1) &= \text{neural network } [\mathbf{a}_y, \mathbf{a}_u] \end{aligned} \qquad (2.2)$$

where

$\mathbf{a}_y$: the vector of polynomial coefficients fitted on the past process output sequence

$\mathbf{a}_u$: the vector of polynomial coefficients fitted on the past process input sequence

2.2 Determine the optimal structure of the neural network process model

It has been shown that the length of the process input and output sequences provided to the neural network model has a significant impact on the model

performance [45]. Cross validation is employed to find the optimal model structure. The training dataset is randomly split into N sections, and one of these subsets is reserved for use as an independent test dataset, while the other N-1 subsets are combined for use as the training dataset. Various model structures with different lengths of process input and output sequences are tested, and each model structure is trained N times, with a different subset of the data reserved for use as the test dataset each time. Thus N different process models are produced, each of which is tested against its corresponding test dataset. The average performance of the N models is an excellent estimate of the performance of the original model (produced with the entire training dataset) on a future independent set of test dataset. The average performances of these models with different structures are compared and the one with the lowest prediction error is selected as the optimal structure. It is worth mentioning that besides cross validation, more advanced nonlinear model validation approaches such as correlation analysis [46] can also be used.

2.3 Selection of the optimal order of the polynomial fitting preprocessor

After the optimal length of process input and output sequences is found, the next step is to determine the optimal order of the polynomial to best represent the sequences by the polynomial coefficients. This again can be done with cross validation. A group of process input and output sequences are selected to represent the whole neural network training dataset. Suppose that each sequence contains d samples of a variable, each sequence is divided into d subsets with one sample in each subset. Polynomials of order

2, 3 and 4 are used to fit the combination of *d*-1 subsets and the last subset is used for testing. The average prediction errors of the polynomials with different order are compared and the one with the smallest error is selected as the optimal preprocessor.

After the optimal structure of the neural network process model and the optimal order of the preprocessor are found, a new neural network process model with reduced dimensionality can be constructed, as shown in Figure 2.4.

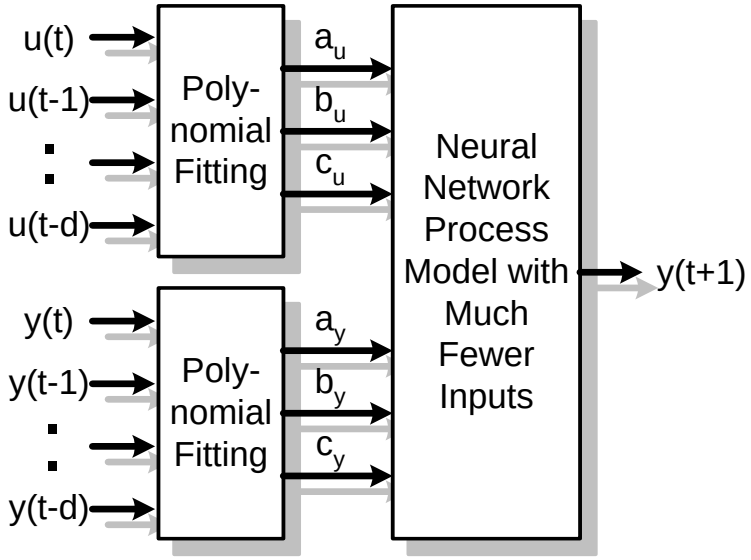


Fig. 2.4 Neural network process model with dimensionality reduction

2.4 Fault classification with radial basis function neural network

Fault diagnosis is implemented with a radial basis function neural network functioning as a classifier. The residual vector generated from the process output and neural network prediction has different structures corresponding to different faults. A neural network classifier uses this feature to isolate faults.

The radial basis function (RBF) network used here consists of two layers: a hidden layer and an output layer. The hidden layer contains a number of RBF neurons, and each of them represents a single radial basis function. The transfer function of the hidden layer neurons is a radial basis function, in the form of $\phi(n) = \exp(-n^2)$. The most popular RBF is a Gaussian type that is characterized by a center ($\mathbf{c_j}$) and a width (r_j). The RBF functions by measuring the Euclidean distance between input vector ($\mathbf{x}$) and the center ($\mathbf{c_j}$), then performs the nonlinear transformation as given in the transfer function of the *j*th hidden layer neuron below:

$$\phi_j(\mathbf{x}) = \exp\frac{-\left\|\mathbf{x}-\mathbf{c_j}\right\|^2}{r_j^2} \tag{2.3}$$

The output layer transfer function is linear, which is given in Eq. (2.4):

$$y_k(\mathbf{x}) = \sum_j w_{kj}\phi_j(\mathbf{x}) + b_k \tag{2.4}$$

Where y_k is the output of the *k*th output layer neuron, w_{kj} is the weight of connection between *j*th hidden layer neuron and the *k*th output layer neuron, ϕ_j is the output of the *j*th hidden layer neuron, and b_k the bias of the *k*th output layer neuron. From the formulation of transfer functions, one can see that after choosing RBF centers and widths, designing the RBF network is simply solving a linear system. There are several methods to choose the centers, such as *K*-means clustering algorithm [47], orthogonal least-squares algorithm [48]. The widths of RBFs can be chosen as the same,

or different for each neuron. In this research, Gaussian type RBF is used as the hidden layer transfer function and the width of every RBF is chosen as the same. Orthogonal least-squares algorithm is used to select the centers from the training set. The connection weights between the hidden layer and the output layer are obtained by solving the linear system using least squares method.

3. Case study 1: a batch polymerization reactor

3.1 Process simulation

The process studied is a batch polymerization reactor for the production of polymethylmethacrylate (PMMA) from methylmethacrylate (MMA) and azobisisobutyronitrile (AIBN). In the stirred tank reactor, MMA as monomer and AIBN as initiator are reacting in toluene solvent. The process is optimized to produce as much PMMA of weight-average molecular weight of 400,000 as possible, under the given restrictions: production rate of PMMA 0.10 kg/h; initial volume fraction of solvent 0.30; loading/startup time 1.0 hour; and batch cycle time 6.0 hours. The reactor is controlled by the net heat added to the jacket using a combination of cooling water and electric heaters. The control input profile described in the literature [49] will be used as the desired input trajectory. The detailed formulation of the kinetic model is omitted here since this process has been studied extensively in the literature [50].

Based on the analytical model found in the literature, a simulation program is developed and used to study the neural network-based fault diagnosis approach. The

most significant variables in the simulation are concentration of MMA (C_m), concentration of AIBN (C_i), reactor temperature (T), jacket temperature (T_j), and electric heater power (u). Since the development of detailed mechanistic models for complex industrial processes is usually difficult or even impossible, empirical models based on operating data should be used. When developing the fault diagnosis methods for this reactor, it is assumed that the dynamic process model is unknown and neural network models are developed.

3.2 Conventional neural network process model

To build neural network models for the polymerization reactor, data from ten batch runs are generated from the process simulation (treated as real process runs in this paper). Those data contain simulated measurements of C_m, C_i, T, T_j, and u. Since it is generally difficult to obtain concentration of a material in real time, it is assumed that measurements of T, T_j, and u are obtained online while the measurements of C_m and C_i are obtained offline. This simulates the current industrial practice that many quality variables of polymerization processes are still measured offline through laboratory analysis. The data are generated by adding random perturbations to the nominal control input profile (u). Random variations are added to the control variable to reflect the different skill levels of different operators. Normally distributed random noises with zero means are also added to all the measurements to simulate sensor noises. The standard deviations of the noises for u, T, T_j, C_m and C_i are 0.02 kW, 0.5 K, 0.5 K,

0.002 kmol/m^3, 0.001 kmol/m^3 respectively. Figs. 2.5 and 2.6 illustrate the effect of adding noise to MMA concentrations. The sampling time is 1.5 minutes and each run takes 6 hours.

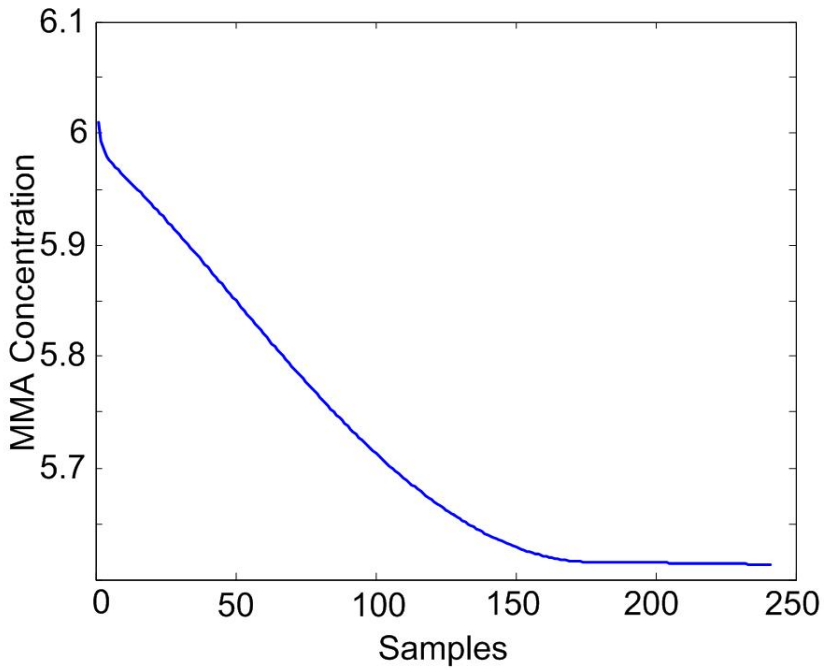


Fig. 2.5 MMA concentration before adding the white noise

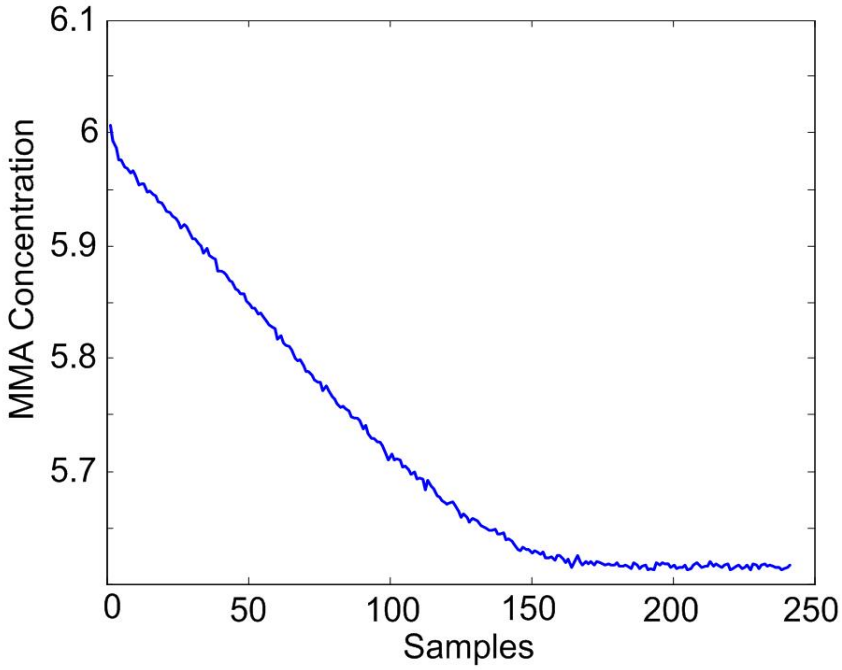


Fig. 2.6 MMA concentration after adding the white noise

A multi-layer feed forward neural network with one hidden layer is used to develop a nonlinear representation of the polymerization reactor based on recurrent network structure described in the literature [43]. Feed forward neural networks are implemented with the Matlab Neural Network Toolbox. The details of the training and optimization process are shown below:

- The model outputs are T , T_j at sampling time t and the model inputs are u, T , and T_j at sampling time t-1, t-2, … t-n (n being the time lag to be optimized);
- All training data are scaled to the range of [0, 1] prior to the training process;

- Neural network weights are initialized to zero-mean random values;
- The neural network weights are initialized as random numbers in the range (-0.1, 0.1);
- The networks are trained using a Levenberg-Marquardt optimization algorithm [51];
- The number of hidden layer neurons in each network is optimized such that the minimum training error is achieved;
- Number of epochs trained: 1000;
- The error in training is computed with mean squared error, and the error in validation is computed with average relative prediction error (ARPE);
- Output layers have two log-sigmoid neurons.

A number of networks with different structures are trained and tested. The number of hidden layer neurons in each network is optimized such that the minimum training error is achieved. Cross validation is employed to find the optimal network structure. The network with the smallest average relative prediction error on the validation sets is considered as the optimal model. The following optimal model structure is obtained and it has eight hidden neurons:

$$\begin{aligned}[T(t), T_j(t)] = \text{neural network}[&T(t-1), T(t-2), ..., T(t-6), \\ &T_j(t-1), T_j(t-2), ..., T_j(t-6), u(t-1), u(t-2), ..., u(t-6)]\end{aligned} \tag{2.5}$$

Since u, T, and T_j can be measured in real time, this model can be used to generate one-step ahead prediction of T and T_j. Model input structures can have a great impact on model accuracy. In this example, the time lags of inputs and outputs are determined by validation results. Different model structures are evaluated and their average relative prediction errors (ARPE) on the test data are compared, as shown in Table 2.1. The result indicates that increasing the number of model inputs may not necessarily improve model performance. For example the last network in the table has more inputs than any other network and its error is also larger than the rest of the networks. This is probably due to the fact that a large number of inputs results in the possibly over-fitting the data.

Table 2.1 ARPE of various model structures

Input variables	ARPE$\times 10^3$
$T(t-1),...,T(t-3),T_j(t-1),...,T_j(t-3),u(t-1),...,u(t-3)$	3.99
$T(t-1),...,T(t-3),T_j(t-1),...,T_j(t-6),u(t-1),...,u(t-6)$	2.52
$T(t-1),...,T(t-6),T_j(t-1),...,T_j(t-3),u(t-1),...,u(t-6)$	3.69
$T(t-1),...,T(t-6),T_j(t-1),...,T_j(t-6),u(t-1),...,u(t-3)$	1.87
$T(t-1),...,T(t-6),T_j(t-1),...,T_j(t-6),u(t-1),...,u(t-6)$	1.10
$T(t-1),...,T(t-9),T_j(t-1),...,T_j(t-6),u(t-1),...,u(t-6)$	3.81
$T(t-1),...,T(t-9),T_j(t-1),...,T_j(t-9),u(t-1),...,u(t-6)$	2.29
$T(t-1),...,T(t-9),T_j(t-1),...,T_j(t-6),u(t-1),...,u(t-9)$	4.68

3.3 Feature extraction

To find the optimal degree of dimensionality reduction, 15 sequences of T, T_j, and u are collected to represent the whole data set, with five sequences for each variable and six consecutive samples in each sequence. 2nd order, 3rd order, and 4th order polynomials are tested to describe these sequences. With the aid of cross validation, the average relative prediction error of each type of polynomial on each variable is calculated, as shown in Table 2.2. It is found that 2nd order polynomial best describes the sequences of T and T_j but for u 3rd order is optimal. For simplicity, a 2nd order polynomial is chosen as the feature extraction preprocessor.

Table 2.2 ARPE of various types of polynomials on each variable

ARPE	2nd order	3rd order	4th order
T	4.23	5.41	6.10
T_j	4.12	5.84	6.49
u	5.75	4.50	7.86

The new neural network model has the following structure:

$$
\begin{aligned}
&[a_T(t-1), b_T(t-1), c_T(t-1)] = \text{2nd order polynomial fitting}[T(t-1), T(t-2), ..., T(t-6)] \\
&[a_{Tj}(t-1), b_{Tj}(t-1), c_{Tj}(t-1)] = \text{2nd order polynomial fitting}[T_j(t-1), T_j(t-2), ..., T_j(t-6)] \\
&[a_u(t-1), b_u(t-1), c_u(t-1)] = \text{2nd order polynomial fitting}[u(t-1), u(t-2), ..., u(t-6)] \\
&[T(t), T_j(t)] = \text{neural network}[a_T(t-1), b_T(t-1), c_T(t-1), \\
&\qquad a_{Tj}(t-1), b_{Tj}(t-1), c_{Tj}(t-1), a_u(t-1), b_u(t-1), c_u(t-1)]
\end{aligned}
\tag{2.6}
$$

A feed forward neural network is built based on the new structure. This network shares the same number of neurons in the hidden layer and the output layer with the obtained optimal model. The original training data are fitted to polynomial coefficients and then fed to the network as new training data. The new network is trained and tested with the same number of epochs of training as the optimal network developed in the previous section. The training time and ARPE are compared and shown in Table 2.3.

Table 2.3 Comparisons of training time and ARPE of neural process models with and without feature extraction

Model structure	Training time (s)	ARPE$\times 10^3$
Feed forward neural net	534.2	1.10
Feed forward neural net with input feature extraction	70.7	1.52

Table 2.3 indicates that the new model structure drastically reduced training time by more than seven folds compared to previously reported methods [43], for a slight sacrifice (ARPE increases by 0.04 percentage point) in model performance. This is probably due to the fact that introducing input feature extraction results in a system with more non-linearity. Absolute one-step-ahead prediction errors of the neural network models with or without input feature extraction are shown in Fig. 2.7, which illustrates the slight increase of prediction error by introducing input feature extraction.

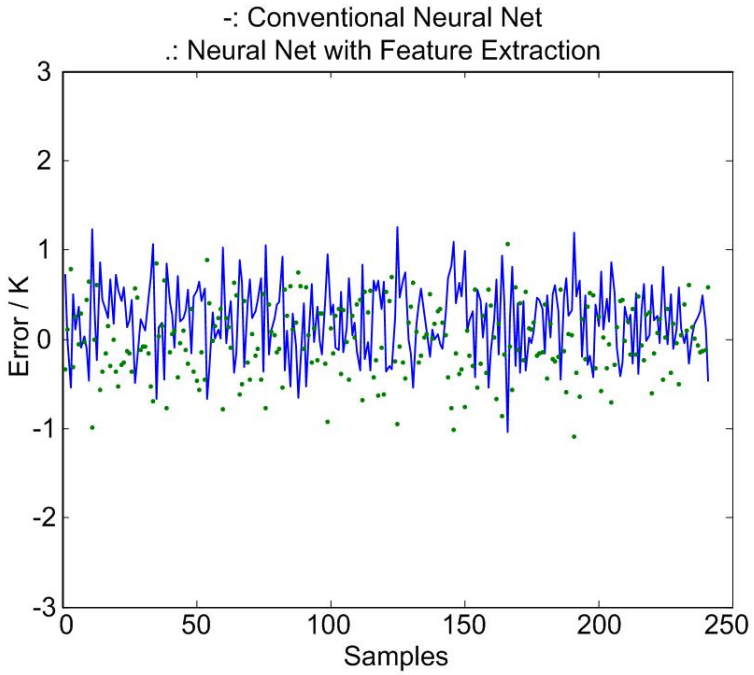


Fig. 2.7 One-step-ahead prediction errors of the neural network model with and without input feature extraction

3.4 Simulation of faults

Two types of faults are studied in this section. One is the sensor fault *T*, which is simulated by superimposing a –10% change of the measured temperature; the other is fouling of the reactor jacket, simulated by a –10% change to B, the parameter in jacket heat transfer coefficient. The analytical process model is modified accordingly to simulate faulty batch runs. Five batches are simulated with two faults occurring, four of them used as training and one as validation set. To ensure that the faults can be detected whenever they occur, five different combinations of occurrence are selected as following.

Training set:

–10% change of T @ t = 0 - 50; –10% change to B @ t = 101-150;

–10% change of T @ t = 101-150; –10% change to B @ t = 0 – 50;

–10% change of T @ t = 101-150; –10% change to B @ t = 151-200;

–10% change of T @ t = 101-150; –10% change to B @ t = 101-150

Validation set:

–10% change of T @ t = 51-100; –10% change to B @ t = 201-250

3.5 Residual generation

The trained feed forward neural network with input feature extraction is used as the fault detection residual generator. The residual is designed as the difference between the real process output and the neural network model output. Since only T and T_j are available in real time, C_m and C_i are not included in the residual.

3.6 Network training and simulation

An RBF network classifier is developed with the Matlab Neural Network Toolbox. It has two inputs (corresponding to the dimension of the residual), two outputs (corresponding to two types of faults to be classified). Residuals generated with the neural network process model and the faulty process simulation are fed into the classifier as training input. The training target is set to zero for both of the classifier outputs for normal operating condition, and to one for a specific fault, with the other

remaining zero. In this way, a classifier output of [1, 0] indicates that the T sensor fault is occurring while [0, 1] indicates that cooling jacket fouling is occurring.

The classifier is then trained with the residuals as inputs against the desired outputs. The details of the training process are shown below:

- The networks are trained using Orthogonal Least Squares Learning Algorithm;
- Performance goal (mean squared error): 0.01;
- Spread constant (RBF width): 1.0;
- Maximum number of neurons: 50;
- The hidden layer contains Gaussian activation function;
- The output layer contains linear activation function;
- The trained network has 26 hidden layer neurons.

The trained classifier is tested using another batch with two faults occurring consecutively. The two outputs of the neural classifier are shown in Figs. 2.8 and 2.9. It is clear that all two faults have been detected and isolated. The fact that classifier outputs are not zero when no fault occurs is due to errors of the neural model and noise.

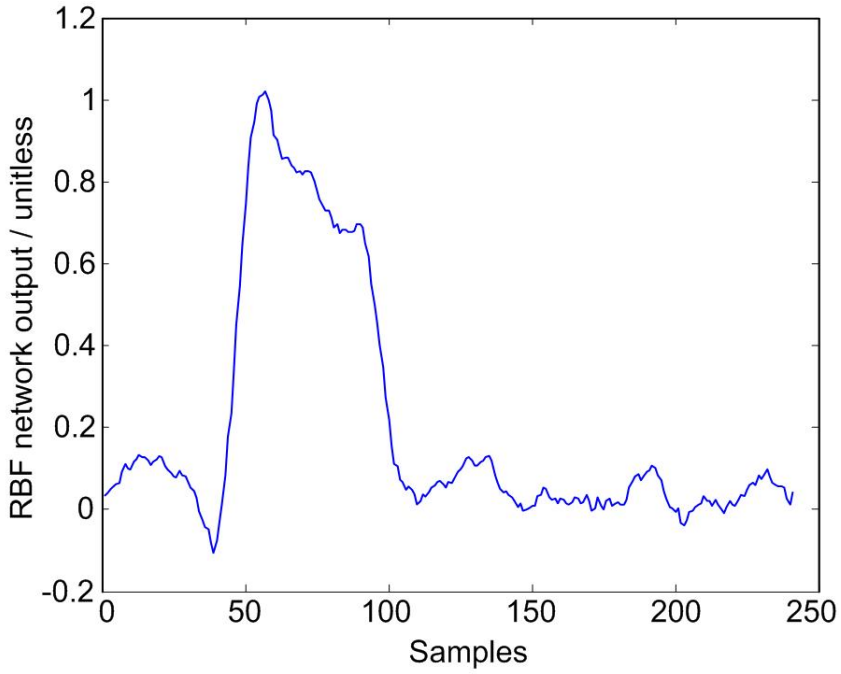


Fig. 2.8 RBF network output 1

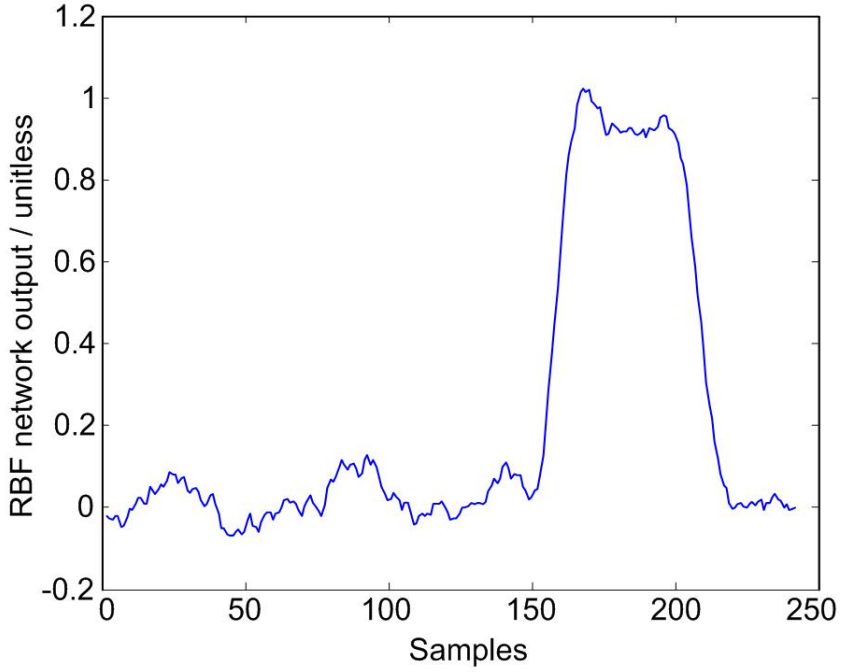


Fig. 2.9 RBF network output 2

4. Case study 2: a pilot distillation column

4.1 Process description and data acquisition

The pilot size distillation column studied in this work (flow scheme shown in Fig. 2.10) is approximately 60 feet high, six inches in diameter and is insulated. It separates a binary mixture of methanol and water using three sets of structured-packing (15 theoretical plates under normal operating conditions) with distributors above each set of packing. The reflux, feed and product flows are all measurable, and the column temperature and pressure profiles can be established from thermal couples and differential pressure measurements along the column. The process is controlled with a distributed control system with an automatic data acquisition system. All process data are recorded to a Microsoft Excel worksheet at an interval of 20 seconds.

Two kinds of faults are induced in this study: a “sticky control valve” fault (fault 1) is induced by setting the corresponding column temperature control loop to manual; a “vapor bypassing” fault (fault 2) is induced by opening the second (lower) feed line to the column. These simulated faults are very realistic in distillation processes.

Nine process variables are chosen to represent the states of the process: column temperatures at theoretical plates 5, 6, 7, and 8; feed low rate; feed temperature; reflux flow rate; reflux temperature; and steam condensate flow.

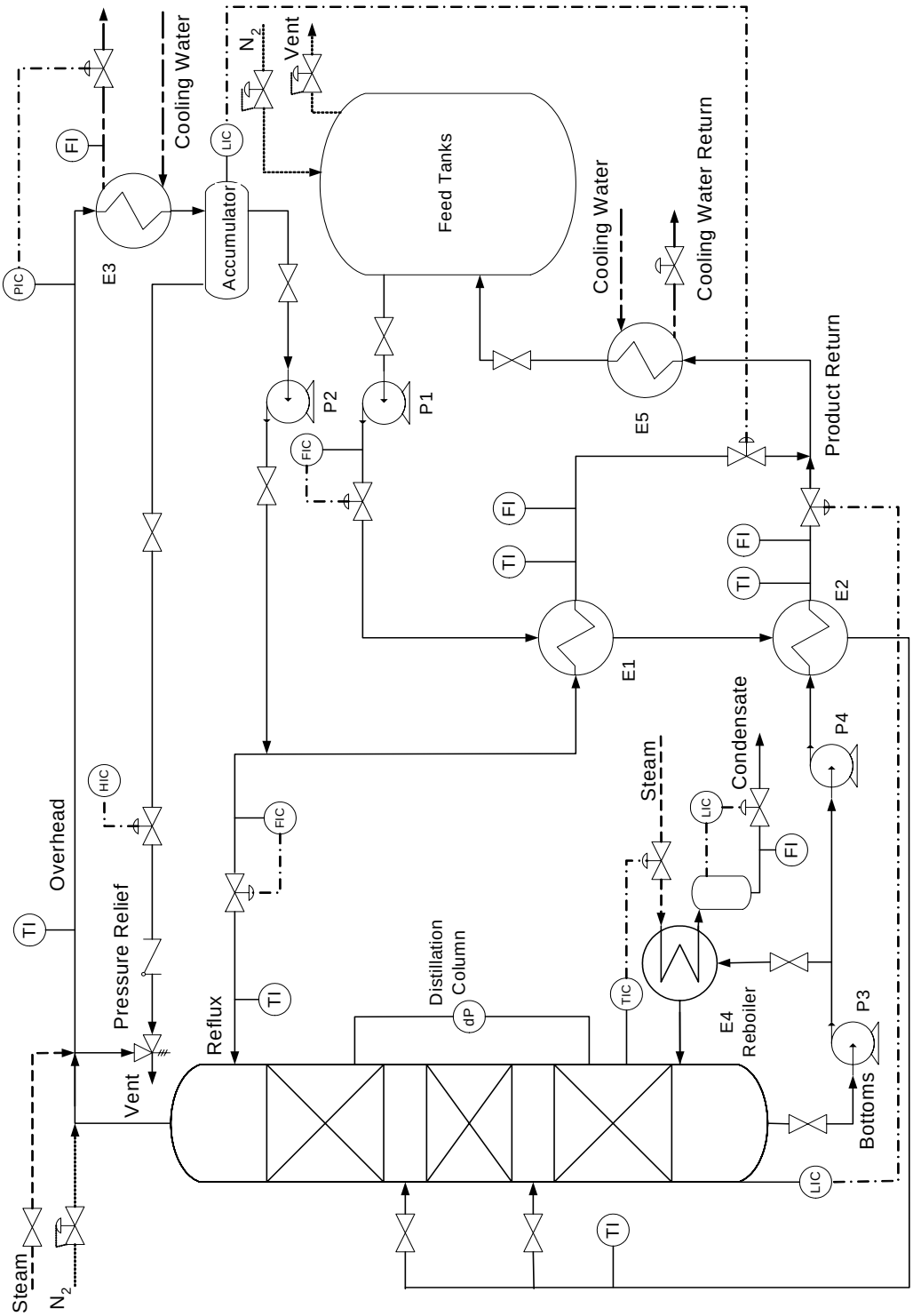


Fig. 2.10 Schematic of the pilot size distillation process [52]

569 points of normal process data, 154 points of “sticky valve” faulty process data, and 115 points of “vapor bypassing” faulty process data are recorded in a run. Another 20 points of “sticky valve” faulty process data, and 28 points of “vapor bypassing” faulty process data are recorded in a separate run, which are to be used as testing data.

4.2 Conventional neural classifier approach

Since the distillation column is operated at steady state, the process model is no longer needed to provide the reference to generate residuals. Instead, a simpler approach as illustrated in Fig. 2.1(a) is employed to differentiate various fault patterns from normal operating conditions, according to different measured process output data. This case study will show that the proposed feature extraction approach is also applicable in this circumstance.

Two output nodes are used to classify three classes of process data, i.e. normal, fault 1, and fault 2. The first output node is to produce a unit value when the inputs belong to fault 1 and zero when the inputs fall into the fault 2 category. Similarly, the second output node produces a value of one for inputs from the category including fault 2 and zero when presented with measurements from fault 1 type.

A multi-layer feed forward neural network with one hidden layer is used to develop a nonlinear classifier. Feed forward neural networks are implemented with the Matlab Neural Network Toolbox. The details of the training and optimization process are shown below:

- Neural network inputs are the nine process variables shown above at sampling times t - 1, t - 2, … t - 6 (optimized with cross validation); Thus the neural network input contains $9 \times 6 = 54$ scalars;
- All training data are scaled to the range of [0, 1] prior to the training process;
- The neural network weights are initialized as random numbers in the range (-0.1, 0.1);
- The networks are trained using a Levenberg-Marquardt optimization algorithm;
- Number of epochs trained: 200;
- The hidden layer has eight sigmoid neurons (optimized such that minimum training error is achieved);
- The output layer has two linear neurons.

The training process takes 639.31 seconds on a Pentium III 600MHz PC and reaches a training error of 1.97×10^{-9}. The test data are then fed into the trained network and the classifier output are shown in Fig. 2.11 and Fig. 2.12.

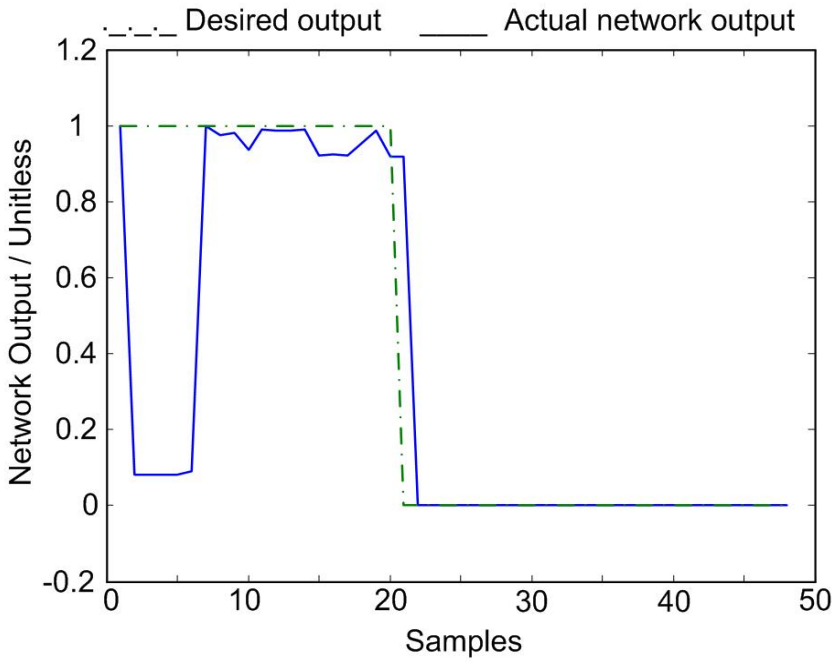


Fig. 2.11 Output 1 of the conventional neural classifier: dash-dot line represents desired values and solid line is the actual classifier output

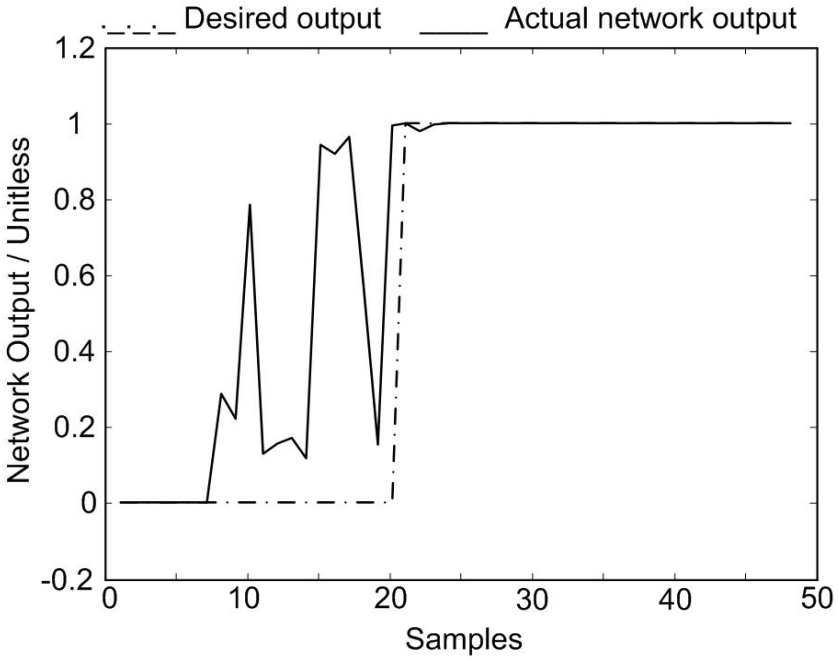


Fig. 2.12 Output 2 of the conventional neural classifier: dash-dot line represents desired values and solid line is the actual classifier output

As can be seen from the figures, the actual neural classifier outputs will lie close to the values desired during training in subsequent use, while perhaps not matching them exactly. Thus, thresholds are specified that define the process state. A fault is deemed active if the corresponding output is greater than 0.7 and inactive if below 0.3. Any output lying between these two thresholds may be the result of the corresponding data not being definite members of a pre-defined category.

Based on the thresholds defined above, the accuracy of the conventional neural classifier is 43 out of 48 for fault 1, and 42 out of 48 for fault 2.

4.3 Neural classifier with feature extraction

The details of the training and optimization process of the neural classifier with feature extraction are shown below:

- Each series (sampling time t - 1, t - 2, … t - 6) of the nine process variables is fitted with a second order polynomial (order optimized with cross validation), then the coefficients of the polynomials ($9 \times 3 = 27$ numbers) are fed to the neural network as input;
- All training data are scaled to the range of [0, 1] prior to the training process;
- The neural network weights are initialized as random numbers in the range (-0.1, 0.1);

- The networks are trained using a Levenberg-Marquardt optimization algorithm;
- Number of epochs trained: 200;
- The hidden layer has eight sigmoid neurons;
- The output layer has two linear neurons.

The training process takes 194.63 seconds on the same Pentium III 600MHz PC and reaches a training error of 1.03×10^{-13}. The test data are then fed into the trained network and the classifier output are shown in Figs. 2.13 and 2.14.

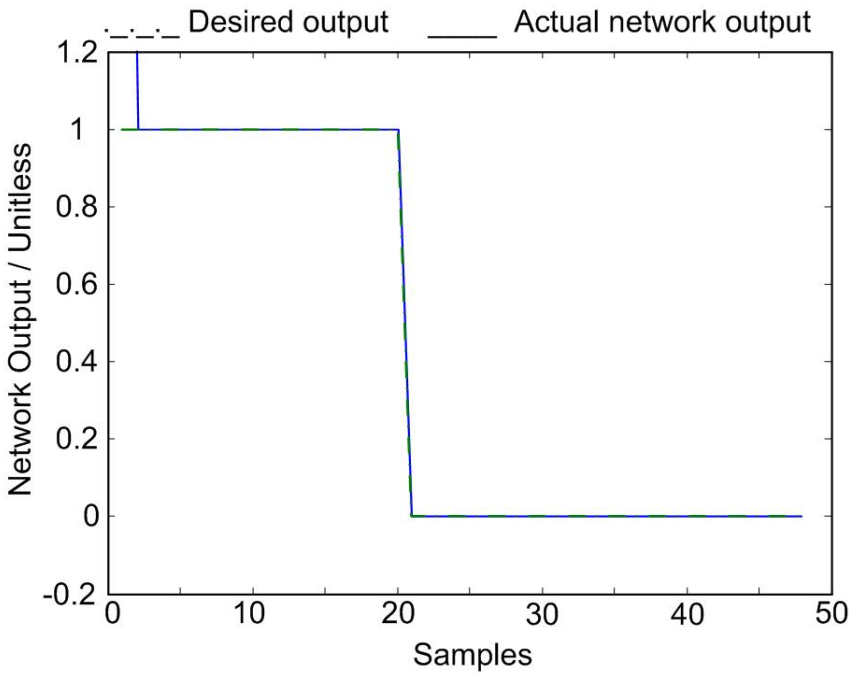


Fig. 2.13 Output 1 of the neural classifier with feature extraction: dotted lines represent desired values and solid lines are the actual classifier output

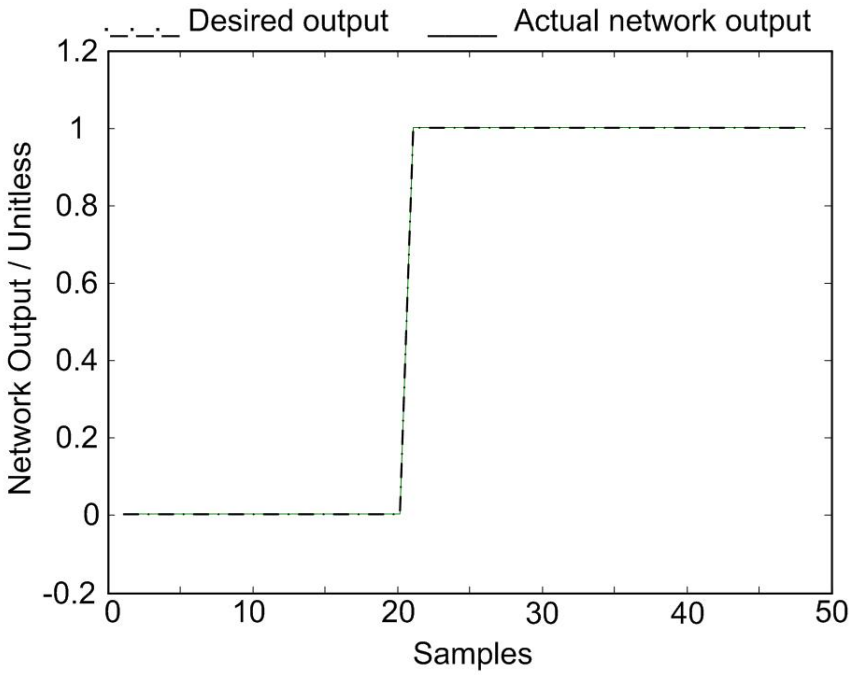


Fig. 2.14 Output 2 of the neural classifier with feature extraction: dotted lines represent desired values and solid lines are the actual classifier output (two lines perfectly overlap)

The accuracy of the neural classifier with feature extraction is 47 out of 48 for fault 1, and 48 out of 48 for fault 2. As can be seen from the results, not only is the training time drastically reduced (about 70%), but the classifier accuracy is also greatly improved.

5. Conclusions

Fault diagnosis for a batch polymerization process and a pilot size distillation process is investigated in this work by a feed forward neural model and a radial basis function neural classifier. Three approaches of applying neural networks to process fault diagnosis are discussed.

In batch processes, operating regions are seldom constant and this may require considerably amount of re-training of the neural process model. This reduces the potential of neural network based fault diagnosis systems applied to batch processes. In this work, by implementing an input feature extraction process for the neural model, the training time required for the neural process model is drastically reduced at the cost of a slight deterioration in model performance.

In the first case study of a batch reactor, a temperature sensor fault and reactor jacket fouling are studied and classified successfully using the neural process model and an RBF neural classifier. In the second case study of a pilot size distillation process, a control valve fault and vapor bypassing are classified with better accuracy and reduced training time of the neural classifier. This is a general approach that can be applied for batch processes fault diagnosis using neural networks.

III. FAULT DETECTION AND CLASSIFICATION BASED ON CLASSIFICATION TREE AND DISCRIMINANT ANALYSIS

1. Introduction

Classification and regression tree (CART) of Breiman et al [53] is an innovative methodology for the analysis of large data sets via binary partitioning procedure. Classification tree analysis has the advantage of being concise, fast to compute, and making no assumption regarding the distribution of the predictor variables.

This section will develop a general process monitoring scheme applicable for multivariate chemical processes based on classification trees. This technique, referred to as classification tree with discriminant analysis, integrates classification tree and Fisher Discriminant Analysis (FDA). FDA is employed here to reduce the dimensionality of the process measurements and to produce an optimal lower dimensional representation of the process in terms of discriminating between classes. The classification tree is then applied to the scores produced by FDA and fault classification is performed. The proposed technique has better performance compared with the original classification tree in terms of classification accuracy and training time.

In what follows, the principles of classification tree analysis are reviewed and then the new methodology is derived. Two case studies, the Tennessee Eastman Process simulation and a pilot size distillation column, are presented next in order to compare

the results with other published methodologies and to demonstrate the effectiveness of the new approach. Conclusions are drawn in the final section.

2. Classification tree analysis principles

Classification trees were first introduced by social scientists in the early 1960s [54]. Breiman et al [53] formulated classification trees in statistics in the 1980s and proposed the Classification and Regression Tree (CART) methodology, on which the classification tree building in this study is primarily based.

2.1 Growing a classification tree

CART is a form of *binary recursive partitioning*, that is, in the process of progressive splitting the set of training samples into smaller and smaller subsets, each subset can only have two branches, and this partitioning can be carried on several times. Fig. 3.1 shows the classification tree resulting from analysis of a set of process data. The training set contains three known classes: normal, fault A, and fault B. At first, all samples are assigned to one node (Node 1) and CART looks for the most informative property to separate samples, in this case this is the pressure. The samples in Node 1 are thus divided into two groups (Node –1 and Node 2), and CART continues to look for the best property to separate each group. For example, in Node 2, CART seeks a property that makes the samples reaching the immediate descendent nodes as “pure” as possible, and it finds that separating the samples by temperature will generate “purer” child nodes than if pressure were used. Thus, the original data set has been split into three pure sets and the partition stops.

The fundamental principle of CART is simple: simple decision trees with fewer nodes are preferred. That is the reason why CART looks for the most informative property that makes the data reaching the immediate child nodes as “pure” as possible, or as less “impure” as possible.

Several different mathematical measures of tree cost (impurity) have been proposed, all of which have approximately the same behavior. Let $i(t)$ denote the impurity of a node t.

One measure is the entropy impurity index:

$$i(t) = -\sum_j p(j \mid t) \log_2 p(j \mid t) \tag{3.1}$$

where $p(j \mid t)$ is the portion of observations in node t belonging to class j. It is obvious that if all the observations are of the same class, the impurity index is 0; otherwise it is positive and the maximum value occurs when the different classes are equally possible.

Another impurity measure, the Gini diversity index [53], describes the expected error at node t if the node label is selected randomly from the class distribution present at that node.

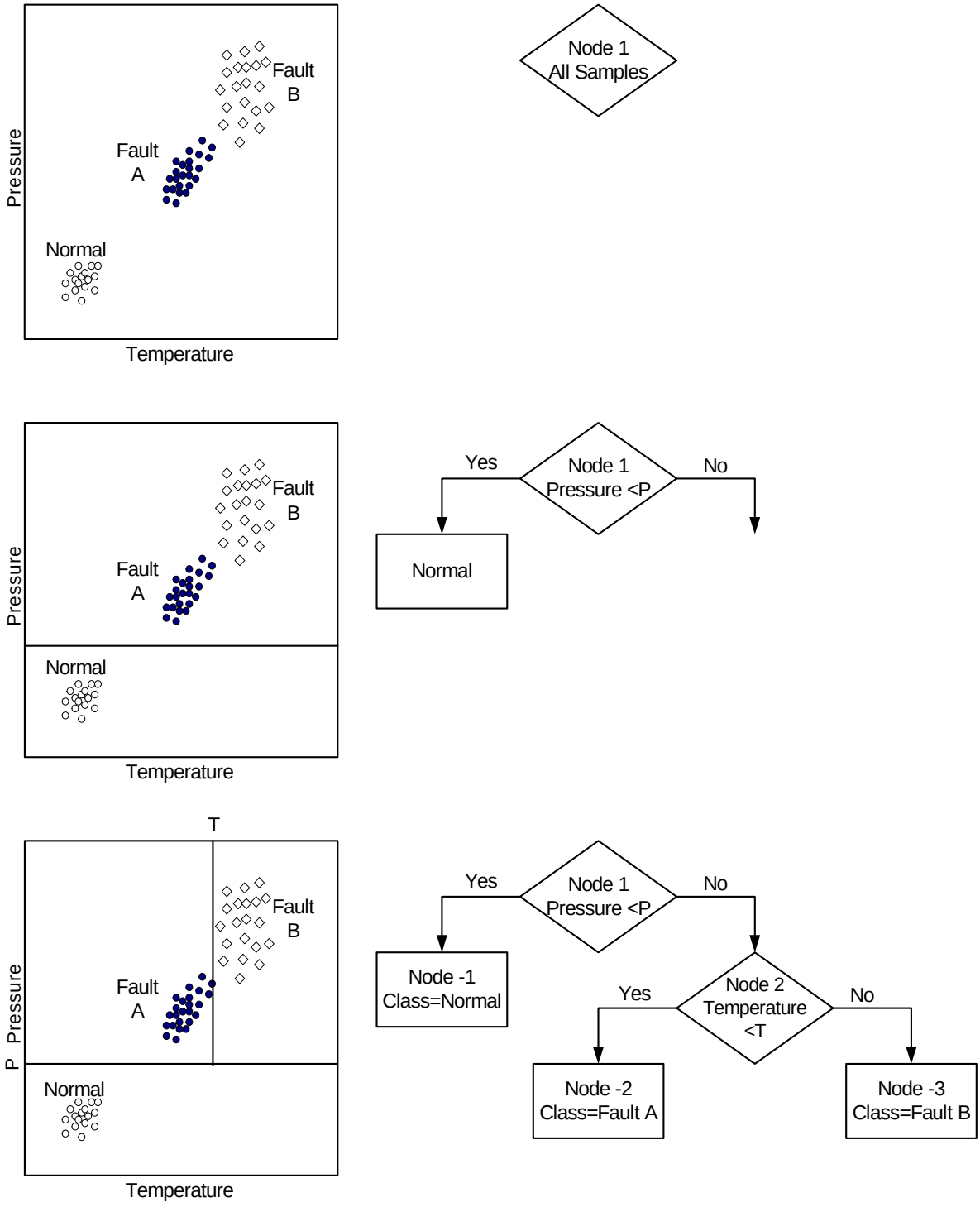


Fig. 3.1 Example of a simple classification tree

The node cost is formulated as

$$i(t) = \sum_{i \neq j} p(i \mid t)\, p(j \mid t) \tag{3.2}$$

where

$i(t)$: Gini diversity index of node t

$p(i \mid t)$: the portion of observations in node t belonging to class i

$p(j \mid t)$: the portion of observations in node t belonging to class j

Given a partial tree down to node t, the best property one choose to split further is the one that generates the lowest impurity index. The decrease in impurity index is formulated as

$$\Delta i(t) = i(t) - p(i \mid L) i(t_L) - [1 - p(i \mid L)]\, i(t_R)$$

where t_L and t_R are the left and right child nodes of t

$p(i \mid L)$: the portion of observations at node t that goes to t_L

$i(t_L)$, $i(t_R)$: impurity indices of node t_L and t_R

Although this optimization is performed at a single node, the recursive splitting process can go on until each leaf node becomes perfectly pure, in the extreme case, each leaf node corresponds to a single training sample. In this case, the tree impurity measure, as defined below, equals 0.

$$I(T) = \sum_{t \in \tilde{T}} i(t) \times p(t) \tag{3.3}$$

where

$\tilde{T}$: the set of terminal nodes (i.e., leaves) of tree T

T : the classification and regression tree

$p(t)$: the portion of observations in node t out of all observations

2.2 Deciding the optimal tree size

If the classification tree is grown fully until each leaf node achieves zero impurity, then the tree typically over-fits the training data and thus cannot be expected to work well with new noisy data. However, if partitioning is stopped too early, the error on the training data is not low enough and the performance on new data may not be sufficiently good.

How shall one find the optimal tree size (depth)? One traditional approach is to use cross-validation. The training dataset is randomly split into N subsets, and one of these subsets is reserved for use as an independent test dataset, while the other N -1 subsets are combined for use as the training dataset. Trees with different sizes are tested, and at each size, N trees are generated, with a different subset of the data reserved for use as the test dataset each time. Thus N different trees are produced, each of which is tested against its corresponding test dataset. The average performance of the N trees is an excellent estimate of the performance of the original tree (produced with the entire training dataset) on a future independent set of test dataset. The average

performance of these trees of different sizes is compared and the one with the lowest prediction error is selected as the optimal tree size.

Another method is to set a threshold value in the reduction in node impurity index. Node splitting is stopped if the best split results in an impurity reduction less than this threshold value. This method utilizes all the training data, and tends to generate classification trees with balanced leaf node impurity. The major disadvantage is that it is often difficult to determine the threshold value, because the relationship between the threshold value and the tree performance is rarely simple.

The cross-validation or threshold method work well and will find the optimal tree if the greatest impurity reduction occurs near the root node and the reduction in impurity reduces as the splitting goes on. However, this is not true for all occasions. Sometimes the most significant classification takes place near the leaf nodes and stopping splitting early will not achieve overall optimal accuracy.

The alternative approach of generating the optimal classification tree to stop splitting is “Pruning”. A tree is grown fully, i.e., all leaf nodes are of perfect purity. Then, all pairs of neighboring leaf nodes are considered for elimination. Any pair whose elimination yields a satisfactory (small) increase in tree cost is deleted, and their common antecedent node becomes a leaf.

The benefit of pruning is that it grows the tree to its full depth thus reduces the chance of missing significant classification functions close to the leaves. Pruning is frequently used with cross-validation to determine the best degree of pruning, which

comes at a huge computational expense. For large problems with high dimensionality and large number of training data, the computational cost could be significant.

2.3 Assigning leaf node labels

Assignment of class labels to the leaf nodes is straightforward. If the classification tree has been grown completely and pruned to its optimal depth, it is most likely that each leaf node has zero or a very small positive impurity index. If a leaf node contains only observations from one class, it will certainly be labeled as that class; if a leaf node is not pure and has observations from more than two classes, it will be labeled by the class that has most observations represented.

3. Implementation issues of CART

A significant benefit of the CART method is that it is straightforward to render the information in CART as logical expressions, which makes it easy to incorporate prior expert knowledge.

As with other data-driven classification techniques, performance of classification trees is highly dependent on the quality and quantity of training data. In the context of process monitoring, faults that the classification tree will be able to recognize are the faults represented by the original training data from which the tree is built.

3.1 Decision boundaries created with CART

Since the decision rule at each CART node only contains one property, which leads to a hyperplane decision boundary that are perpendicular to the coordinate axes, CART creates decision boundaries with portions perpendicular to the property axes (Fig. 3.2). With a sufficiently large tree, any decision boundary can be approximated arbitrarily well, provided that enough training data are present.

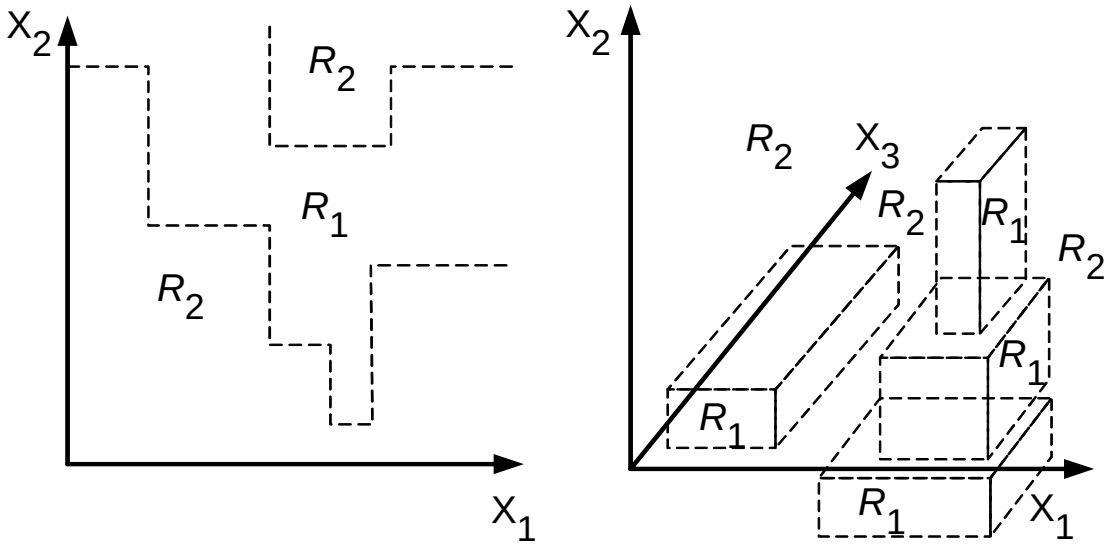


Fig. 3.2 Decision regions created by CART

Since the decision boundary generated by CART is perpendicular to the property axes, unnecessarily complicated classification trees may result in the cases where decision boundaries do not align with the property axes. An obvious example is shown in Fig. 3.3, in which the simple decision boundary has to be approximated with segments of lines. However, using an appropriate linear combination of the variables, it

is possible to result in a much simpler tree. In this study, discriminant analysis is employed to find this optimal combination.

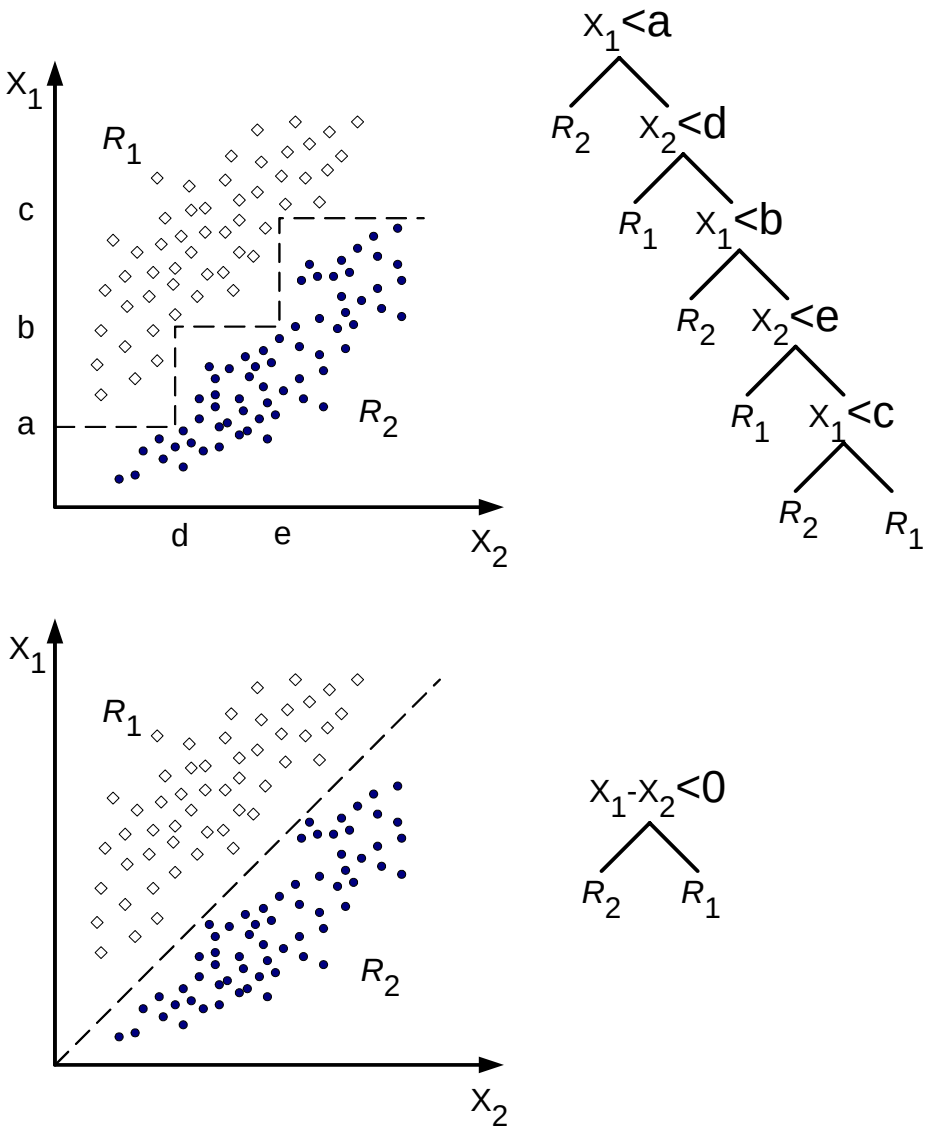


Fig. 3.3 A complicated classification tree could result from a simple decision boundary

3.2 Computational efficiency

Because of its tree structure, it is very fast to perform classification using CART. It takes only h steps of comparison to classify a pattern, where h is the average length from root node to leaf nodes and it can be approximated as $\log(n)$, where n is the number of training patterns.

Building a classification tree is computationally expensive. Suppose one has n training patterns and the dimension of the patterns is d. The computational complexity of building a fully-grown tree is illustrated as follows.

At the root node, one has to sort the training patterns on each of the d dimensions. This step takes $O(dn\log n)$. Calculating impurity index takes $O(dn)$, therefore, the computational cost of the root node is $O(dn\log n)$. Since there are two nodes on the next level, and each node takes $O\left(d\frac{n}{2}\log\frac{n}{2}\right)$, the computational cost of level 2 is $O\left(dn\log\frac{n}{2}\right)$. Similarly, the computational cost for level 3 is $O\left(dn\log\frac{n}{4}\right)$ and level 4 $O\left(dn\log\frac{n}{8}\right)$, and so on. Noticing that the depth of the tree is $\log n$, summing up the cost of each level the total cost is $O\left(\frac{1}{2}dn(\log n)^2\right)$.

It is beneficial to get a rough idea on the computational time of CART. In an example performed by the author, it took about 200 seconds to build a classification tree

from a data set of 2000 samples and 33 variables with a Pentium III 600 MHz PC. If the data set is considerably large, computing time could be significant.

In the process of pruning a classification tree, cross-validation is often employed to determine the optimal tree depth. This step can also be computationally expensive. For instance, one wants to examine the classification performance of D different depths of tree, and divides n training patterns into M groups. For each depth, M classification trees are built and its performance evaluated, which takes $O\left(M\frac{1}{2}d\frac{M-1}{M}n(\log\frac{M-1}{M}n)^2 \right)$, or $O\left(\frac{1}{2}d(M-1)n(\log\frac{M-1}{M}n)^2 \right)$. A total of D different depths of trees are examined, which results in a computational cost of $O\left(\frac{1}{2}dD(M-1)n(\log\frac{M-1}{M}n)^2 \right)$. If one assumes that $M >> 1$, the step of determining the optimal tree depth is $O\left(\frac{1}{2}DMdn(\log n)^2 \right)$, that is DM times the complexity of building a classification tree.

4. Process monitoring based on classification tree and discriminant analysis

Dimensionality reduction can be a key factor in reducing the misclassification rate when a pattern classification scheme is applied to new data [55]. The dimensionality reduction is especially important when the dimension of the observation space is large while the number of observations is relatively small. Another benefit provided by dimensionality reduction is that computational intensity is greatly reduced

for some applications like neural network or classification tree, when training or constructing time is proportional to the dimension of the process data.

Fisher Discriminant Analysis, a dimensionality reduction technique that has been extensively studied in the pattern classification literature, takes into account the information between the classes and provides an optimal lower dimensional representation in terms of discriminating among classes of data [56]. FDA determines a set of projection vectors, ordered in terms of maximizing the scatter between the classes while minimizing the scatter within each class.

Define n as the number of observations, m as the number of measurement variables, p as the number of classes, and n_j as the number of observations in the j^{th} class. x_i represents the vector of measurement variables for the i^{th} observation. If the training data for all classes have already been stacked into the matrix $X \in R_{n\times m}$, then the transpose of the i^{th} row of X is the column vector x_i. To perform FDA, one needs to calculate the total-scatter, the within-class scatter, and the between-class scatter. The total-scatter matrix is:

$$S_t = \sum_{i=1}^{n} (\mathbf{x_i} - \bar{\mathbf{x}})(\mathbf{x_i} - \bar{\mathbf{x}})' \tag{3.4}$$

where $\bar{\mathbf{x}}$ is the total mean vector

$$\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x_i} \tag{3.5}$$

Define χ_j as the set of vectors $\mathbf{x_i}$ that belong to class j, the within-class scatter matrix of class j is

$$S_j = \sum_{\mathbf{x_i} \in \chi_j} (\mathbf{x_i} - \bar{\mathbf{x}}_\mathbf{j})(\mathbf{x_i} - \bar{\mathbf{x}}_\mathbf{j})' \tag{3.6}$$

where $\bar{\mathbf{x}}_\mathbf{j}$ is the mean vector of class j

$$\bar{\mathbf{x}}_\mathbf{j} = \frac{1}{n_j} \sum_{\mathbf{x_i} \in \chi_j} \mathbf{x_i} \tag{3.7}$$

The within-class scatter matrix is

$$S_w = \sum_{j=1}^{p} S_j \tag{3.8}$$

and the between-class scatter matrix is

$$S_b = \sum_{j=1}^{p} n_j (\overline{\mathbf{x}_\mathbf{j}} - \bar{\mathbf{x}})(\overline{\mathbf{x}_\mathbf{j}} - \bar{\mathbf{x}})' \tag{3.9}$$

The total scatter matrix is equal to the sum of the between-class scatter matrix and the within-class scatter matrix

$$S_t = S_b + S_w \tag{3.10}$$

The objective of the first FDA vector is to maximize the scatter between classes while minimizing the scatter within classes.

$$\max \frac{\mathbf{v}' S_b \mathbf{v}}{\mathbf{v}' S_w \mathbf{v}}, \mathbf{v} \neq \mathbf{0} \tag{3.11}$$

The second FDA vector is computed so to maximize the scatter between classes while minimizing the scatter within classes among all axes perpendicular to the first FDA vector, and so on for the remaining FDA vectors. Solving the optimization problem and the FDA vectors are equal to the eigenvectors w_k of the generalized eigenvalue problem

$$S_b \mathbf{w_k} = \lambda_k S_w \mathbf{w_k} \tag{3.12}$$

where the eigenvalues λ_k indicate the degree of overall separability among the classes by projecting the data onto $\mathbf{w_k}$. Define matrix $W_p \in R^{m \times p}$ with the p FDA vectors as columns. Then the projection of the data from m-dimensional observation space to p-dimensional score space is described by

$$\mathbf{z_i} = W_p' \mathbf{x_i} \tag{3.13}$$

With the desirable property of FDA, a new process monitoring scheme that integrates FDA and classification tree is proposed. FDA extracts the most significant scores in the original process data and achieves optimal discrimination among different faults. The classification tree uses the FDA scores, the lower dimensional representation produced by FDA, to separate observations into different fault classes.

To determine the optimal order of the dimensionality reduction (i.e., the order of the FDA score space), the following stopping rule is proposed, as illustrated in Fig. 3.4: at each stage of the FDA procedure, construct the classification tree and calculate its tree cost. Stop the FDA procedure when the addition of the next FDA score to the

classification tree does not produce a tree that has a lower cost than the current one. The order of the FDA dimensionality reduction is thus determined. This procedure selects the order of FDA scores that gives a minimum tree cost.

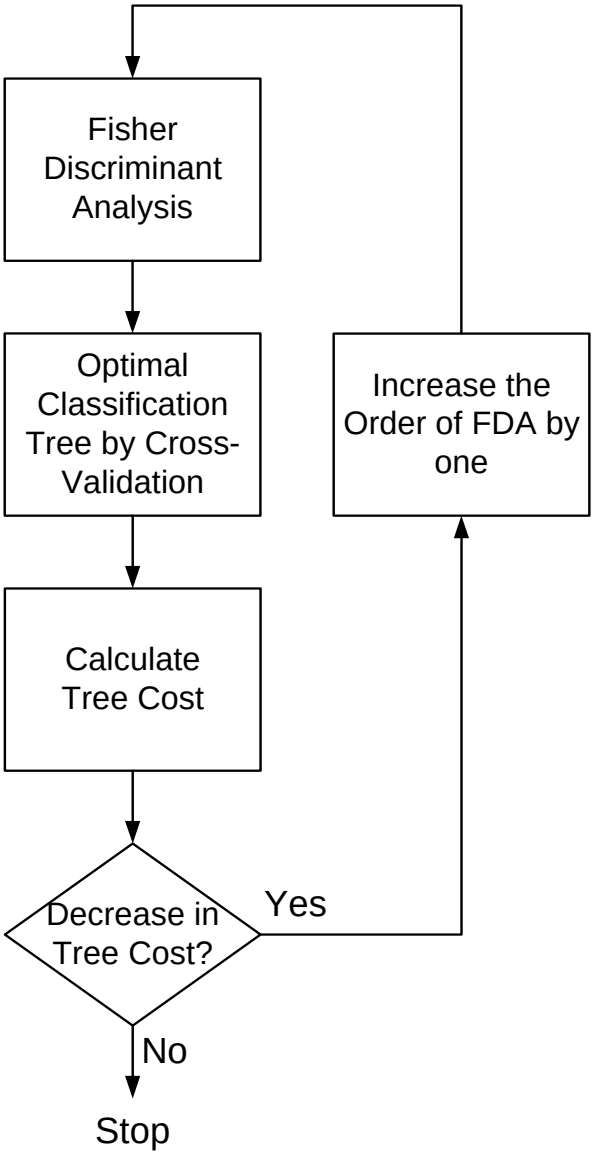


Fig. 3.4 Procedure to determine the optimal order of FDA

5. Case study 1: Tennessee Eastman Process

5.1 Process description and data preparation

The Tennessee Eastman Process (TEP) was developed by Eastman Chemical Company to provide a realistic industrial process for benchmarking process control and process monitoring methods. TEP is based on an actual chemical process but the components, kinetics, and operation conditions are modified for trade secret reasons. TEP has been widely used by process monitoring researchers as a source of data for comparing various methods [57, 58].

The process consists of a reactor, compressor, stripper, separator and condenser. Eight components are involved in the process. A flow sheet is shown in Fig. 3.5. The process produces two products G and H from four reactants A, C, D, and E. In the reaction system also present inert B and byproduct F.

The reactions involved in the process are:

$$A(g) + C(g) + D(g) \rightarrow G(l), \text{ product 1}$$

$$A(g) + C(g) + E(g) \rightarrow H(l), \text{ product 2}$$

$$A(g) + E(g) \rightarrow F(l), \text{ byproduct}$$

$$3D(g) \rightarrow 2F(l), \text{ byproduct} \qquad (3.14)$$

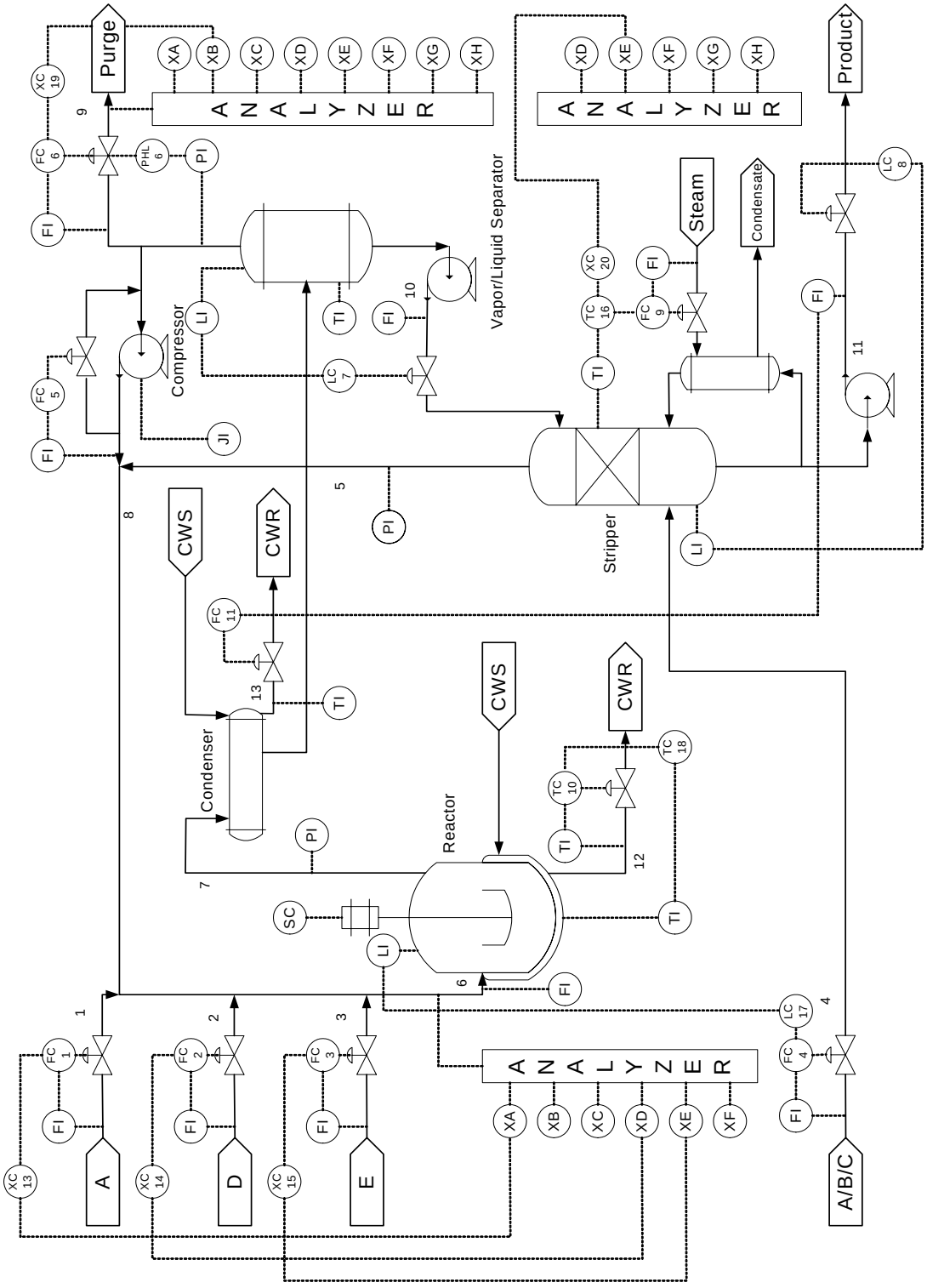


Fig. 3.5 Schematic of the Tennessee Eastman Process [59]

All the reactions are irreversible, exothermic, and approximately first-order with respect to the reactant concentrations. The reaction rates follow an Arrhenius expression and the reaction to produce G has high activation energy, resulting in a high sensitivity to temperature.

The reactant gases are fed into the reactor where they form liquid products, catalyzed by a nonvolatile catalyst dissolved in the liquid. Cooling water runs inside the reactor to remove the heat of reaction. The gaseous products leave the reactor and the catalyst remains in. The product gas stream is cooled through a condenser and then fed to a vapor-liquid separator. Non-condensed vapor from the separator recycles back to the reactor through a compressor. The inert and byproduct are purged from the process in the vapor-liquid separator. Condensed stream from the separator moves to a stripper with feed stream to remove the remaining reactants. The products G and H exiting at the base of the stripper are pumped to a downstream unit that is not included in the illustration.

The process contains 41 measured and 12 manipulated variables and all the process measurements include Gaussian noise. Of the 41 process variables, XMEAS(1) to XMEAS(22) are continuous measurements and the rest are composition measurement which are not available in real time (for example, XMEAS(23) to XMEAS(28) are sampled at 0.1hr interval and 0.1hr of dead time). XMEAS(1) through XMEAS(22) are listed in Table 3.1 and XMEAS(23) to XMEAS(41) are presented in Table 3.2. The 12 manipulated variables are described in Table 3.3.

Table 3.1 Process measurements XMEAS(1) through XMEAS(22) (sampling interval 3 mins)

Process Variable	**Description**	**Units**
XMEAS(1)	A Feed (stream 1)	kscmh
XMEAS(2)	D Feed (stream 2)	kg/hr
XMEAS(3)	E Feed (stream 3)	kg/hr
XMEAS(4)	Total Feed (stream 4)	kscmh
XMEAS(5)	Recycle Flow (stream 8)	kscmh
XMEAS(6)	Reactor Feed Rate (stream 6)	kscmh
XMEAS(7)	Reactor Pressure	kPa Gauge
XMEAS(8)	Reactor Level	%
XMEAS(9)	Reactor Temperature	ºC
XMEAS(10)	Purge Rate (stream 9)	kscmh
XMEAS(11)	Product Separator Temperature	ºC
XMEAS(12)	Product Separator Level	%
XMEAS(13)	Product Separator Pressure	kPa Gauge
XMEAS(14)	Product Separator Underflow (stream 10)	m^3/hr
XMEAS(15)	Stripper Level	%
XMEAS(16)	Stripper Pressure	kPa Gauge
XMEAS(17)	Stripper Underflow (stream 11)	m^3/hr
XMEAS(18)	Stripper Temperature	ºC
XMEAS(19)	Stripper Steam Flow	kg/hr
XMEAS(20)	Compressor Work	kW
XMEAS(21)	Reactor Cooling Water Outlet Temperature	ºC
XMEAS(22)	Separator Cooling Water Outlet Temperature	ºC

Table 3.2 Composition measurements

Process Variable	Component	Stream	Sampling Interval/Dead Time	Units
XMEAS(23)	A	6	6min/6min	mol%
XMEAS(24)	B			
XMEAS(25)	C			
XMEAS(26)	D			
XMEAS(27)	E			
XMEAS(28)	F			
XMEAS(29)	A	9	6min/6min	
XMEAS(30)	B			
XMEAS(31)	C			
XMEAS(32)	D			
XMEAS(33)	E			
XMEAS(34)	F			
XMEAS(35)	G			
XMEAS(36)	H			
XMEAS(37)	D	11	15min/15min	
XMEAS(38)	E			
XMEAS(39)	F			
XMEAS(40)	G			
XMEAS(41)	H			

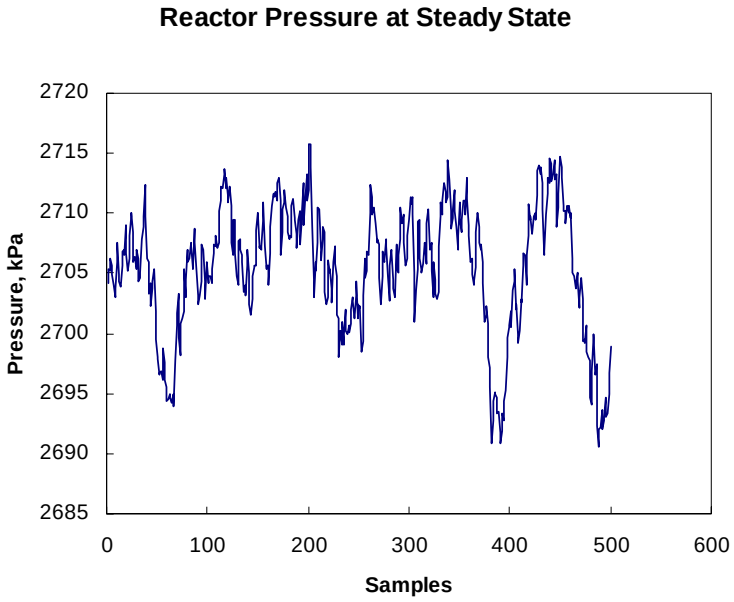
Table 3.3 Manipulated variables

Process Variable	**Description**	**Units**
XMV(1)	D Feed Flow Rate (stream 2)	kg/hr
XMV(2)	E Feed Flow Rate (stream 3)	kg/hr
XMV(3)	A Feed Flow Rate (stream 1)	kscmh
XMV(4)	A and C Feed Flow Rate (stream 4)	kscmh
XMV(5)	Compressor Recycle Valve	%
XMV(6)	Purge Valve (stream 9)	%
XMV(7)	Separator Pot Liquid Flow Rate (stream 10)	m^3/hr
XMV(8)	Stripper Liquid Product Flow Rate (stream 11)	m^3/hr
XMV(9)	Stripper Steam Valve	%
XMV(10)	Reactor Cooling Water Flow Rate	m^3/hr
XMV(11)	Condenser Cooling Water Flow Rate	m^3/hr
XMV(12)	Agitator Speed	rpm

The TEP simulation also contains 21 preprogrammed faults, 16 of them are known, and 5 are unknown. These faults are associated with step changes in the process variables, an increase in the variability of process variables, and actuator faults such as sticking valves. The simulation code and description of TEP is available in FORTRAN from several sources [59] [60].

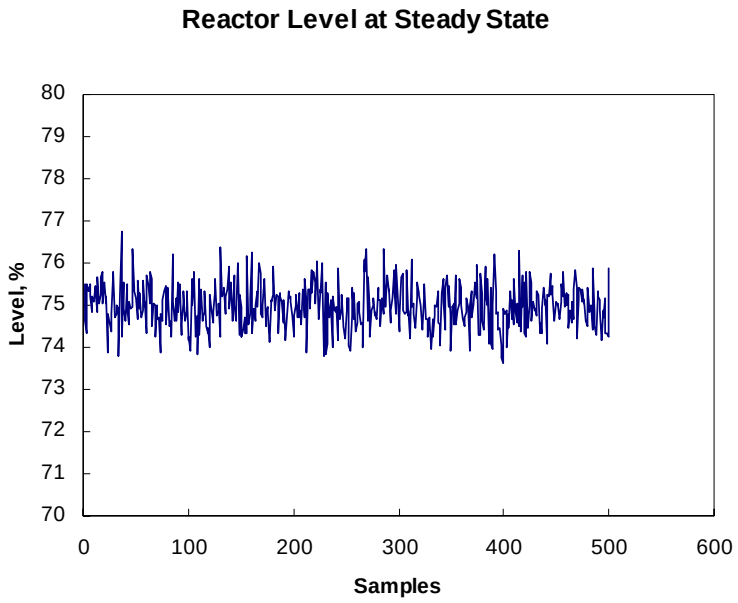
Comparing different monitoring results is difficult since each approach employs a different control scheme that affects the system behavior and the consequent correlation between the process variables. To illustrate the presented approach and

make a meaningful comparison with other techniques, the simulation data for the faults and the normal conditions generated with the plant-wide control scheme described in the literature [61] are used in this study. Some process variables at the steady state and faulty state are shown in Figs. 3.6 and 3.7.

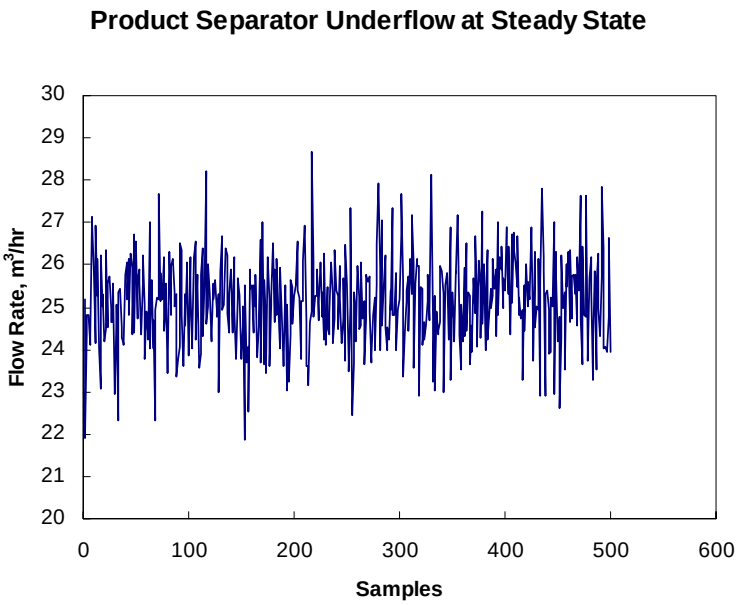


(a)

Fig. 3.6 Closed-loop simulation of TEP without faults, the upper line represents %G and the lower line %H in (e)

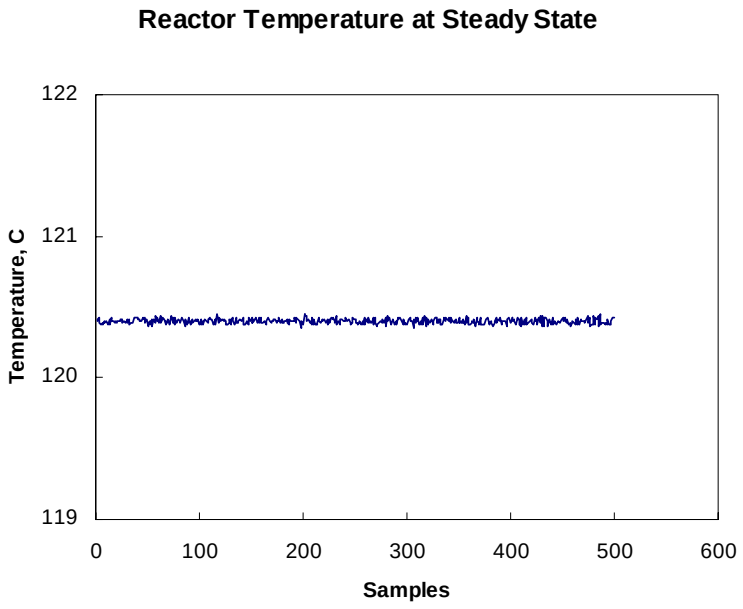


(b)

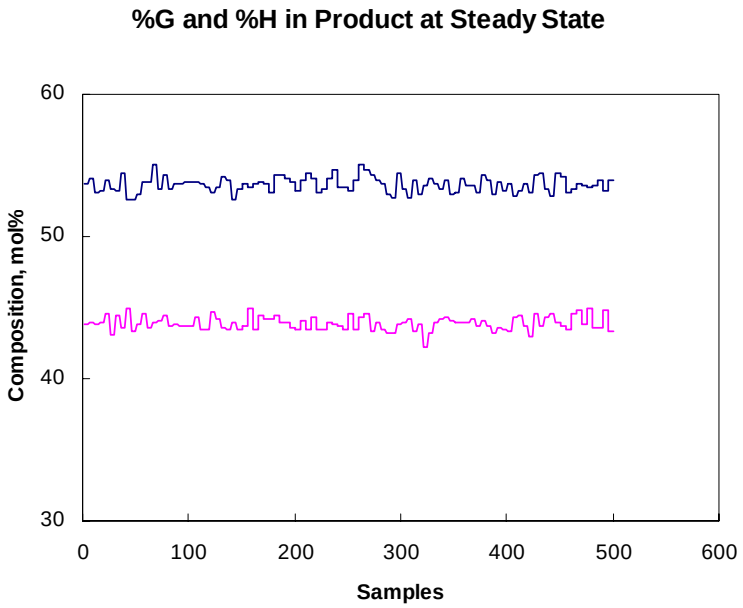


(c)

Fig. 3.6 Continued

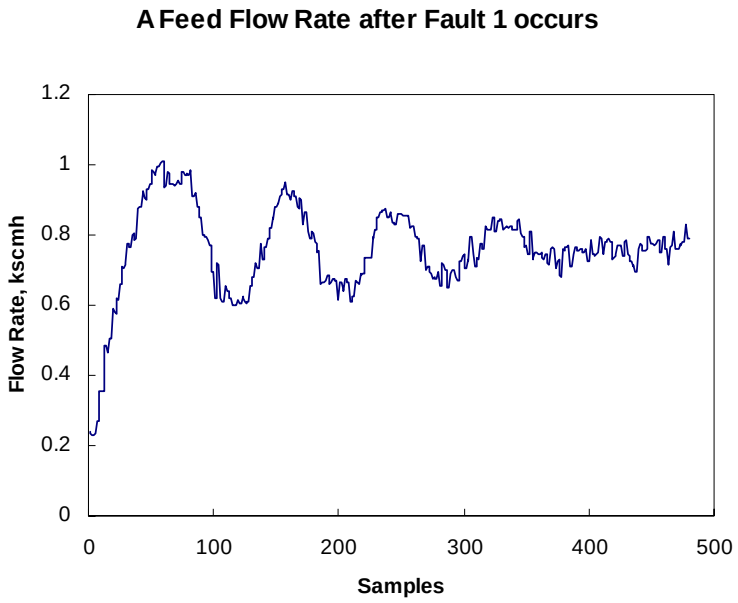


(d)

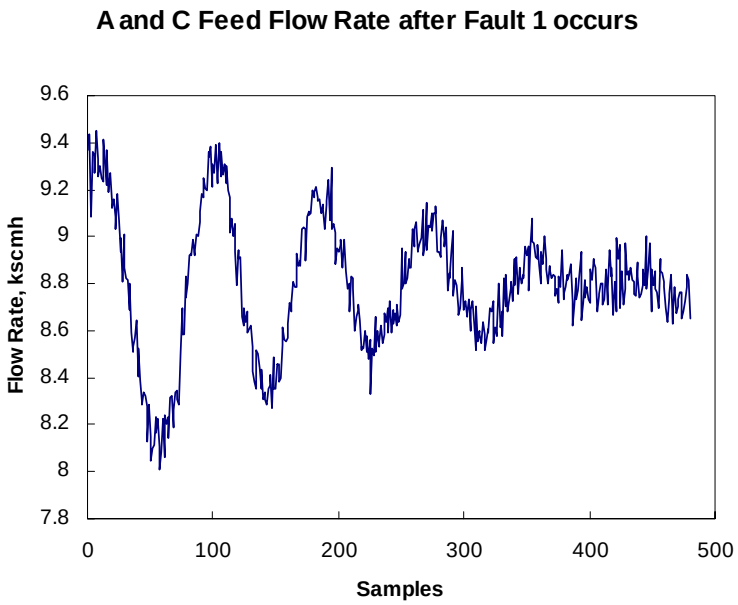


(e)

Fig. 3.6 Continued

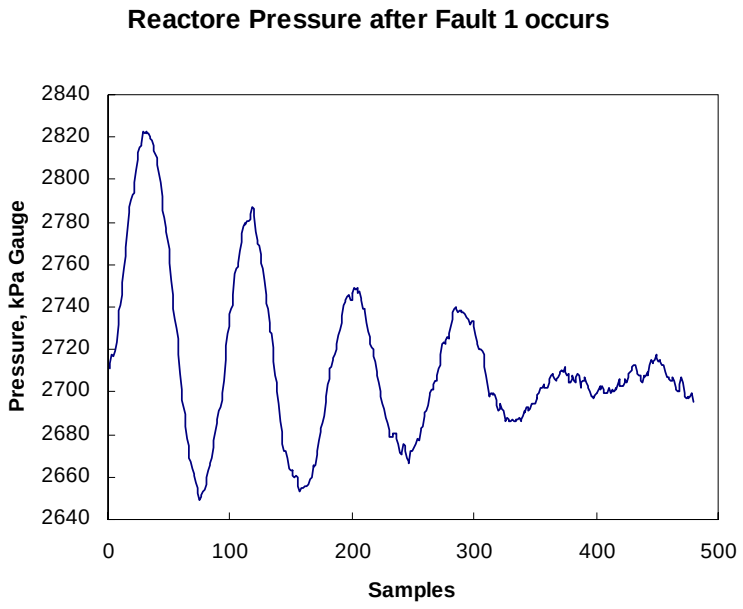


(a)

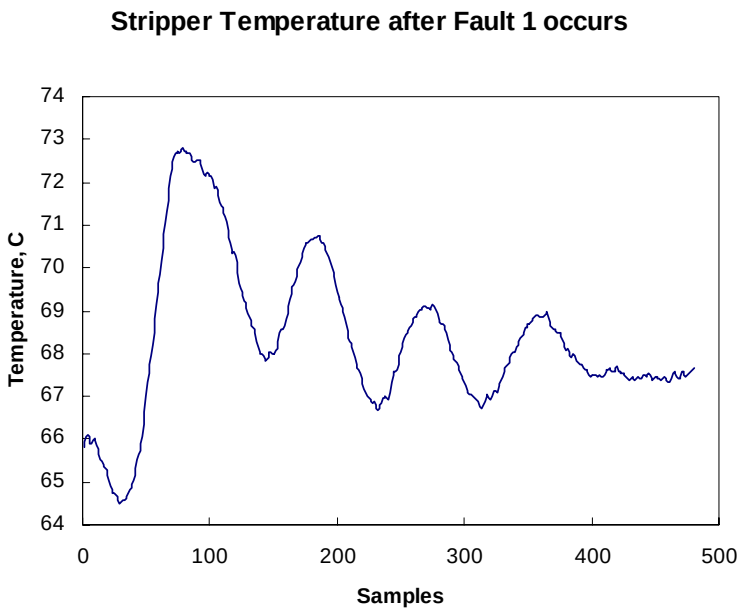


(b)

Fig. 3.7 Closed-loop simulation of some process variables after Fault 1 occurs



(c)



(d)

Fig. 3.7 Continued

Among the 21 preprogrammed faults, eight are selected as the testing faults. Seven of them are caused by step changes in process variables, one by increased variability in process variables. Since 19 measured variables are compositions that are not available in real time, they are not used in the process monitoring schemes. A sampling interval of 3 minutes was selected here to allow fast detection. 1055 samples are collected in a simulation as the training data and 1615 as the testing data in another simulation. Classification tree is implemented with the Statistics Toolbox of MATLAB 6.5 for Windows. All the computing time is based on a Pentium III 600MHz PC.

5.2 Fault detection and diagnosis with classification tree

A maximum classification tree (e.g., a tree with every leaf node containing only one class) is created with the training data set with their respective fault classes (see Fig. 3.8). This tree can perform fault detection and diagnosis by classifying process measurements into different classes. With this maximum tree, there is a strong possibility that it fits the training data set well but would not perform well at classifying new data. Some of its lower branches may be strongly affected by process noise of the training data. A simpler tree is preferred to avoid the problem of over-fitting.

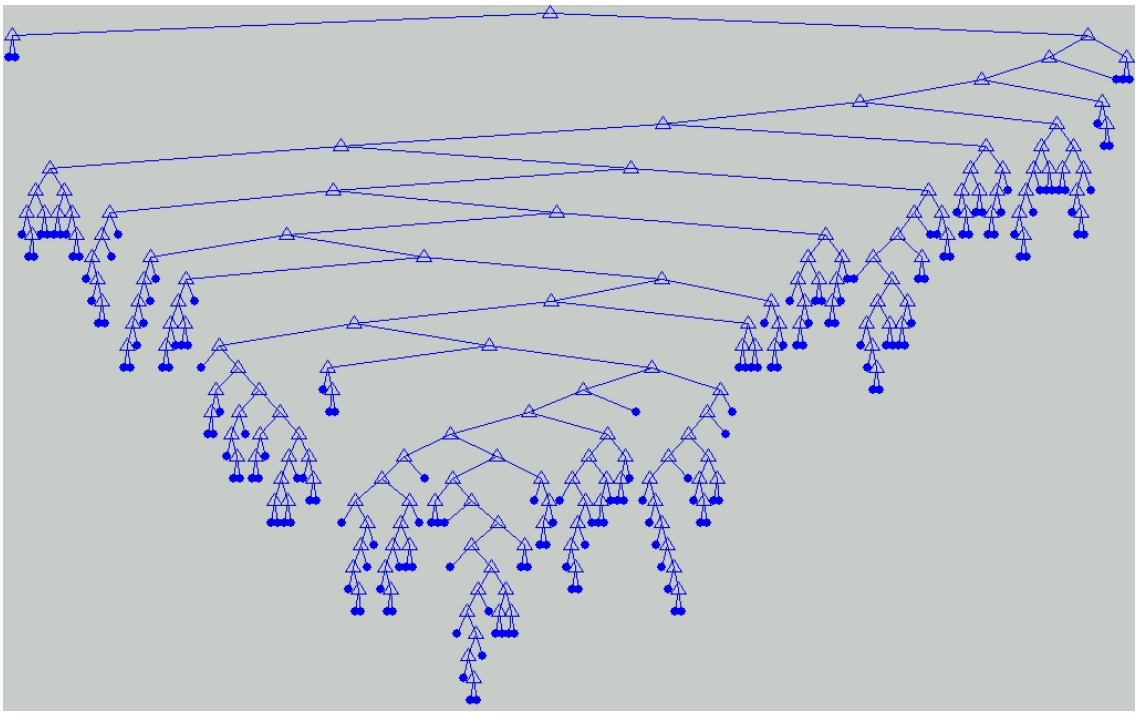


Fig. 3.8 Maximum classification tree, node labels removed for clarity

The optimal tree size is estimated with cross validation. The original training set is partitioned into 10 subsets, chosen randomly but with equal size and roughly the same class proportions. For each subset, a tree is fitted using the remaining data and use it to predict the subset. The information from all subsets is pooled to compute the cost of a tree of certain size. The cross validation error of various tree sizes is calculated and plotted in Fig. 3.9. The best tree size is selected as the one that has a cross validation error that is no more than one standard error above the minimum value along the cross validation line [53]. The optimal tree is thus obtained by pruning the maximum tree according to the best tree size. Fig. 3.10 shows the pruned tree.

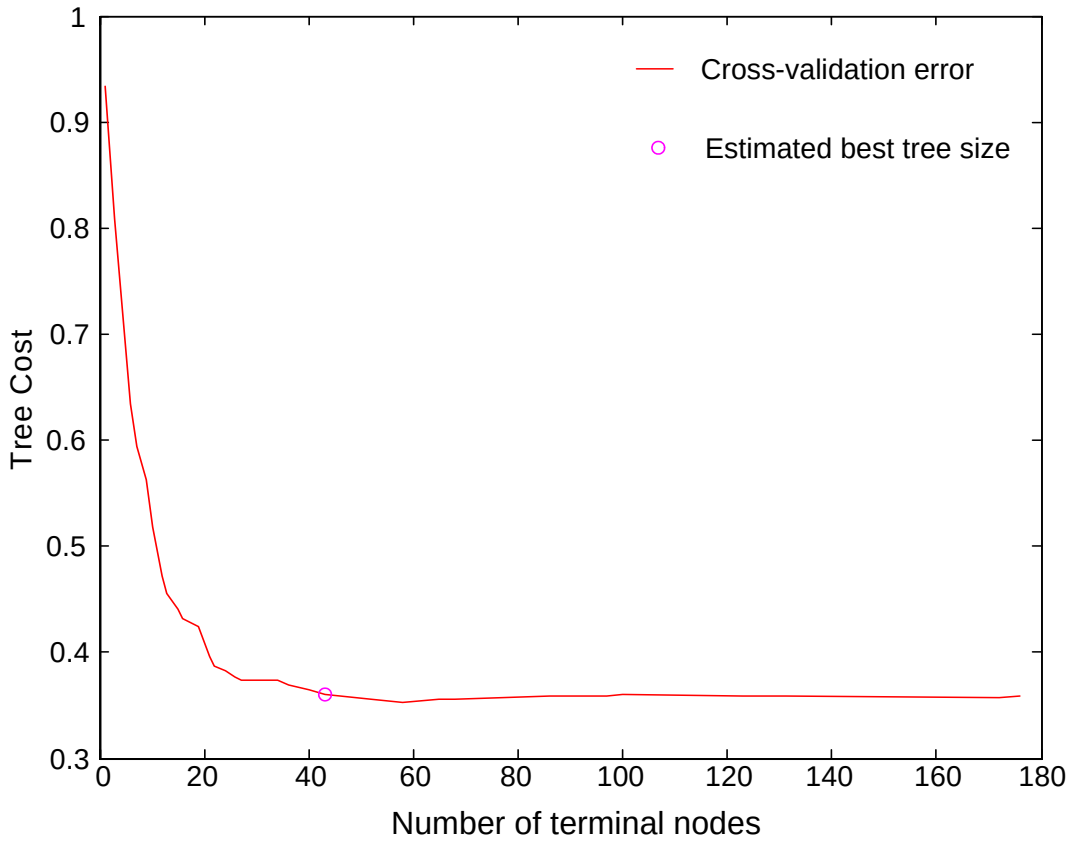


Fig. 3.9 Choosing the optimal tree size by cross validation

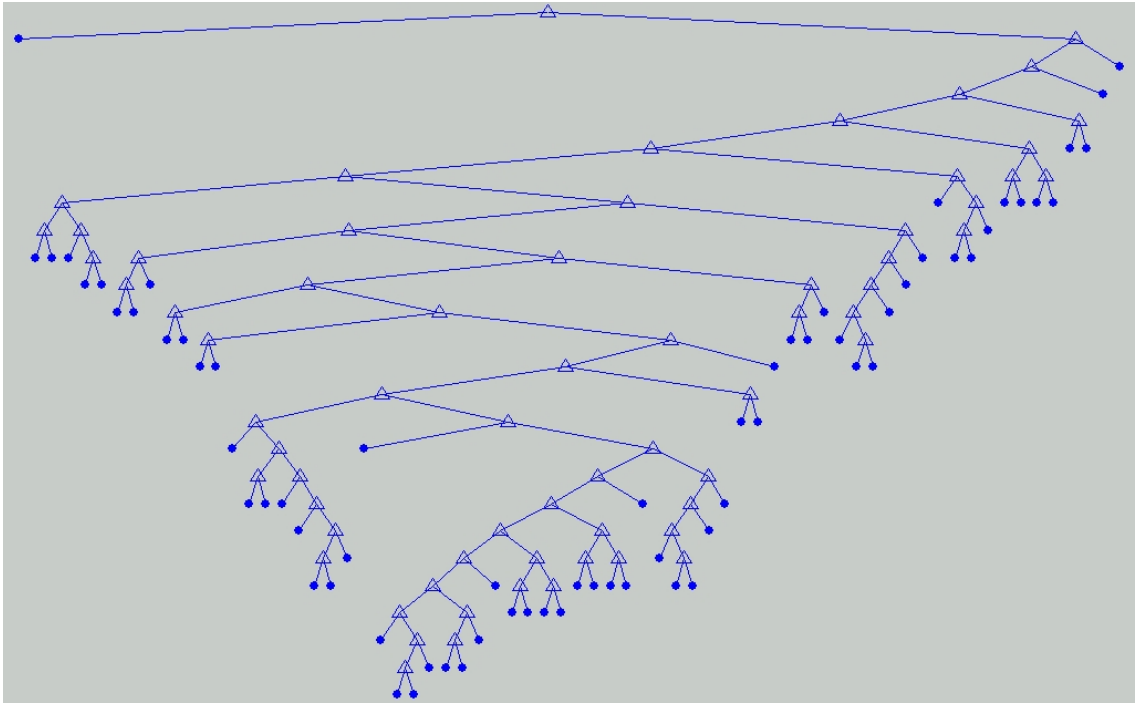


Fig. 3.10 Pruned tree, node labels removed for clarity

5.3 Fault detection and diagnosis with classification tree and discriminant analysis

Fisher Discriminant Analysis is also implemented with MATLAB 6.5 based on the algorithm described in Section III.4. The original training data are transformed into FDA scores, which are subsequently used to build the classification tree. New process measurements need to be transformed in the same way to get FDA scores for the classification tree to perform fault detection and diagnosis.

To find the optimal FDA order, starting from 1, a series of FDA of consecutively increasing orders are performed and classification trees are generated and

tested. Cross validation is employed to produce the optimal classification tree and the cost of the trees is plotted along the FDA order in Fig. 3.11. It is found that from order 9 to order 10 the cost of the optimal tree does not decrease; therefore 9 is chosen as the optimal FDA order. A much more simplified classification tree (Fig. 3.12) is generated due to the reduced dimensionality.

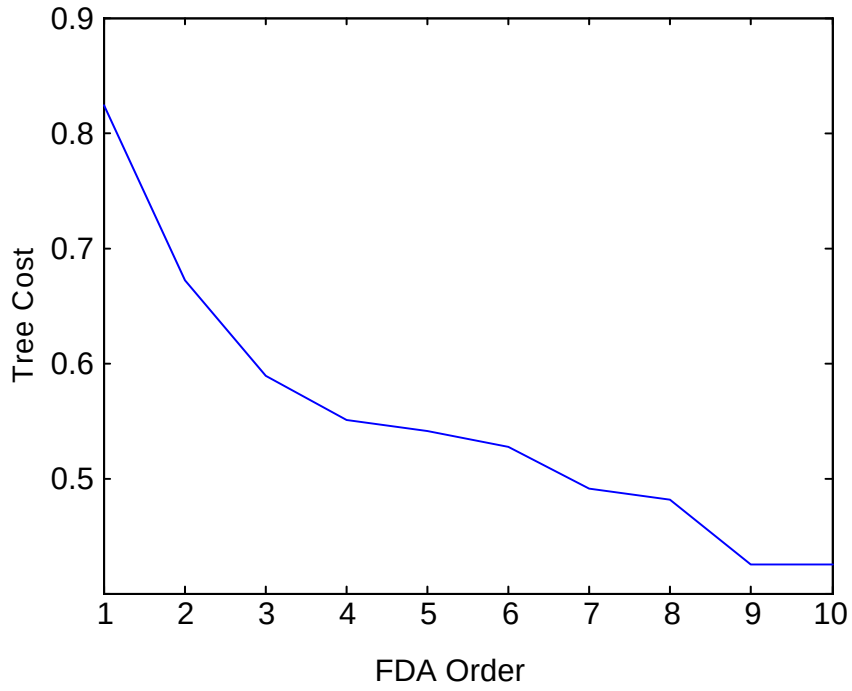


Fig. 3.11 Optimal FDA order

To compare the performance of various approaches, the misclassification rates on the testing data of classification tree, classification tree with discriminant analysis, and various statistical methods (results taken from literature [55]) are compared in Table 3.4.

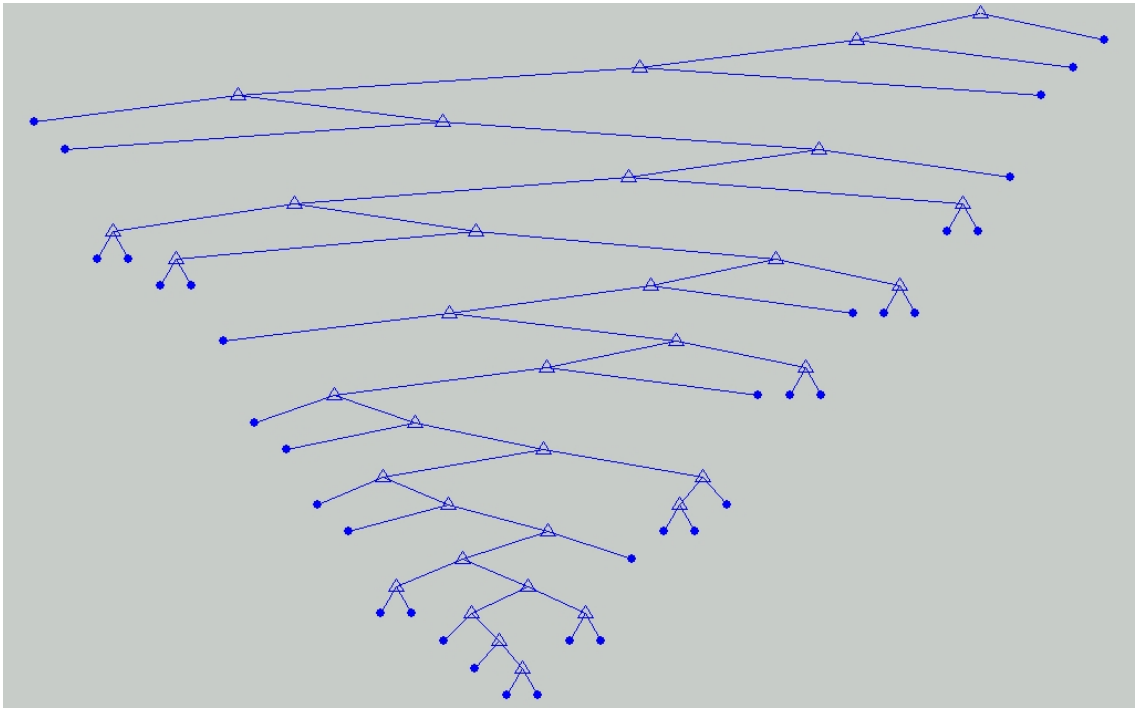


Fig. 3.12 FDA scores result in a much smaller classification tree, node labels removed for clarity

Table 3.4 Comparison of misclassification rates and construction time for various methods, best results shown in bold

Fault Number and Description	Fault Type	Classification Tree	Classification Tree w/ Discriminant Analysis	Principal Component Analysis – T^2	Partial Least Squares	Canonical Variate Analysis - Q
1A/C Feed Ratio	Step	0.026	0.025	0.024	**0.013**	0.245
2 B Composition	Step	**0.000**	**0.000**	0.018	0.014	0.155
3 D Feed Temp.	Step	**0.724**	0.863	0.783	0.961	0.978
4 Reactor Cooling Water Temp.	Step	**0.046**	0.121	0.163	0.170	0.890
5 Condenser Cooling Water Temp.	Step	0.454	0.011	0.021	**0.006**	0.174
6 A Feed Loss	Step	0.015	**0.000**	**0**	0.435	0.014
7 C Header Pressure Loss	Step	**0.000**	**0.000**	**0**	**0**	0.578
8 A B C Feed Composition	Random Variation	0.569	0.419	**0.03**	0.851	0.670
Constructing Time, sec		103	**36**			

When Fault 1 or Fault 2 occurs, the distribution of the variables associated with material balances changes correspondingly. Since more than half of the process variables deviate significantly from their normal operating states, these two faults are easily classified by all methods. The misclassification rates of Fault 3 are extremely high for all methods, due to the fact that the control system effectively compensates the disturbance such that this fault is unobservable from the process data. A significant effect of Fault 4 is to induce a step change in the reactor cooling water flow rate, while the other process variables remain steady after the fault occurs. This makes it harder for all methods to classify. Fault 5 is similar to Fault 4, except that it affects more variables. Like Fault 1 and 2, Fault 6 and 7 are easy for most methods because many process variables deviate from their normal states after they occur. Unlike Fault 1-7, Fault 8 is caused by increased variability in feed compositions. One can see that both classification tree methods and the other two statistical techniques performed poorly.

It can be seen from the comparison that classification tree with discriminant analysis results in the lowest average misclassification rate for Fault 1~7, as well as greatly reduced training time compared to classification tree. However, classification tree-based methods are not effective when applied to random variation type of fault. The possible explanation is that classification tree examines process variables statically and a big portion of the abnormal states falls closely to normal states.

6. Case study 2: a pilot distillation column

6.1 Process description and data acquisition

The pilot size distillation column studied is the same one as described in Section II.4.1. Two types of faults are artificially induced in this study: a “sticky control valve” fault (fault 1) is induced by setting the corresponding column temperature control loop to manual; a “vapor bypass” fault (fault 2) is induced by opening the second (lower) feed line to the column. These simulated faults are very realistic in distillation processes.

Nine process variables are chosen to represent the state of the process: column temperatures at theoretical plates 5, 6, 7, and 8 (TC/5 ~ TC/8); feed low rate; feed temperature; reflux flow rate; reflux temperature; and steam condensate flow. 300 points of normal process data (samples 1~300), 133 points of “sticky control valve” faulty process data (samples 301~433), and 114 points of “vapor bypass” faulty process data (samples 434~547) are recorded in a run, as shown in Fig. 3.13. Another 20 points of “sticky control valve” faulty process data and 28 points of “vapor bypass” faulty process data are recorded in a separate run, which are to be used as testing data.

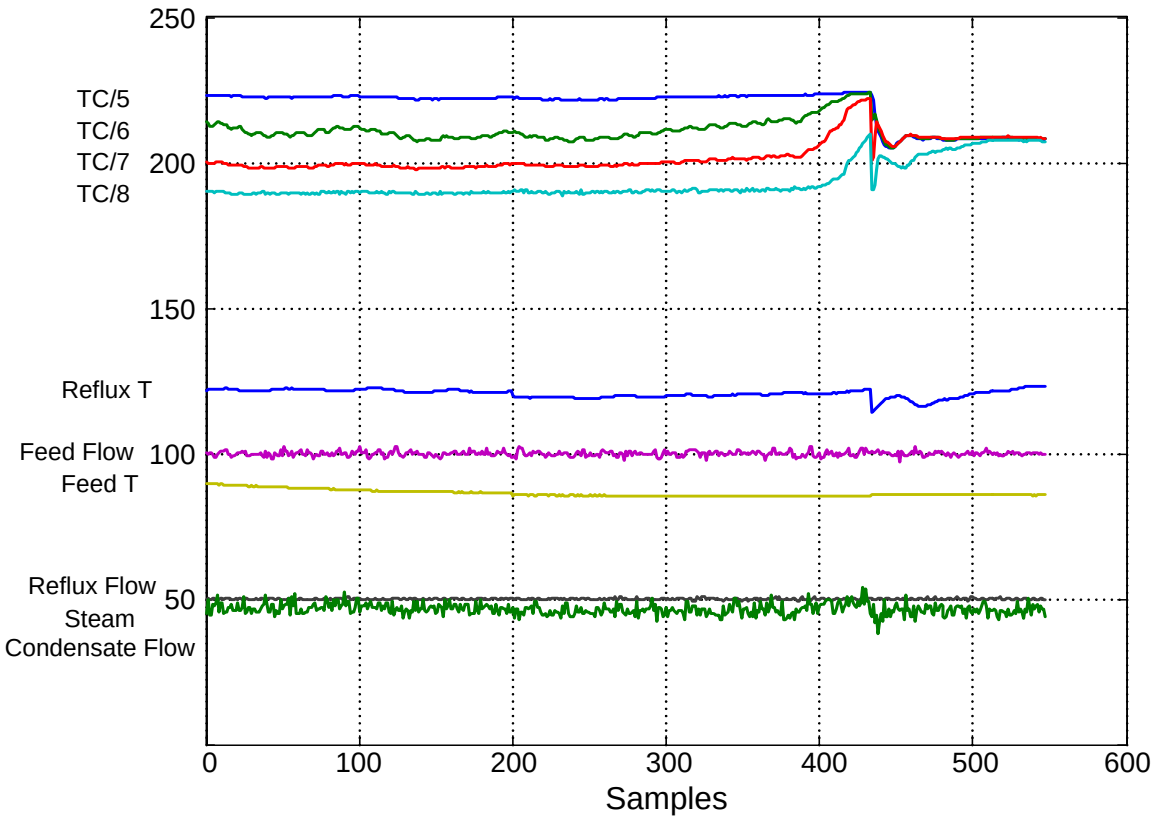


Fig. 3.13 Distillation column measurements used as the training data

6.2 Discussion

Both methods of classification trees with and without discriminant analysis are trained with the training data and tested against the test data. The misclassification rates of the methods are compared in Table 3.5. As one can see from the results, first order FDA is not capable of capturing enough variations in the original data and does not yield a satisfactory classification. Due to the strong correlations between temperature measurements, classification trees with only 2nd order FDA outperformed classification tree method by classification rates. The reduction in training time is not significant because of the relatively small size of the training data set.

Table 3.5 Comparison of misclassification rates of classification trees with and without discriminant analysis

Fault Number and Description	Classification Tree Only	Classification Tree w/ FDA Order 1	Classification Tree w/ FDA Order 2	Classification Tree w/ FDA Order 3
0 Normal	0	0	0	0
1 Sticking Control Valve	0.04	0.64	0.04	0.04
2 Vapor Bypassing	0.0909	0	0.0303	0.0303

7. Conclusions

Classification tree analysis has the advantage of being concise, fast to compute, and making no assumption regarding the distribution of the predicting variables. However, cross validation used in building a classification tree is a computationally intensive process, and it is even more difficult when applied to large-scale systems with high dimensional observations. Another disadvantage arises from the fact that the decision boundary generated by classification tree is perpendicular to the property axes, thus unnecessarily complicated classification tree may result in the cases where decision boundaries do not align with the property axes.

The newly developed process monitoring scheme integrates classification tree and Fisher Discriminant Analysis (FDA). FDA extracts the most significant components in the original process data and achieves optimal discrimination among different faults, as well as reduces the dimension of the original data. Classification tree uses the FDA scores, which are the lower dimensional representation produced by FDA, to separate observations for different fault class. A stopping rule is applied to determine

the optimal order of FDA. This procedure selects the order of FDA scores that gives a minimum tree cost.

Two case studies are presented to illustrate the effectiveness of the proposed methods compared with the original classification tree method. In the first case study, Tennessee Eastman Process simulations with 34 process variables and 15 simulated faults are examined. The results show a greatly reduced cross validation time in classification tree construction, as well as better classification performance. Process data collected from a pilot distillation column are used in the second case study. The new method again generates better classification accuracy and uses less construction time, although insignificant due to the relatively small size of the data set. This is a general approach that can be applied for process monitoring using classification trees.

IV. CONCLUSIONS AND RECOMMENDATIONS

1. Conclusions

Current approaches to process monitoring can be generally grouped into three categories: (1) Analytical model based methods, (2) Knowledge based methods, and (3) Historical data driven methods. Methods in (1) are theoretically elegant, but the difficulty in obtaining accurate analytical process model and model non-linearity seriously limit their applicability to process industries. Fault tree and digraph based methods belong to category (2), which are easy in principle but time consuming to develop. Historical data driven methods, including those that are based on multivariate statistical analysis (PCA, PLS, CVA, etc.) and most neural network based methods, consist of two steps, feature extraction and pattern recognition, and have been the most widely applied ones in process industries.

Neural networks have been popular in process modeling and fault pattern recognition. However, in batch processes, operating regions are seldom constant and this may require considerable amount of re-training of the neural process model, whose computational expense is often prohibitive for today's complex process systems. This reduces the potential of neural network-based fault diagnosis systems applied to batch processes. In this work, a polynomial regression based feature extraction/preprocessing scheme drastically reduced the dimensionality of input to neural model, thus reducing the neural network size, as well as the required training time. The method comes with a

slight sacrifice in model performance due to increased non-linearity introduced by the feature extraction step.

Two case studies demonstrate the effectiveness of the new approach. In both cases, the new approach greatly reduces neural model training time and in the second case, the classification performance is also improved.

Classification trees suffer from the computationally expensive cross validation during the building process, which can create real difficulties when applied to large-scale systems with high dimensional observations. Another disadvantage is that the decision boundary generated by a classification tree is perpendicular to the property axes, thus unnecessarily complicated classification trees may result in the cases where decision boundaries do not align with the property axes.

The newly developed process monitoring scheme addresses the issues by employing Fisher Discriminant Analysis (FDA). FDA extracts the most significant components in the original process data and achieves optimal discrimination among different faults, as well as reduces the dimension of the original data and projects old data into new “scores” that are perpendicular to each other. Classification trees use the FDA scores to separate observations into different fault classes. A procedure identifies the order of FDA scores that gives a minimum tree cost as the optimal order.

Application of the new scheme on Tennessee Eastman Process simulations shows a greatly reduced cross validation time in classification tree construction, as well as better classification performance. The new approach is also compared with other

published statistical methods and demonstrated superior performance. The second case study on a distillation column setup shows results along similar lines.

2. Recommendations

Besides the issues addressed in this dissertation, historical data driven process monitoring methods suffer from their dependency on *a priori* fault information/process knowledge, i.e., if a new fault occurs, the system would not react.

The author suggests the following research topics that may help to tackle the problem:

- **Hybrid methods**: combining historical data driven methods with knowledge-based methods, such as a hybrid system of statistical analysis and fault tree: statistical analysis performs the fault detection and pattern clustering/classification, and fault tree is used for reasoning and diagnosis. The fault tree can be a dynamic one, which can be updated with new faults. This could be a viable research project since it can take advantage of knowledge gained in the fault tree research.
- **Process knowledge management**: there should be a unified data structure to facilitate the management of process knowledge that is generated from HAZOP and process monitoring.
- **Data mining**: modern chemical plants are producing an exploding amount of data, collected by thousands of sensors and stored with the help of data

librarian software. How to effectively extract useful information out of those data, e.g., distinguishing faults from fluctuations due to legitimate causes, and in the mean time get information to improve statistical process control (SPC) and process optimization.

REFERENCES

[1] R. Isermann and P. Ball, Trends in the application of model based fault detection and diagnosis of technical processes. in Proc. of the 13th IFAC World Congress, volume N, 1-12, IEEE Press, Piscataway, NJ, 1996

[2] A. C. Raich and A. Cinar, Statistical process monitoring and disturbance diagnosis in multivariable continuous processes. AIChE J., 42, (1996) 995-1009

[3] W. L. Brogan, Modern Control Theory, Prentice-Hall, Englewood Cliffs, NJ, 1991

[4] T. Kailath, Linear Systems, Prentice-Hall, Englewood Cliffs, NJ, 1980

[5] P. M. Frank and J. Wünnenberg, Robust fault diagnosis using unknown input observer schemes, in R. J. Patton, P. M. Frank, R. N. Clard (eds), Fault Diagnosis in Dynamic Systems: Theory and Application, Prentice-Hall, New York, 47-98, 1989

[6] N. Kazantzis and C. Kravaris, Discrete-time nonlinear observer design using functional equations, Systems & Control Letters, 42, (2001) 81-94

[7] Q. Zhang and B. Delyon, A new approach to adaptive observer design for MIMO systems, Proceedings of the 2001 American Control Conference, volume 2, 1545-1550, 2001

[8] R. K. Mehra and J. Peschon, An innovative approach to fault detection and diagnosis in dynamic systems, Automatica, 7, 637-640, 1971

[9] X. Ding and L. Guo, Observer-based fault detection optimized in the frequency domain, in Proc. of the 13th IFAC World Congress, volume N, 157-162, IEEE Press, Piscataway, NJ, 1996

[10] P. M. Frank, Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy – a survey and some new results, Automatica, 26, 459-474, 1990

[11] E. Gomez, H. Unbehauen, P. Kortmann, and S. Peters, Fault detection and diagnosis with the help of fuzzy-logic and with application to a laboratory turbogenerator, in Proc. of the 13th IFAC World Congress, volume N, 175-180, IEEE Press, Piscataway, NJ, 1996

[12] P. M. Frank and N. Kiupel, Residual evaluation for fault diagnosis using adaptive fuzzy thresholds and fuzzy inference. in Proc. of the 13th IFAC World Congress, volume N, 115-120, IEEE Press, Piscataway, NJ, 1996

[13] K. Danai and V. B. Jammu, Robust residual generation for fault diagnosis thru pattern classification. in Proc. of the 13th IFAC World Congress, volume N, 193-198, IEEE Press, Piscataway, NJ, 1996

[14] R. Isermann and B. Freyermuth, Process fault diagnosis based on process model knowledge – part I: principles for fault diagnosis with parameter estimation, ASME J. of Dynamics, Measurement, and Control, 113, (1991) 620-626

[15] C. N. Nett, C. A. Jacobson, and A. T. Miller, An integrated approach to controls and diagnostics: the 4-parameter controller, in Proc. of the American Control Conf., 824-835, IEEE Press, Piscataway, NJ, 1988

[16] M. L. Tyler and M. Morari, Optimal and robust design of integrated control and diagnostic modules, in Proc. of the American Control Conf., 2060-2064, IEEE Press, Piscataway, NJ, 1994

[17] P. Kesavan and J. H. Lee, Diagnostic tools for multivariable model-based control systems, Ind. Eng. Chem. Res., 36, (1997) 2725-2738

[18] W. Ku, R. H. Storer, and C. Georgakis, Uses of state estimation for statistical process control, Comp. & Chem. Eng., 18, (1994) S571-575

[19] M. Iri, K. Aoki, E. O'shima, and H. Matsuyama, An algorithm for diagnosis of system failures in the chemical process, Comp. & Chem. Eng., 3, (1979) 489-493

[20] J. Shiozaki, H. Matsuyama, E. O'shima, and M. Iri, An improved algorithm for diagnosis of system failures in the chemical process, Comp. & Chem. Eng., 9, (1985) 285-293

[21] M. A. Kramer and F. E. Finch, Development and classification of expert systems for chemical process fault diagnosis, Robotics and Computer-integrated Manufacturing, 4, (1988) 4346-4376

[22] E. S. Yoon and J. H. Han, Process failure detection and diagnosis using the tree model, in Proc. of the IFAC World Congress, 126-129, Pergamon Press, Oxford, UK, 1987

[23] Q. Zhang, A frequency and knowledge tree/causality diagram based expert system approach for fault diagnosis, Reliability Engineering & System Safety, 43, (1994) 17-28

[24] Y. S. Oh, J. H. Yoon, D. Nam, C. Han, and E. S. Yoon, Intelligent fault diagnosis based on weighted symptom tree model and fault propagation trends, in Joint 6th International Symposium on Process System Engineering and 30th European Symposium on Computer Aided Process Engineering, S941-946, Elsevier Science Ltd, Oxford, UK, 1997

[25] J. Zhang, E. B. Martin, and A. J. Morris, Fault detection and classification through multivariate statistical techniques, in Proc. of the American Control Conference, 751-755, IEEE Press, Piscataway, NJ, 1995

[26] R. Rengaswamy and V. Venkatasubramanian, A fast training neural network and its updation for incipient fault detection and diagnosis, Comp. & Chem. Eng., 24, (2000) 431-437

[27] R. Dunia, S. J. Qin, Joint diagnosis of process and sensor faults using principal component analysis, Control Engineering Practice, 6 (4) (1998) 457-469

[28] R. Dunia, S. J. Qin, T. F. Edgar, T. J. McAvoy, Identification of faulty sensors using principal component analysis, AIChE Journal, 42 (10) (1996) 2797-2812

[29] P. Amann, E. A. Garcia, B. Koeppen-Seliger, P. M. Frank, Knowledge- and data-based models for fault diagnosis, Systems Analysis Modeling Simulation, 35 (1) (1999) 25-44

[30] T. Kourti, P. Nomikos, J. F. MacGregor, Analysis, monitoring and fault diagnosis of batch processes using multiblock and multiway PLS, Journal of Process Control, 5 (4) (1995) 277-284

[31] I.-C. Chang, C.-C. Yu, C.-T. Liou, Model-based approach for fault diagnosis. 1. Principles of deep model algorithm, Industrial & Engineering Chemistry Research, 33, (6) (1994) 1542-1555

[32] Y. Zhang, J. Jiang, Interacting multiple-model based fault detection, diagnosis and fault-tolerant control approach, in Proceedings of the IEEE Conference on Decision and Control, 4, 3593-3598, 1999

[33] V. Venkatasubramanian, R. Vaidyanathan, Process fault detection and diagnosis using neural networks I. Steady-state processes, Computers and Chemical Engineering, 14 (1990) 699-712

[34] C. M. Shen, T. Bi, Z. Yan, Y. Wen, On-line fault section estimation in power systems with radial basis function neural network, International Journal of Electrical Power & Energy Systems, 24 (2002) 321-328

[35] M. Catelani, A. Fort, Fault diagnosis of electronic analog circuits using a radial basis function network classifier, Measurement, 28 (2000) 147-158

[36] C. M. Bishop, Neural Networks for Pattern Recognition, Oxford Univ. Press, New York, 1995

[37] K. Funahashi, On the approximate realization of continuous mappings by neural networks, Neural Networks, 2 (1989) 183-192

[38] N. E. Cotter, The Stone-Weierstrass theorem and its application to neural networks, IEEE Trans. on Neural Networks, 1 (1990) 290-295

[39] Y. Ito, Representation of functions by superpositions of a step or sigmoid function and their applications to neural network theory, Neural Networks, 4 (1991) 385-394

[40] M. Borairi, H. Wang, Actuator and sensor fault diagnosis of non-linear dynamic systems via genetic neural networks and adaptive parameter estimation technique, in Proc. of the 1998 IEEE International Conference on Control Applications. 278-282, IEEE Press, Piscataway, USA, 1998

[41] J. B. Gomm, Adaptive neural network approach to on-line learning for process fault diagnosis, Transactions of the Institute of Measurement and Control , 20 (1998) 144-152

[42] A. Alessandri, M. Baglietto, T. Parisini, Robust model-based fault diagnosis using neural nonlinear estimators, in Proc. of the 1998 37th IEEE Conference on Decision and Control, 72-77, IEEE Press, Piscataway, NJ, 1998

[43] D. L. Yu, J. B. Gomm, D. Williams, Sensor fault diagnosis in a chemical process via RBF neural networks, Control Engineering Practice, 7 (1999) 49-55

[44] M. Y. Chow, B. Li, and G. Goddu, Intelligent motor fault detection, in L. C. Jain, R. P. Johnson, Y. Takefuji, and L. A. Zadeh, (eds) Knowledge-based Intelligent Techniques in Industry, 191-223, CRC Press, New York, 1999

[45] Y. Tian, J. Zhang and J. Morris, Optimal control of a fed-batch bioreactor based upon an augmented recurrent neural network model, Neurocomputing, 48 (2002) 919-936

[46] S. A. Billings, Q. M. Zhu, Nonlinear model validation using correlation tests, International Journal of Control, 60 (6) (1994), 1107-1120

[47] J. Moody, C. J. Darken, Fast learning in networks of locally-tuned processing units, Neural Compututation, 1(2) (1989), 281-294

[48] S. Chen, P. M. Grant, and C. F. N. Cowan, Orthogonal least-squares algorithm for training multioutput radial basis function networks, Radar and Signal Processing, IEE Proceedings F, 139 (1992) 378 –384

[49] C. Kravaris, S. Palanki, Robust nonlinear state feedback under structured uncertainty, AIChE Journal 34 (1988) 1119-1127

[50] M. Soroush, C. Kravaris, Optimal design and operation of batch reactors. 2. A case study, Industrial & Engineering Chemistry Research, 32 (1993) 882-893

[51] D. Marquardt, An algorithm for least squares estimation of nonlinear parameters, SIAM Journal of Applied Mathematics, 11 (1963) 431-441

[52] Chemical Engineering Department, Texas A&M University, Chemical Engineering Unit Operations Laboratory II (lab instruction), Texas A&M University, College Station, TX, 2000

[53] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, Classification And Regression Trees, Wadsworth, Belmont, CA, 1984

[54] J. N. Morgan, J. A. Sonquist, Problems in the analysis of survey data, and a proposal, J. Am. Stat. Assoc., 58 (1963) 415-434

[55] E. L. Russell, L. H. Chiang, R. D. Braatz, Data-Driven Techniques for Fault Detection and Diagnosis in Chemical Processes, Springer-Verlag, London, 2000

[56] R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis, John Wiley & Sons, New York, 1973

[57] H. B. Aradhye, B. R. Bakshi, and R. Strauss, Process monitoring by PCA, dynamic PCA and multiscale PCA – Theoretical analysis and disturbance detection in the Tennessee Eastman process, in AIChE Annual Meeting, paper 224g, 1999

[58] G. Chen and T. J. McAvoy, Predictive online monitoring of continuous processes, J. of Process Control, 8, (1997) 409-420

[59] J. J. Downs and E. F. Vogel, A plant-wide industrial-process control problem, Computers & Chemical Engineering, 17, (1993) 245-255

[60] R. D. Braatz, Multiscale Systems Research Laboratory (web page) http://brahms.scs.uiuc.edu, 2002

[61] P. R. Lyman, C. Georgakis, Plant-wide control of the Tennessee Eastman problem, Computer and Chemical Engineering, 19, (1995) 321-331

APPENDIX A

INTRODUCTION TO FEED-FORWARD NEURAL NETWORKS

1. Architecture of Feed-Forward Neural Networks

Although there are already a number of textbooks on neural networks, it is still necessary to briefly introduce the basic concepts of neural networks.

Neural networks attempt to mimic the computational structures of the mammal brains by a nonlinear mapping between input and output that consists of interconnected nodes arranged in layers. The layers are connected such that the signal on the input side can propagate through the network and reach the output side. Neural network behaviors are determined by the transfer functions of the units, network topology, and connection pattern of layers.

Among all forms of neural networks, the two-layer feed forward network has been most popular (see Fig. A.1 for an example). This class of networks consists of two layers of nodes, namely the hidden layer and the output layer, and two layers of weights serving as the connections between input and the hidden layer, as well as between the hidden layer and the output layer. No connection is allowed within its own layer and the information flow is one direction only. The weights represent the strength of connection between two linked nodes. Neural network training process adjusts the weights to achieve the best approximation between the network output and the desired output. Besides regular nodes, bias nodes that supply a constant output are connected to each

nodes in the hidden and the output layers. The bias node provides an activation threshold for the nodes, and is essential in order to classify network inputs into various subspaces.

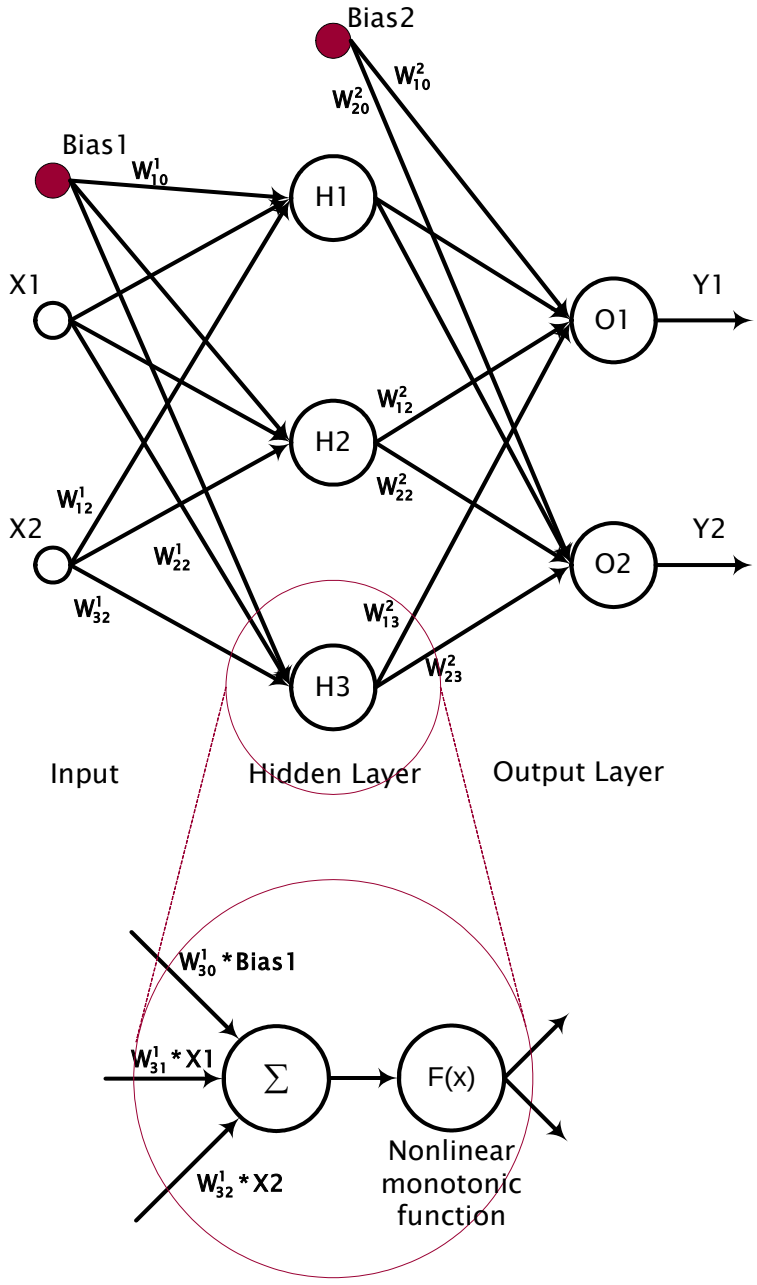


Fig. A.1 A two layer feed-forward neural network

Each node in the hidden and the output layers is described by a transfer function. Usually a logistic sigmoid function is used,

$$F(x) = \frac{1}{1+e^{-x}} \tag{A.1}$$

which is plotted in Fig. A.2.

Some other transfer functions can also be used, such as tanh function,

$$F(x) = \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \tag{A.2}$$

ramp function,

$$F(x) = \begin{cases} 0, & x \le 0 \\ x, & x > 0 \end{cases} \tag{A.3}$$

and step function,

$$F(x) = \begin{cases} -1, & x < 0 \\ 1, & x \ge 0 \end{cases} \tag{A.4}$$

However, for a specific feed-forward neural network, the nodes in the hidden and the output layers are usually fixed on the same transfer function, e.g., logistic sigmoid function for the hidden nodes and ramp function for the output nodes.

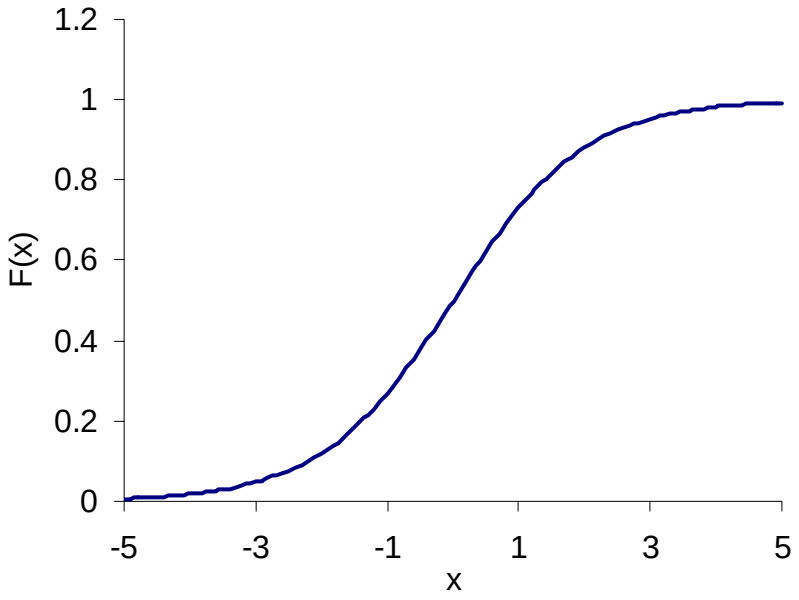


Fig. A.2 Plot of the logistic sigmoid transfer function

As a nonlinear modeling tool, feed-forward neural networks have a number of advantages compared to conventional statistical methods. Conventional statistical methods require explicit formulation of the functions, which is not always readily available for many nonlinear problems. If these functions are incorrectly specified, the regression will not be satisfactory. Furthermore, convergence in nonlinear regression also requires extensive mathematical and numerical expertise. In the contrary, neural networks do not need a specific form of the correlation, therefore is very fast to implement. Neural network modeling has other benefits such as when data used for training can are noisy or incomplete, continuous learning through new training data can be applied to improve performance.

2. Training of Feed-Forward Neural Networks

Neural networks utilize the network weights and input values to calculate outputs, take a two-layer network as the example:

$$y_k = \tilde{g}\left(\sum_{j=0}^{M} w_{kj}^{(2)} g(\sum_{i=0}^{d} w_{ji}^{(1)} x_i) \right) \tag{A.5}$$

where

d: number of inputs

M: number of hidden nodes

$w_{ji}^{(1)}$: a weight in the first layer, going from input i to hidden unit j

$w_{kj}^{(2)}$: a weight in the second layer, going from hidden unit j to output node k

g : hidden layer transfer function

$\tilde{g}$: output layer transfer function

y_k : kth output

x_i : ith input

The objective of network training is to minimize the sum-of-squares error with respect to weight vector $\mathbf{w}$, given by a sum over all patterns in the training set, and over all outputs, of the form

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{c} \left(y_k(\mathbf{x}^n; \mathbf{w}) - t_k^n \right)^2 \tag{A.6}$$

where

N: the number of training patterns

c: the number of outputs

$\mathbf{x}^n$: input vector

t_k^n : target value for output node k when the input vector is $\mathbf{x}^n$

The popular training algorithm, error back-propagation, involves a forward path calculating the outputs and backward path updating the weight vector. In the *m*th training iteration, the weight is adjusted as

$$w_{ji}^{(m)} = w_{ji}^{(m-1)} + \Delta w_{ji}^{(m)} \tag{A.7}$$

where

$w_{ji}^{(m)}$: the weight between the *j*th node of the upper layer and the *i*th node of the lower layer, in the *m*th training iteration

$w_{ji}^{(m-1)}$: the weight between the *j*th node of the upper layer and the *i*th node of the lower layer, in the *(m-1)*th training iteration

$\Delta w_{ji}^{(m)}$: the weight adjustment

Weight adjustment is given by

$$\Delta w_{ji}^{(m)} = \eta \delta_j^{(m)} o_i^{(m)} + \alpha \Delta w_{ji}^{(m-1)} \tag{A.8}$$

where

η: learning rate, providing the step size during gradient decent. To achieve fast convergence, the largest possible value that does not lead to oscillation is used.

$\delta_j^{(m)}$: error signal of the *j*th node in the *m*th training iteration

$o_i^{(m)}$: output value of the *i*th node of the previous layer in the *m*th training iteration.

α: momentum term, $0<\alpha<1$, optimized to achieve fast convergence

If j is an output layer node, $\delta_j^{(m)}$ is given by

$$\delta_j^{(m)} = (t_j^{(m)} - y_j^{(m)})\tilde{g}'(\sum_i w_{ji}^{(m)} o_i^{(m)} + w_{jo}^{(m)}) \quad \text{(A.9)}$$

$t_j^{(m)}$: target value for output layer node j

$y_j^{(m)}$: network output value of node j

$\tilde{g}'$: derivative of output layer transfer function

$w_{jo}^{(m)}$: the weight between the *o*th node of the hidden layer and the *j*th node of the output layer, in the *m*th training iteration

If j is a hidden layer node,

$$\delta_j^{(m)} = g'(\sum_i w_{ji}^{(m)} o_i^{(m)} + w_{jo}^{(m)}) \sum_k \delta_k^{(m)} w_{kj}^{(m)} \quad \text{(A.10)}$$

where g' is the derivative of the hidden layer transfer function.

As one can see, error back propagation algorithm calculates an error for each node in the output and hidden layers using equation (A.9) and (A.10), then recursively updates the weights of all the nodes using equation (A.8), starting from the output layer to the hidden layer.

3. Computational efficiency of error back propagation

In a practical neural network, the number of weights W is much larger than the number of nodes, therefore, the computational cost in both the forward and backward propagation phases is $O(W)$. Thus for a single training pattern, the computational complexity of error back propagation is $O(W)$. For a total of N training patterns, the computational complexity becomes $O(NW)$.

4. Implementation issues

4.1 Data pre-processing

In principle, neural networks can perform essentially arbitrary nonlinear mapping between the raw input data directly and the required final output values. Such an approach generally gives poor results because the training of the neural network may involve a nonlinear optimization algorithm. In many applications, data pre-processing is one of the most significant factors in determining the network performance.

The simplest pre-processing may take the form of a linear transformation of the input and the output data. Input variables are often correlated, thus more complex pre-

processing may include dimensionality reduction and the incorporation of prior knowledge, which also reduces network size and training time.

4.2 Network topology

It is generally accepted that in order to perform arbitrary nonlinear mapping between sets of data, only one hidden layer is necessary in a neural network using sigmoid transfer function.

The number of hidden nodes is decided by the nonlinearity of the mapping, the amount of data, and error tolerance. However, there is no universal method to determine how many hidden nodes are needed in a two-layer feed-forward neural network. Too small number of hidden nodes may not achieve a satisfactory accuracy and too large the number may result in poor generality. It is often practical to start with a small number of hidden nodes and gradually increase the number.

4.3 Convergence

The objective of neural network training is to minimize the error function $E(\mathbf{w})$ (A.6), which is typically a highly nonlinear function of the weights $\mathbf{w}$ and there may exist many local minima that will trap the training algorithm. The momentum factor α, which tends to keep the weight changes moving in a certain direction, allows the training algorithm to slip over small local minima and reach the global minimum. An alternative approach is to start the training again using a different set of initial weights if it is found that the error function keeps oscillating around a set of weights without improvement.

APPENDIX B

MATLAB CODE

5. Polymerization reactor subroutine

```
function f=fedbatch(t, x)
f=zeros(4,1);
Cm=x(1);
Ci=x(2);
T=x(3);
Tj=x(4);

% Parameters
m=1.26;

% control input
u= 0.00001*(0.0419*(t/3600)^3 - 0.145*(t/3600)^2 -
0.1335*(t/3600) + 319.5);

% Modeling equations
f(1)=-.2706652952e-27*(1+.8458069884e-1*Cm)*Cm/exp(-
352.0958084                        *1/T)/exp(2.3-
.1916666667e-2*(601.00-100*Cm)/(.6997288005+ .4277900178e-
1*Cm))*(.20-.821e-5*(T-387)^2-.2500000000e-4*(601.00-
100*Cm)/(.6997288005+.4277900178e-
1*Cm))/exp(17485.02994*1/T)* (.1193640e24*exp(-
352.0958084*1/T)*exp(-15449.10180*1/T)*Ci+2*sqrt
(.3561941124e46*exp(-352.0958084*1/T)^2*exp(-
```

```
15449.10180*1/T)^2* Ci^2+.4241271488e62*exp(-
352.0958084*1/T)*exp
(2.3-.1916666667e-2*(601.00-
100*Cm)/(.6997288005+.4277900178e-1*Cm))^2/(.20-.821e-5*(T-
387)^2-.2500000000e-4*(601.00-
100*Cm)/(.6997288005+.4277900178e-
1*Cm))^2*exp(17485.02994*1/T) ^2*exp(-
15449.10180*1/T)*Ci))*(.492e6*exp(-2191.616766*1/T)/
(1+.4409514079e-35/exp(-352.0958084*1/T)/exp(2.3-
.1916666667e-2*(601.00-100*Cm)/(.6997288005+.4277900178e-
1*Cm))^2*(.20-.821e-5*(T-387)^2-.2500000000e-4*(601.00-
100*Cm)/(.6997288005+.4277900178 e-
1*Cm))^2/exp(17485.02994*1/T)*(.1193640e24*exp(-352.0958084
*1/T)*exp(-15449.10180*1/T)*Ci+2*sqrt(.3561941124e46*exp(-
352.0958084*1/T)^2*exp(-
15449.10180*1/T)^2*Ci^2+.4241271488e62 *exp(-
352.0958084*1/T)*exp(2.3-.1916666667e-2*(601.00-
100*Cm)/(.6997288005+.4277900178e-1*Cm))^2/(.20-.821e-5*(T-
387)^2-.2500000000e-4*(601.00-
100*Cm)/(.6997288005+.4277900178e-
1*Cm))^2*exp(17485.02994*1/T)^2*exp(-
15449.10180*1/T)*Ci))*exp(-2191.616766*1/T)/exp(-
14011.97605*1/T))+.466e10*exp(-8922.155689*1/T));

f(2)=-.105e16*exp(-15449.10180*1/T)*Ci-.2289305982e-
28*Cm^2/exp(-352.0958084*1/T)/exp(2.3-.1916666667e-
2*(601.00-100*Cm)/ (.6997288005+.4277900178e-1*Cm))*(.20-
.821e-5*(T-387)^2-.2500000000e-4*(601.00-
100*Cm)/(.6997288005+.4277900178e-
```

```
1*Cm))/exp(17485.02994*1/T)*(.1193640e24*exp(-
352.0958084*1/T) *exp(-
15449.10180*1/T)*Ci+2*sqrt(.3561941124e46*exp(-
352.0958084*1/T)^2*exp(-
15449.10180*1/T)^2*Ci^2+.4241271488e62* exp(-
352.0958084*1/T)*exp(2.3-.1916666667e-2*(601.00-100*Cm)/
(.6997288005+.4277900178e-1*Cm))^2/(.20-.821e-5*(T-387)^2-
.2500000000e-4*(601.00-100*Cm)/(.6997288005+.4277900178e-
1*Cm))^2*exp(17485.02994*1/T)^2*exp(-15449.10180*1/T)*Ci))
*(.492e6*exp(-2191.616766*1/T)/(1+.4409514079e-35/exp(-
352.0958084*
1/T)/exp(2.3-.1916666667e-2*(601.00-100*Cm)/(.6997288005
+.4277900178e-1*Cm))^2*(.20-.821e-5*(T-387)^2-.2500000000e-
4*(601.00-100*Cm)/(.6997288005+.4277900178e-1*Cm))^2/
exp(17485.02994*1/T)*(.1193640e24*exp(-
352.0958084*1/T)*exp(-
15449.10180*1/T)*Ci+2*sqrt(.3561941124e46*exp(-
352.0958084*1/T)^2*exp(-15449.10180*1/T)^2*Ci^2+
.4241271488e62*exp(-352.0958084*1/T)*exp(2.3-.1916666667e-
2*(601.00-100*Cm)/(.6997288005+.4277900178e-1*Cm))^2/(.20-
.821e-5*(T-387)^2-.2500000000e-4*(601.00-
100*Cm)/(.6997288005+ .4277900178e-
1*Cm))^2*exp(17485.02994*1/T)^2*exp (15449.10180*1/T)*
Ci))* exp (-2191.616766*1/T)/exp(-
14011.97605*1/T))+.466e10*exp(-8922.155689*1/T));

f(3)=.4055679109e-20*exp(-2191.616766*1/T)/(1+.4409514079e-
35/exp (-352.0958084*1/T)/exp(2.3-.1916666667e-2*(601.00-
100*Cm)/(.6997288005+.4277900178e-1*Cm))^2*(.20-.821e-5
```

```
*(T-387)^2-.2500000000e-4*(601.00-
100*Cm)/(.6997288005+.4277900178e-
1*Cm))^2/exp(17485.02994*1/T)*(.1193640e24*exp(-
352.0958084*1/T) *exp(-
15449.10180*1/T)*Ci+2*sqrt(.3561941124e46*exp(-
352.0958084*1/T)^2*exp(-
15449.10180*1/T)^2*Ci^2+.4241271488e62 *exp(-
352.0958084*1/T)*exp(2.3-.1916666667e-2*(601.00-
100*Cm)/(.6997288005+.4277900178e-1*Cm))^2/(.20-.821e-5*(T-
387)^2-.2500000000e-4*(601.00-
100*Cm)/(.6997288005+.4277900178e-
1*Cm))^2*exp(17485.02994*1/T)^2*exp(-
15449.10180*1/T)*Ci))*exp(-2191.616766*1/T)/exp(-
14011.97605*1/T))/exp(-352.0958084*1/T)/exp(2.3-
.1916666667e-2*(601.00-100*Cm)/
(.6997288005+.4277900178e-1*Cm))*(.20-.821e-5*(T-387)^2-
.2500000000e-4*(601.00-100*Cm)/(.6997288005+.4277900178e-
1*Cm))/exp(17485.02994*1/T)*(.1193640e24*exp(-
352.0958084*1/T) *exp(-
15449.10180*1/T)*Ci+2*sqrt(.3561941124e46*exp(-
352.0958084*1/T)^2*exp(-15449.10180*1/T)^2*Ci^2 ...
+.4241271488e62*exp(-352.0958084*1/T)*exp(2.3-.1916666667e-
2*(601.00-100*Cm)/(.6997288005+.4277900178e-1*Cm))^2/(.20-
.821e-5*(T-387)^2-.2500000000e-4*(601.00-
100*Cm)/(.6997288005+.4277900178 e-
1*Cm))^2*exp(17485.02994*1/T)^2*exp(-
15449.10180*1/T)*Ci))*Cm/ (1+.8458069884e-1*Cm)-.38e-
2*(1+.50833*(1-.1663893511*Cm)/(1+
.1663893511*Cm))*(.2+.8*exp(-7*(1-.1663893511*Cm)^3
```

```
/(1+.1663893511*Cm)^3))*(T-Tj);

f(4)=.8e-3*(1+.50833*(1-.1663893511*Cm)/(1+.1663893511
*Cm)) *(.2+.8*exp(-7*(1-
.1663893511*Cm)^3/(1+.1663893511*Cm)^3))*(T-Tj)+.10841-
.37e-3*Tj+m*u;
```

6. Generating data with faults for the neural network approach

```
% main program: generating data for neural network approach
y0=[6.01 0.13 319.5 293]';
% Adjust sampling rate by specify the middle number
t=0:90:21600;
length_t=length(t);
[t,y]=ode45('polyreactor', t ,y0);
% noise generation
n1=0.002 * randn(length_t, 1);
n2=0.001 * randn(length_t, 1);
n3=0.01 * randn(length_t, 1);
n4=0.01*randn(length_t, 1);

noise=[n1 n2 n3 n4];
% add noise to y
yn=y+noise;
%plot(t, yn);
%pause;

% reproduce u
u=zeros(length_t, 1);
i=1;
for t1=0:180:21600
    u(i,1)= 0.00001*(0.0419*(t1/3600)^3 - 0.145*(t1/3600)^2
- 0.1335*(t1/3600) + 319.5);
    i=i+1;
end
```

```
%plot(t,q)

% cleaning workspace
clear n1 n2 n3 n4 n5 noise t1 i y0 y;

yn=real(yn);
u=real(u);

% prepare data for neural nets
% input 1~6: Cm(t-1) ... Cm(t-6)
% input 7~12: u(t-1) ... u(t-6)
% output : Cm(t), Ci(t), T(t), Tj(t)
in=zeros(length_t-6, 12);
out=zeros(length_t-6,4);

% Normalization
normalized_yn=zeros(length_t, 4);
normalized_yn(:,1)=(yn(:,1) -
mean(yn(:,1))./(max(yn(:,1))-min(yn(:,1)));
normalized_yn(:,2)=(yn(:,2) -
mean(yn(:,2))./(max(yn(:,2))-min(yn(:,2)));
normalized_yn(:,3)=(yn(:,3) -
mean(yn(:,3))./(max(yn(:,3))-min(yn(:,3)));
normalized_yn(:,4)=(yn(:,4) -
mean(yn(:,4))./(max(yn(:,4))-min(yn(:,4)));

normalized_u=zeros(length_t,1);
normalized_u=(u-mean(u))./(max(u)-min(u));
```

```
% add jacket fouling fault
plant_output=normalized_yn(7:length(yn(:,4)),:);
temp=plant_output;
% plant_output is the final values used for residual
generation
plant_output(151:200,4)=temp(151:200,4)*.8;

% prepare input for neural model
temp=normalized_yn;
normalized_yn(51:100,3)=temp(51:100,3)*.8;
for i=7:length_t
    in(i-6,6)=normalized_yn(i-1,4);
    in(i-6,5)=normalized_yn(i-2,4);
    in(i-6,4)=normalized_yn(i-3,4);
    in(i-6,3)=normalized_yn(i-4,4);
    in(i-6,2)=normalized_yn(i-5,4);
    in(i-6,1)=normalized_yn(i-6,4);

    in(i-6,12)=normalized_yn(i-1,3);
    in(i-6,11)=normalized_yn(i-2,3);
    in(i-6,10)=normalized_yn(i-3,3);
    in(i-6,9)=normalized_yn(i-4,3);
    in(i-6,8)=normalized_yn(i-5,3);
    in(i-6,7)=normalized_yn(i-6,3);

end
in=in';
```

7. Generating training data for CART approach

```
load d00.dat;
load d01.dat
load d02.dat
load d03.dat
d00=d00';
d00=[d00,zeros(500,1)];
d01=[d01,zeros(480,1)];
d02=[d02,zeros(480,1)];
d03=[d03,zeros(480,1)];
d01(:,53)=ones(480,1);
d02(:,53)=2*ones(480,1);
d03(:,53)=3*ones(480,1);
d00=[d00(:,1:22),d00(:,42:53)];
d01=[d01(:,1:22),d01(:,42:53)];
d02=[d02(:,1:22),d02(:,42:53)];
d03=[d03(:,1:22),d03(:,42:53)];
training=[d00;d01;d02;d03];
```

8. Generating testing data for CART approach

```
load d00_te.dat;
load d01_te.dat
load d02_te.dat
load d03_te.dat
d00_te=[d00_te,zeros(960,1)];
d01_te=[d01_te,zeros(960,1)];
d02_te=[d02_te,zeros(960,1)];
d03_te=[d03_te,zeros(960,1)];
d01_te(:,53)=ones(960,1);
d02_te(:,53)=2*ones(960,1);
d03_te(:,53)=3*ones(960,1);
d00_te=[d00_te(:,1:22),d00_te(:,42:53)];
d01_te=[d01_te(:,1:22),d01_te(:,42:53)];
d02_te=[d02_te(:,1:22),d02_te(:,42:53)];
d03_te=[d03_te(:,1:22),d03_te(:,42:53)];
testing=[d00_te;d01_te;d02_te;d03_te];
```

9. CART main program

```
function m=fda_cart_main(train_x,train_y,test_x,test_y)
tic;
Wp=FDA(train_x, train_y);
fda_train_x=train_x*Wp;
fda_test_x=test_x*Wp;
T = optimal_tree_fit(fda_train_x,train_y);
YFIT_num = tree_evaluate(T, fda_test_x);
m=misclass(YFIT_num, str2num(test_y));
toc



function m=FDA(train_x, train_y)
% Fisher Discriminant Analysis
% train_x: numerical, training data, predicting variables
% train_y: string, training data, classes
%
% function returns Wp -- loading vector
row_size_of_set=size(train_x)*[1;0];
column_size_of_set=size(train_x)*[0;1];
% backup train_y value
temp=train_y;
% convert to numerical
train_y=str2num(train_y);
% maximum number of classes = 20
boundary_of_classes=zeros(20,2);
current_class=1;
boundary_of_classes(current_class,1)=1;
```

```
counter=1;
% find class boundaries
while counter < row_size_of_set
                                                        wh
ile train_y(counter)== 0 && counter < row_size_of_set
        counter=counter+1;
     end
    boundary_of_classes(current_class,2)=counter-1;
    current_class=current_class+1;
    boundary_of_classes(current_class,1)=counter;
    train_y=train_y-ones(row_size_of_set,1);
end
boundary_of_classes(current_class-1,2)=counter;
number_of_class=current_class-1;
% restore train_y
train_y=temp;
% total mean and covariance
avgT=mean(train_x);
sT=(row_size_of_set-1)*cov(train_x);
% calculating class averages and covariances
% Maximum number of classes equals 20
% sb -- between class, sw -- within class
s=zeros(column_size_of_set,column_size_of_set, 20);
avg=zeros(20,column_size_of_set);
sb=zeros(column_size_of_set,column_size_of_set);
sw=zeros(column_size_of_set,column_size_of_set);
for i=1:number_of_class
```

```
avg(i,:)=mean(train_x(boundary_of_classes(i,1):boundary_of_
classes(i,2),:));
    s(:,:,i)=(boundary_of_classes(i,2)-
boundary_of_classes(i,1))*...

cov(train_x(boundary_of_classes(i,1):boundary_of_classes(i,
2),:));
    sw=sw+s(:,:,i);
    sb=sb+(boundary_of_classes(i,2)-
boundary_of_classes(i,1)+1)*(avg(i,:)-avgT)'*(avg(i,:)-
avgT);
end
% testing if sT=sb+sw; erase this line after debug
% sT-sb-sw
[V,D]=eig(sb,sw);
diag(D)
order=input('enter the FDA order you chose:\n');
m=V(:,1:order);

function m=cart_main(train_x,train_y,test_x,test_y)
tic;
T = optimal_tree_fit(train_x,train_y);
YFIT_num = tree_evaluate(T, test_x);
m=misclass(YFIT_num, str2num(test_y));
toc

function m=misclass(actual, target)
```

```
% Compare the actual CART classification with desired
values on the test
% set
% actual and target:both of them are integer.
%
size_of_set=size(actual);
% maximum number of classes = 20
boundary_of_classes=zeros(20,2);
% backup target value
temp=target;
current_class=1;
boundary_of_classes(current_class,1)=1;
counter=1;
while counter < size_of_set(1)
     while target(counter)== 0 && counter < size_of_set(1)
        counter=counter+1;
     end
    boundary_of_classes(current_class,2)=counter-1;
    current_class=current_class+1;
    boundary_of_classes(current_class,1)=counter;
    target=target-ones(size_of_set);
end
boundary_of_classes(current_class-1,2)=counter;
% restore target value
target=temp;
number_of_class=current_class-1;
% last component stores overall misclassification rate
misclass=zeros(number_of_class+1,1);
temp=sign(abs(actual - target));
```

```
for i=1:number_of_class

misclass(i,1)=sum(temp(boundary_of_classes(i,1):boundary_of
_classes(i,2),1))/(boundary_of_classes(i,2)-
boundary_of_classes(i,1)+1);
end
misclass(number_of_class+1,1)=sum(temp)/size_of_set(1);
m=misclass;



function T = optimal_tree_fit(train_x,train_y_cart)
% Find the optimal classification tree by cross-validation
% Inputs: train_x -- numerical
% train_y_cart -- string
%
% Output: T -- Classification tree
%
% Get the original tree
T=treefit(train_x, train_y_cart, 'prune', 'on');
% Optimizing the CART using cross-validation
[c,s,n,best] = treetest(T,'cross',train_x,train_y_cart);
% Prune the tree
T = treeprune(T,'level',best);

function YFIT_num = tree_evaluate(T, test_x)
% Test the classification tree using testing data
% returns numerical values of classes
YFIT = treeval(T,test_x);
YFIT_num=double(YFIT)-ones(size(YFIT));
```

VITA

Born in July 1975 in Luoyang, Henan, China, Yifeng Zhou holds a B.S. degree with honors in chemical engineering from Xian Jiao-Tong University (1996), and a M.S. degree from the Research Institute of Petroleum Processing (1999). During his Ph.D. studies at Texas A&M University, he worked for the Mary Kay O'Connor Process Safety Center for four years as a graduate researcher focusing on abnormal situation management and process monitoring. In Fall 2003 and Spring 2004, Yifeng worked as a college intern with Applied Materials, Inc. in Austin, Texas.

Yifeng Zhou's permanent address is:

Luoyang Zhonghao Gongsi, #20-9
Yiyang, Henan, 471600
China