PATTERN RECOGNITION APPROACHES TO STATE IDENTIFICATION IN CHEMICAL PLANTS

BY

WANG CHENG

NATIONAL UNIVERSITY OF SINGAPORE

2003

PATTERN RECOGNITION APPROACHES TO STATE IDENTIFICATION IN CHEMICAL PLANTS

WANG CHENG

(B.Eng., USTB, P.R China)

A THESIS SUBMITTED

FOR THE DEGREE OF PHILOSOPHY DOCTOR

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2003

Acknowledgements

I would like to express my deepest gratitude to my research supervisor, Dr. Rajagopalan Srinivasan for his excellent guidance and valuable ideas. His wealth of knowledge and accurate foresight have greatly impressed and enlightened me. I am indebted to him for his care and advice not only in my academic research but also in my daily life. Without him, my research would not be successful.

I am also grateful to Prof. Ho Weng Khuen and Prof. Lim Khiang Wee for their stimulating suggestions and clever insights which benefited my research a lot.

I would like to thank my lab mates in iACE lab — Kashyap, Anand and Mingsheng for their abundant chemical process knowledge, which is very helpful to locate problems.

In addition, I would like to give due acknowledgement to National University of Singapore, for granting me research scholarship and funds needed for the pursuit of my Ph.D degree. It has been a wonderful experience for me in NUS. I sincerely thank the University for this opportunity.

Finally, this thesis would not have been possible without the loving support of my family. I devote this thesis to them and hope that they will find joy in this humble achievement.

Contents

Summary

Applying operating state-based supervisory control to chemical process becomes more and more attractive since chemical processes operate in multiple steady state operating conditions and transition between them. Global process control using fixed control models and configurations leads to poor process performance and quality control when the process moves away from the pre-considered operating state. A local control strategy that adapts to the current process operating state is an optimal operating strategy. Monitoring of steady state and transition operations of industrial processes is the base to realize such a control strategy. In this thesis, three closely related problems towards the uses of effective operation have been addressed.

Offline clustering of process states in historical data can be used to compare different operating states. Different stages of a multi-step operation (such as startup of FCCU) can be assessed for similarity. Also, different runs of the same operation (such as catalyst loading) can be compared. These lead to improved understanding of transitions. Furthermore, by correlating features of successful runs to product properties, process efficiency, etc, process operations can be optimized. The obvious need for efficient and automatic identification of the different process states using large historical datasets, in lieu of manual annotation by an engineer provides the motivation for the work. Traditional clustering methods are computationally expensive and normally perform poorly on temporal signals. A two-step clustering method based on Dynamic Principal Component Analysis (DPCA) is proposed in this thesis. Temporal data are first classified into modes corresponding to quasi-steady states and transitions. Dynamic PCA based similarity measures are then used in the second phase to compare the different modes and the different transitions and cluster them. This methodology can be applied to high dimensional, temporal data and has low computational requirements.

Once offline clustering has provided the essential understanding of the process, an online classifier has to be built to monitor and identify the process state in real time. A number of techniques for this purpose have been developed. While each technique has its own advantages, artificial neural networks have been widely used in industrial applications because their ability to approximate any well-defined nonlinear function with arbitrary accuracy. However, one common problem arises during the training of neural network. Usually the structure of the network is decided based on the input dimensionality and the complexity of the underlying classes. A typical chemical process section has hundreds of sensors each generating thousands of observations every day. These data are noisy and contains patterns from different operating states. The construction of an accurate neural classifier for such multi-variate, multi-class temporal classification problem suffers from the "curse of dimensionality". Two new neural network structures — One-Variable-One-Network (OVON) and One-Class-One-Network (OCON) — that overcome this problem are proposed in this thesis. Both the architectures use a set of neural networks – in OVON there is one network for each variable, while in OCON, one network is used for each pattern class to be identified. In comparison to traditional monolithic neural networks, both the proposed architectures improve classification accuracy and minimize the training complexity. In addition, OVON is robust to sensor failures and OCON is well suited for addition of new pattern classes.

Context-based pattern recognition arises when the interpretation of a pattern varies across contexts. It is shown that the identification of the state of chemical or biological processes is context-dependent. The resulting one-to-many mapping between patterns and their classes cannot be adequately handled by traditional pattern recognition approaches. To address this problem, a neural network based architecture ─ operating state identification neural network (OSINN) ─ is proposed in this thesis. In OSINN, process measurements can be used as primary features for identifying the current process state, and the previous process state provides the context in which the primary features have to be interpreted. Three variations of the architecture, each using a different approach to identify change of context, are described.

All the proposed methods in this thesis are tested on a number of industrial-scale problems. Their performances are compared with traditional methods and analyzed in detail.

Nomenclature

a_{k_i}	The i^{th} element of k^{th} eigenvector $\{a_{k_1}, a_{k_2}, \cdots, a_{k_{nh}}\}$ obtained from dynamic PCA operation, $nh = l \times d$
$A_1, A_{2,\ldots}, A_l$	Regression parameters with number of *l*.
C_i	i^{th} class of a total number of *nm* classes $\{C_1, C_2, \ldots, C_{nm}\}$
$CN_{\hat{j}}$	$\hat{j}$ -th sub-network of OCON corresponding to $\hat{S}_{\hat{j}}$
d	Number of process variables
D	Distance between two vectors
$D_j^{nd_i}$	Feature vectors collection $\tilde{x}_i^{nd_i}(t) : [\tilde{x}_i(t), \tilde{x}_i(t-1), \cdots, \tilde{x}_i(t-l_i)]$ corresponding to each sub-state $S_j^{x_i}$.
$D_{\hat{j}}^{nd}$, $D_{\hat{k}}^{nd}$	Process feature vectors collection
$\widehat{D}^x(t)$	Rounded output of sub-state identification layer of OVON, $\widehat{S}^{x_1}(t) = \text{round}(S^{x_1}(t))$.
e	Positive real number
$f_{CN_{\hat{j}}}$	Mapping embedded in the $CN_{\hat{j}}$
f_{VN_i}	Mapping embedded in the VN_i
G	Transform function used in OSINN-N data preprocessor
H,O	$d \times k$ matrix of weights from PCA operation
i, j	Index for process variable, *i, j = 1... d*
$\hat{i}$, $\hat{j}$	Index for operating state, $\hat{i}, \hat{j} = 1 \ldots nk$
k	Number of PCs retained after PCA transform
l	Window size for feature vector
l_i	Time lag of process variable x_i for VN_i
$l_{\hat{j}}$	Time lag for $CN_{\hat{j}}$
L	Length of data window moving step
M_i^B	i^{th} mode of air blower section
M_i^R	i^{th} mode of regenerator section
nd	Dimensionality of process feature vector, *nd=dx(l+1)*
nk	Number of operating states $\{\widehat{S}_1, \widehat{S}_2, \cdots, \widehat{S}_{nk}\}$

nk_i	Number of sub-states of variable x_i
nm	Number of classes $\{C_1,C_2,...,C_{nm}\}$
ns	Total number of elements of time series S $(s_1,s_2, ... ,s_{ns})$
nt	Total number of elements of time series T $(t_1, t_2, ... , t_{nt})$
N_{mis}	Number of samples misclassified.
N_{total}	Number of samples for validation.
p,q	$(t\text{-}q)$ and $(t\text{-}p)$ represent two time instant
p_i	Probability $p_i(C_i \mid X_{new}^{nd})$ that X_{new}^{nd} belongs to i^{th} class C_i
P_i	Percentage of i^{th} eigenvalue over the sum of all eigenvalues
$PA(t)$	Process pattern identified by Data Pre-processor at time t
r	Resolution of edge detection in steady state identification
$\mathbb{R}^n$	n-dimensional real number space
$\hat{S}(t)$	Process operating state at time t, where $\hat{S}(t) \in \{\hat{S}_1, \hat{S}_2, \cdots, \hat{S}_{nk}\}$
$\hat{S}_{con}$	Contextual feature
$\hat{S}_{\hat{i}}$, $\hat{S}_{\hat{j}}$	$\hat{i}$ -th, $\hat{j}$ -th operating state of operating state set $\{\hat{S}_1, \hat{S}_2, \cdots, \hat{S}_{nk}\}$
$S_i^{x_i}$	i^{th} sub-state of variable x_i $S_i^{x_i} \in [S_1^{x_i}, S_2^{x_i}, \cdots, S_{nk_i}^{x_i}]$
$S^{x_i}(t)$	Sub-state of variable x_i at time t $S^{x_i}(t) \in [S_1^{x_i}, S_2^{x_i}, \cdots, S_{nk_i}^{x_i}]$
$\hat{S}_{\hat{i}}$	$\hat{i}$ -th operating state of operating state set $\{\hat{S}_1, \hat{S}_2, \cdots, \hat{S}_{nk}\}$
$\hat{S}_{pre}$	Previous operating state
S_M	Similarity factor of process modes
S_{PCA}	PCA similarity factor proposed by Krzanowski (1982)
S_{PCA}^{λ}	PCA similarity factor used by Singhal and Seborg (2001)
S_{DPCA}^{λ}	Proposed Dynamic PCA similarity factor
T_d	Dwell-time required for state change detection
T_e	Evaluation-interval
T_f	Threshold to define a steady state by Jiang (2003)
T_w	Size of moving data window for steady state identification
TS_{min}	Minimum duration of a mode
T_i^B	i^{th} transition of air blower section
T_i^R	i^{th} transition of regenerator section
U_1,U_2	Neurons in OSINN structures for the process pattern and the context,

	respectively
U^T	Eigenvectors matrix for dynamic PCA transform
VN_i	i^{th} sub-network of OVON corresponding to variable x_i
x_i	Value of process variable X_i
$\tilde{X}_i$	Normalized process variable X_i
X_H , X_L	High/low limits of the sensor range of variable X_i
X_{delay}	$nh \times (nn - l + 1)$ dimensional matrix containing process dynamics
X_{nn}^d	Historical data matrix
$X_{cen_i}^d$	Central vector of i^{th} cluster of $\{C_1, C_2, ..., C_{nm}\}$
X_{new}^{nd}	Feature vector whose class is unknown
$X^{nd}(t)$	*nd*-dimensional feature vector at time *t*
X^{nd_i}	nd_i-dimensional feature vector
$X^{nd_{\hat{j}}}$	$nd_{\hat{j}}$ -dimensional feature vector
$X^d(t-l)$	*d*-dimensional process vector $[x_1, x_2, \cdots, x_d]$ at time *t-l*
$\tilde{X}_{nn}^d$	Normalized historical data matrix based on measurement ranges
$\bar{X}_{nk}$	Dataset $\{\bar{X}(1), \bar{X}(2), \cdots, \bar{X}(t)\}$ containing all pre-processed process feature vectors generated from operating state $\hat{S}_{nk}$
$\bar{X}(t)$	Output of data pre-processor
y_k	The k^{th} score value obtained from dynamic PCA transform
Y_{cen}^k	The chosen central vector of current scores window
$Y_{\max}^k, Y_{\min}^k$	High and Low limits of the score matrix from PCA operation
Y^k	Score matrix constructed by the first *k* PCs from PCA operation
Y_i^k	The i^{th} vector of the scores set $[Y_1^k, Y_2^k, \cdots, Y_{nn}^k]$ obtained from PCA transform
$z_{\hat{j}}$	Output of $\hat{j}$ -th sub-network of OCON
$Z(t)$	Output of traditional neural network
λ_i	The i^{th} eigenvalue of eigenvalue set $\{\lambda_1, \lambda_2, \cdots\}$
$\hat{\mu}(t)$	Estimated mean of data in data window at time *t*
$\hat{\mu}_{M_i}$, $\hat{\mu}_{M_j}$	Estimated mean vector of process modes M_i and M_j respectively.
θ	Threshold of mean difference to define a steady state in a uni-variate

	process
θ_{ij}	Angle of the ith and jth PCs obtained after PCA transform
θ_d	Variation tolerance for steady state identification
θ_M	Threshold for two mean vectors' similarity comparison
θ_T	Threshold for two transitions' DPCA similarity comparison
$\theta_P, \theta_S, \theta_N$	Pre-defined threshold for state-change detector in OSINN
θ_r	User-defined threshold in regulator of OCON
ε	Ratio of the misclassified samples over the whole validation set

List of Figures

List of Tables

Chapter 1. Introduction

1.1 Introduction

Industrial processes are operated in a number of steady states named as operating modes and frequently undergo transitions among them. An operating mode is a particular process status with most variables varying in a narrow band. Small fluctuations caused by disturbances or process noise are allowed within a mode. A transition occurs when the process moves from one steady state to another. During a transition, state variables usually undergo a relatively large change. A transition could arise in many situations, like unit start up or shutdown, grade change or fluctuations caused by big disturbances, and faults. The product quality control during transitions is normally poor, and sometimes the energy and utility consumption high. Controlling the process to transit quickly and smoothly to the next state is important and can result in large benefit.

In traditional process monitoring and control strategies, the relevant control parameters and configurations, such as PID parameters, process models and alarm limits are uniformly applied for the entire process operation from start-up to shutdown. This set of parameters is normally tuned and set based on the main operating modes. However, many processes of concern to chemical engineers exhibit non-linear behavior, where the relationship between the controlled variable and the manipulated variable is dependent on the operating conditions. Examples of such processes include pH neutralization, exothermic chemical reactions, biological systems, and batch processes. While the low-level control constituted by feedback and feed-forward control loops is usually sufficient under normal conditions when the characteristics of the process are reasonably constant, as the operational conditions change during different operating states, the control set points often have to be adjusted accordingly to obtain the desired operation. In addition, for some advanced control techniques such as model based control, good process models are essential to guarantee a good performance. When the process moves to a different operating state, sometimes the embedded process models have to be adapted. Otherwise, the control performance will degrade. Therefore a supervisory control layer which can enable the lower layer level controllers to adapt to the current operating state is necessary. The corresponding local control strategy can be applied. To achieve such supervisory control, it is necessary to monitor the process variables and identify the current operating state in real time. Developing this supervisory control layer is the main goal of this thesis.

The identification of current process operating states can be considered as a pattern recognition problem. Some attributes of the process defined by user can be used to characterize the process. The unique behavior of the attributes within a particular operating state differentiates an operating state from others. The most frequently used features are online process variables, such as flowrate, temperature, pressure, level, and analyzer data. The measurements of these variables are monitored and recorded to provide the information of the process for operation or analysis purpose.

An offline analysis of the process and its operation has to be conducted before the construction of the online monitoring system. Clustering of process states in historical data can be used to compare operating conditions. Different stages of a multi-step operation (such as startup of FCCU) can be assessed for similarity. Also, different runs of the same operation (such as catalyst loading) can be compared. These lead to improved understanding of operating states. Furthermore, by correlating features of successful runs to product properties, process efficiency, etc, process operations can be optimized.

By clustering, the process is segmented to distinguish operating states, and the features of each operating state can then be extracted. If a clustering operation results in many trivial operating states without useful operation information, the construction of the on-line monitoring system will become difficult. On the other hand, if a clustering operation results in only a few states at a low resolution, the information provided will be inadequate. Therefore, an accurate analysis is needed. Several automated clustering techniques have been proposed in literature. One shortcoming of these clustering methods is that the number of clusters has to be specified *a priori*. In addition, most methods consider the entire process data monolithically and the temporal information is missed. These methods are therefore inapplicable for process states which are characterized by the temporal evolution. In this thesis, these problems for clustering are addressed.

An online operating state monitoring and identification system can be built based on the process knowledge provided by the clustering. The objective of this system is to extract useful information from the process measurements. The information obtained in the monitoring phase can be used to identify the current operating state by comparing the information with pre-stored operating state information. The construction of the online classifier is achievable for industrial processes because many chemical processes continue to operate through the same set of states without drastic changes for long periods. The same operating states can repeat with the same features as well as small deviations. Once the pattern of a state has been learnt, it can be used for future state identification. Therefore, the problem of constructing a supervised classifier is that of extracting and storing historical information such that relevant patterns can be retrieved and compared easily during on-line operations. In this thesis, artificial neural networks (ANNs) have been used for this purpose.

Artificial Neural Networks is attractive for industrial applications because theoretically it can approximate any well-defined nonlinear function with arbitrary accuracy. The main advantages of ANNs appear when dealing with hard problems, *e.g.*, in the case of significant overlapping patterns, high noise, and dynamically changing environments. Among the different types of neural networks, Elman recurrent network and Time Delay Neural Network (TDNN) have been frequently used for temporal information classification. The performances of these structures in terms of recognition accuracy are basically rather similar and there is no universal criterion for selecting a specific structure for a practical application. Usually the structure of the network is decided based on the input dimensionality and the complexity of the underlying classes. However, general neural network structure cannot scale well to the large-scale multivariate temporal patterns that occur in state identification. Specialized neural network architectures have been therefore developed. A typical chemical process section has hundreds of sensors each generating thousands of observations every day. These data are noisy and contains patterns from different operating states. The construction of an accurate neural classifier for such multi-variate, multi-class temporal classification problem suffers from the "curse of dimensionality". This is because classification is based not only on the process vector but also the temporal evolution. If the process has d variables and has a memory of l, the input to neural network will be of dimension $d \times (l+1)$. This high dimensionality introduces extra complexities such as amplifying the effect of noise, especially during transitions, and increasing the number of parameters needed to construct a classifier, and overlap among process patterns resulting from the time lag l. Therefore, training takes a considerable computation time and even then, the resultant network may perform poorly. In this thesis, two neural network structures are proposed to solve this problem. They overcome the "curse of dimensionality" by decomposing the initial identification problem to a set of sub-problems, which are less complex in terms of the dimensionality of inputs and the complexity of patterns. Consequently, the training of the system can be simplified and the accuracy of the network increased.

In many real-world domains, the context of a pattern has to be taken into the consideration in addition to the pattern itself. This is especially true for activities such as identifying and explaining unanticipated events and helping to handle them. Context is defined as the information that constrains problem solving without intervening in it explicitly. Many pattern recognition problems have to consider "context". For example, suppose we are attempting to distinguish healthy people (class A) from sick people (class B), using an oral thermometer. Context 1 consists of temperature measurements made on people in the morning, after a good sleep. Context 2 consists of temperature measurements made on people after heavy exercise. Sick people tend to have higher temperatures than healthy people, but exercise also causes higher temperature. When the two contexts are considered separately, diagnosis is relatively simple. If we mix the contexts together, correct diagnosis becomes more difficult. It is shown in this thesis that the identification of the state of chemical or biological processes is also context-dependent. The resulting one-to-many mapping between patterns and their classes cannot be adequately handled by traditional pattern recognition approaches which do not consider the context information. A novel neural network-based structure is proposed in this thesis to address this problem. It can employ context information in addition to process measurements to improve state identification accuracy.

1.2 About This Thesis

The importance of operating state based control strategies was discussed in above Section. As discussed, this requires the solving of three sub-problems: (1) data clustering, (2) temporal pattern recognition, and (3) context-based pattern recognition. The shortcomings of the existing methods were reviewed in Chapter 2. Novel methods for these problems have been developed in this thesis specifically.

In **Chapter 3**, the importance of process data clustering is discussed and a dynamic PCA-based multivariate clustering method is proposed. Clustering of process states in historical data can be used to compare operating conditions. These lead to improved understanding of operating states and their optimization.

A process unit's state can be classified into modes and transitions. A clustering method which is based on differentiating between the states—modes and transitions in the process is developed in chapter 3. It segments the multivariate process data by identifying steady state operating regimes. These steady states can therefore be used to segment the data into different operating modes and transitions. The operating states are then grouped into different clusters based on the similarity between them. If the similarity degree between two modes or two transitions is sufficiently large, they will be concluded as belonging to the same cluster. Therefore the proposed method includes two sub-problems: (1) Steady state identification, and (2) Similarity comparison.

During a steady state, most observations of state variable should be concentrated in a small region (in terms of their values) while the observations obtained during transitions will distribute in scattered manner. The procedure for state clustering can be summarized as: Firstly, PCA is performed on the auto-scaled historical data to reduce data dimensionality. The obtained scores are *k*-dimensional comprised of first *k* PCs.

Next, a data window with length T_w is moved along the dataset. Each k-dimensional vector Y_n^k within the window is compared with some randomly selected centers Y_{cen} and the distance D between Y_n^k and Y_{cen} calculated. If at least δ fraction of the vectors in the window lie within a short distance from the selected centers, the process is concluded to be within a mode during the current window. The data window is then moved forward by step size L and the process repeated.

After steady states are located, all remaining regions are then tagged as transitions. The segments are then divided into two groups containing modes and transitions respectively. Similarity comparison is carried out separately in two groups. A mode is characterized by constant variables. Hence, the mean is the principal property of the mode. The differences between elements of two means will be used to evaluate the dissimilarity degree of two segments. DPCA similarity factor is used in this thesis to compare two multivariate transitions. DPCA transformation is carried out on time-lagged sets to generate k PCs. The corresponding matrices of weights are denoted by H and O respectively. The DPCA similarity factor is defined based on the average value of the cosines of the angles between every two principal component of H and O. Once similar operating states are found, they are grouped into different clusters.

The two-step clustering strategy has been tested on data generated from ShadowPlant and Tennessee Eastman (TE) plant. The ShadowPlant is a simulator of Fluidized Catalytic Cracking (FCC) released by Honeywell while the Tennessee Eastman (TE) plant is a popular testbed for process systems applications such as plant-wide control, optimization, predictive control, faults diagnosis and signal comparison. The examination of the results reveals that in all cases the identified states agree with *a* *priori* process knowledge and similar transitions could be picked out by the DPCA factor.

Once the process data has been clustered into different modes and transitions, the obtained knowledge can be used to develop the online classifier to monitor the process even during non-steady state operation. This is discussed in **Chapter 4**. Due to the advantages mentioned, we adopt neural network as classification tools. The construction of an accurate neural classifier for the multivariate, multi-class, temporal classification problem suffers from the "curse of dimensionality". To address this, the One-Variable-One-Net (OVON) and One-Class-One-Net (OCON) architectures are proposed in chapter 4.

In OVON, the traditional network is replaced by a set of networks where each network processes only one variable. The OVON comprises of two layers: the sub-state identification layer and the unification layer. The sub-state identification layer consists of *d* sub-networks corresponding to *d* variables, each sub-network identifies the sub-state of a single variable. The outputs of the sub-state identification layer $[S^{x_1}(t), S^{x_2}(t), \cdots, S^{x_d}(t)]$ form the input to the unification layer where the process state of the entire process is classified based on the mapping: $\hat{S}(t) \leftarrow D^x(t):[S^{x_1}(t), S^{x_2}(t), \cdots, S^{x_d}(t)]$. The structure and training method are discussed in detail in Chapter 4. Another structure which can decompose the original problem into a number of simpler ones is the One-Class-One-Net (OCON) system. The system also consists of two layers: the sub-network identification layer and the regulator layer. The sub-network identification layer consists of *nk* neural networks, corresponding to *nk* operating states. All the networks share the same input variables at time *t*. A sub-network is trained to identify only a specific operating state. That is, only when data are generated from a particular state, the corresponding sub-network will output one. In the regulator layer, a set of rules are used to infer the operating state based on the *nk* networks outputs $[Z_1, Z_2, \cdots, Z_{nk}]$. Instead of the common method "winner-takes-all" strategy, we propose a novel rule to infer the final operating state. The proposed structures are tested on a number of units of the ShadowPlant simulator. Compared with traditional neural networks, OVON and OCON yield higher classification accuracy and require less training burden.

In **chapter 5**, the problem of context-based pattern recognition is discussed in detail. In pattern recognition, a feature can be considered as contextual information if it does not directly determine the class of a pattern. However, the absence of this feature would lead to ambiguous or erroneous classification. The presence of contextual features usually becomes evident when a change in the context leads to a radical change in the interpretation of a pattern (Brezillon, 1999). Traditional pattern recognition approaches are suitable for one-to-one or many-to-one mappings and cannot adequately characterize one-to-many situations, which arise in context-based pattern recognition problem.

A dynamic neural network architecture for context-based operating state identification network — OSINN — is proposed in chapter 5. Three variations of OSINN, each using a different approach to identify change of context, are described. OSINN includes three blocks: Context Manager, State Identification Block, and Data-preprocessor. A data-preprocessor is used to ameliorate the input data before it is used for state identification. Preprocessing can either be a normalization based on the contextual information $\hat{S}_{con}$ or a preliminary classification to identify the process pattern PA_i. The context manager detects changes in context and provides the correct contextual feature to the state identification block. The state identification block uses the contextual feature along with the primary features to identify the current operating state of the process.

The proposed strategy has been tested on data generated from the ShadowPlant simulator and a lab-scale fed-batch process. The results reveal that in all cases, the state identification accuracy is improved by OSINN.

Finally in **Chapter 6**, the summary of this work and conclusions are presented. Also recommendations for future enhancements are given in this chapter.

Chapter 2. Literature Review

As presented in the introduction of the thesis, an operating state based supervisory control becomes more and more crucial in modern industrial process. Rosen and Yuan (2001) have mentioned some reasons why a supervisory control is needed:

1. A process may display non-linear behavior when the operational conditions are far from the normal operating point, requiring changes to control set points.
2. During extreme operational conditions such as hydraulic shocks or toxicity, the aim of the operation may shift significantly. Thus, a higher-level control system is needed to determine the control set points or control structure of the low-level control systems.

In Rosen and Yuan's paper, an approach to automatic supervisory control of wastewater treatment operation is proposed. By integrating on-line monitoring and control, appropriate low-level controller set point and structures for the current operational state of the process can be determined. The authors declare that the plant can benefit a lot from local control strategy.

Another typical operating state based application is alarm management system. Along with the development of Distributed Control Systems (DCS), the problem called "alarm flood" has attracted more and more attention. A large number of alarms occur during upset conditions, and long lists of standing alarms start to build up during normal operations. Operators are therefore becoming "numb" to alarms, and cannot easily identify the real important alarms. This can cause serious problems, such as abnormal shutdown and even accidents. One of the reasons for alarm floods is improper alarm limits setting. When the process is operating under different conditions from the ones for which the initial alarm limits are set, the process measurements will be out of range and trigger alarms. Jensen (1997) suggested that the alarm configuration should switch dynamically according to the current operating state to avoid alarm floods. Although traditional DCS do not generally allow the selective application of alarm configurations for different operating states, but they do offer opportunities to manage alarm configuration through application programs. A process monitoring tool is necessary to switch configuration along with the operation state. Moore (1997) indicates that this dynamic alarm configuration strategy can be realized by monitoring the process operating state in either a manual or automatic way. The former is obviously impractical due to the complexity of large-scale processes. Arnold (1989) suggested establishing a logic structure for dynamic configuration. The alarm system will disable the unnecessary alarm setting dynamically based on the process operating state. Such advanced strategies for alarm management will need to identify the current operating condition accurately.

Fault detection and diagnosis is another example of the operating state based applications. While existing techniques for fault detection have largely focused on steady-state operations and are not directly applicable during transitions, Anshuman *et al.* (2003) proposed a novel model-based fault detection scheme that explicitly caters to the non-steady states and wide operating condition changes during transitions. The proposed approach is based on dividing a process into different phases. Different process models are employed for fault detection and diagnosis based on the current operating condition.

2.1 Data Clustering

Automated clustering techniques can be broadly categorized into static and dynamic clustering techniques. Given *nn* observations X_{nn}^{d}, static clustering techniques such as k-means and c-means clustering partition them into *nm* clusters, *[C₁, C₂, ... , C*$_{nm}$*]* with $1 \le nm \le nn$, each centered at $X_{cen_i}^{d}$ with $1 \le i \le nm$. The objective of the clustering is to find the centers to minimize a given cost function. Sebzalli and Wang (2001) proposed a two-step strategy to apply the c-means fuzzy clustering method to industrial process data. In the first step, Principal Component Analysis (PCA) is applied to reduce the dimensionality of the input. In the second step, fuzzy c-means clustering is used to locate the optimal centers. The authors concluded that the results from c-means clustering are comparable to the ones from manual examination of two-dimensional principal component plots. Zullo (1996) also reported a similar conclusion. One shortcoming of these clustering methods is that the number of clusters has to be specified *a priori*. Eltoft and de Figueiredo (2001) proposed a neural network-based clustering algorithm that overcomes this. In their approach, clustering starts with a single hidden layer neuron and a new neuron is added to the hidden layer every time the Euclidean distance between the input vector and existing neurons exceeds a predefined threshold. However, in the presence of process noise and disturbances, this method may result in unnecessary clusters arising from a few outliers in the data. In addition, in all these methods, temporal information is lost since only the relative position between feature vectors and centers is taken into consideration. These methods are therefore inapplicable for process states which are characterized by the temporal evolution of the process variables.

Dynamic clustering methods segment the time series data by investigating the underlying temporal relationships among the process variables. Consider an autoregressive process where the variable value x_t at time t can be approximated by a linear function $\mathrm{f}: \mathrm{x}_t = \mathrm{a}_1\mathrm{x}_{t-1} + \mathrm{a}_2\mathrm{x}_{t-2} + \ldots + \mathrm{a}_1\mathrm{x}_{t-1}$. It is assumed that the underlying function *f* governing the process in one cluster is uniform but is different from that in another cluster (Gupta, *et al.*, 2000). Klaus *et al.* (1996) proposed a neural network system consisting of *q* single networks, and $q>m$, where *m* is the estimated number of clusters. The system is trained so that each network approximates the underlying regression function *f* of a single cluster. After training, clustering of a new feature vector is achieved through the comparison of *q* prediction errors from the *q* networks. However, this method suffers from an inadequateness to work well in the face of process noises also it is not suitable to multivariate process monitoring.

A typical chemical process can be operated in a set of modes connected by transitions. It is then possible to cluster the multivariate process data by identifying steady state operating regimes. These segments are grouped into different clusters based on the similarity degree between any two modes or transitions.

Several methods for steady state identification have been proposed in recent years, a review can be found in the paper of steady state identification by Cao and Rhinehart (1995). An intuitive approach for identifying steady states in a uni-variate process is to estimate the variable's mean in a moving data window. If the estimated mean in the data window $\hat{\mu}(t)$ at time t deviates significantly from the one at the previous time $\hat{\mu}(t-1)$, i.e., $\left|\hat{\mu}(t)-\hat{\mu}(t-1)\right|>\theta$, where θ is a user-defined threshold, the process is said to be in a non-steady state. However, this method will lead to incorrect results in presence of sudden disturbances. In addition, the average value has to be calculated at every time instant, which is computationally expensive. A related approach calculates standard deviation of the process variable data over a moving window. The process is considered to move out of a steady state whenever the standard deviation exceeds a threshold. The threshold is normally determined based on steady state historical data. This method is also computationally expensive.

An alternate statistical approach is the use of the t-test (Lawrence, 1970; John, 1990). A t-test is carried out on the slope of a linear model built using a window of data. If the slope is found to be deviating from zero with a high confidence factor, the process is said to be in a non-steady state. Another approach based on the F-test was proposed by Cao and Rhinehart (1995). Here, the variance of the data in the most recent window is calculated by using two different methods. The ratio R of the two variances is used to identify steady state. The computational load is reduced in this method by calculating the variance using a regression approach. Jiang *et al.* (2003) proposed a wavelet-based method for on-line steady state detection. Sundarraman *et al.* (2003) presented a trend analysis-based approach to segment modes and transitions. A wavelet-based trend identification approach is used to identify quasi-steady and transition in a process. The temporal evolution of each variable is decomposed into a set of sequenced trends which are also known as primitives and examined to identify successive quasi-steady states. A segment of multivariate process is considered to be in steady state only when all the variables are in steady state during this period. All above methods are uni-variate. For multivariate case, each variable has to be analyzed separately and the results of the individual analysis are combined using a variety of rules (Brown, 2000). In this thesis, a PCA-based multivariate steady state identification technique is proposed.

The similarity degree between two steady states can be defined based on the means of two states. Two modes are defined to be instances of the same canonical mode if all their constituent variables overlap substantially. However, the comparison of two transitions is more complex. Given two time sequence S $(s_1, s_2, \ldots, s_{ns})$ and T $(t_1,$

$t_2, \ldots, t_{nt})$, with *ns* and *nt* number of observations respectively, the degree of similarity is usually based on estimating the "distance" between the two. The difference among the various approaches is largely related to the definition of the "distance" metric. One popular approach for time series comparison is Dynamic Time-Warping (DTW) (Kassidas, *et al.*, 1998). DTW shifts two sets of data in parallel until the best match is found. This method has been widely used in speech recognition and signal processing. Kassidas *et al.* (1998) reported the application of DTW for synchronizing batch trajectories. However, DTW is directly applicable only to one-dimensional signals. When applied to multivariate industrial processes, each variable has to be analyzed separately. Two temporal series can also be compared using the sequence of trends (Sundarraman, et al., 2003). However, like DTW, this method also analyzes only one-dimensional signals.

Another approach to sequence comparison is based on PCA. PCA is a commonly used dimensionality reduction technique (Jolliffe, 1986). It can transform the measurement data through a set of linear combinations. Thus, the process measurements can be reduced to a smaller informative set. Krzanowski (1982) defined a PCA similarity factor SPCA for estimating the degree of similarity between two data sets. Consider two temporal data sets *S* and *T* that have the same dimensionality, *d*. PCA transformation is carried out on both data sets to generate *k* PCs. If the corresponding $d \times k$ matrix of weights are denoted by H and O respectively, the S_{PCA} is defined based on H and O as:

$$S_{PCA} = \frac{trace(H^T O O^T H)}{k} \qquad [2\text{-}1]$$

It can also be written as the average of the cosines of the angles between pairs of principal components in H and O as:

$$S_{PCA} = \frac{1}{k}\sum_{i=1}^{k}\sum_{j=1}^{k}\cos^2\theta_{ij} \qquad [2\text{-}2]$$

Equation [2-2] can be understood as a comparison of the trend of the first *k* PCs of the two sets of data. Singhal and Seborg (2001) used the modified PCA similarity factor S_{PCA}^{λ} instead of Equation [2-2] to account for the variance.

$$S_{PCA}^{\lambda} = \frac{\sum_{i=1}^{k}\sum_{j=1}^{k}\lambda_i^S\lambda_j^T\cos^2\theta_{ij}}{\sum_{i=1}^{k}\lambda_i^S\lambda_i^T} \qquad [2\text{-}3]$$

Here, λ_i^S and λ_j^T are the i^{th} and j^{th} eigenvalues of the S & T time series respectively. Singhal and Seborg (2001) compared the PCA similarity factor with other methods such as T^2 statistic and *Q* statistic and concluded that PCA similarity factor results in a more accurate comparison.

One problem with traditional PCA is that it implicitly assumes that the measured variables are independent of each other across the time series (Chen and Liu, 2002). However, this situation is only possible when sampling interval is long enough. To reflect the dynamics of the process, Ku *et al.* (1995) proposed dynamic principal component analysis (DPCA). DPCA shows better modeling ability than static PCA as it considers not only the relationship across different variables but also that of the same variable across time (Chen *et al.,* 2001). Therefore, a DPCA based similarity factor is proposed in this thesis to overcome the problem of traditional PCA.

2.2 Temporal Pattern Recognition

A supervised classifier can be developed for operating state identification based on historical data. The construction of the supervised classifier becomes possible for industrial processes because: (1) computer-based process control systems measure thousands of process variables, (2) the process continues to operate in a series of states without drastic run to run changes for long periods, and (3) historical databases with several months or years of operations data are becoming common. Since the same process states repeat in different runs and display single patterns with small deviations, the expectation that there are good quality historical data for all operating states is justified and is the basis for the current work. Once the pattern of a state has been learnt, it can be used for future state identification. Therefore, the problem of constructing a supervised classifier is that of extracting and storing historical information such that relevant patterns can be retrieved and compared easily on-line.

Data classification or pattern recognition methods can be categorized into three classes (Schalkoff, 1992): statistical pattern recognition, syntactic pattern recognition and machine learning. The basis of the statistical method is the Bayes rule. Given an input feature vector X_{new}^{nd} whose class is unknown, the probability $p_i(C_i \mid X_{new}^{nd})$ that X_{new}^{nd} belongs to i^{th} class C_i can be estimated. The vector X_{new}^{nd} will be labeled with class C_i if $P_i < P_j, \forall i \neq j$, where $i, j \in nm$ and nm is total number of classes. The construction of a Bayes classifier is to find out a set of discriminant functions to calculate the posterior probability $p_i(C_i \mid X_{new}^{nd})$.

In the syntactic approach, a complex pattern is first decomposed to many simple patterns referred as primitives. Then, a structural language is used to describe the relationships among these sub-patterns. Finally, two patterns are compared by “string matching” or “parsing”.

Support vector machines (SVMs) and neural networks are two typical examples of machine learning. A SVM projects the original input vector to a high dimensional space to make the problem linearly separable (Schalkoff, 1992). Then support vectors, which maximize the margin between separating hyperplane and patterns are found. Artificial neural networks (ANNs) simulate the working mechanism of the human brain. Neural networks have been widely used for pattern recognition due to their powerful ability to approximate complex nonlinear functions. Hecht *et al.* (1988, 1989) indicated that a multilayer neural network with a sigmoid activation neuron can approximate arbitrary nonlinear functions with any desired level of accuracy. Later, Hornik *et al.* (1989) confirmed this conclusion by proving that an arbitrary nondecreasing activation function can approximate a continuous mapping $\phi:\ R^n \leftarrow [-X, X]^m$ with any small error *(e)*. Furthermore, Kreinovich (1991) gave a more general result: Assume *h(x)* is an arbitrary smooth function $R \to R$, X and e are positive real numbers, and ϕ is a continuous mapping from $[-X, X]^m$ to R^n. Then there exists a neural network that can approximate the mapping under the error *e*. Because of this ability, the applications of neural networks cover a wide variety of real world problems: such as chemical process related pattern recognition problems (Bulsari, 1995; Baughman and Liu, 1995; Muthuswamy and Srinivasan, 2003), speech recognition (Bengio, 1993; Kim *et al.* 1993; Levin *et al.* 1993), image processing (Li and Wang, 1993; Li and Nasrabadi, 1993), signature verification (Bromley *et al.,* 1993; Burges *et al.,* 1993; Drucker *et al.*, 1993) and industrial process identification (Chen *et al.,* 1999; Tsai *et al.,* 1996; Wang *et al.,* 1999).

Many approaches have been developed for temporal pattern recognition since it is very common in industrial processes. The main problem involved is how to store time information in the neural network. One approach is the use of past information explicitly as in the Time Delayed Neural Network (TDNN) (Bambang *et al.*, 2001; Martin, 2001; Wohler and Anlauf, 2001,). In TDNN, the information in the recent past is stored in a buffer and presented to the network along with the current inputs. The method can be represented as a mapping: $F:\ Z_t \leftarrow X_t^{nd} : [X_t^d, X_{t-1}^d, \cdots, X_{t-l}^d]$, where $nd = d \times (l+1)$. By converting the time domain information into space domain, the TDNN makes use of simple static neural networks to model dynamic processes (Figure

2-1). The system regressive order, *l*, has to be estimated before TDNN can be utilized. In addition, the applicable *l* is limited by the size of the neural network input layer and hardware computational limit.

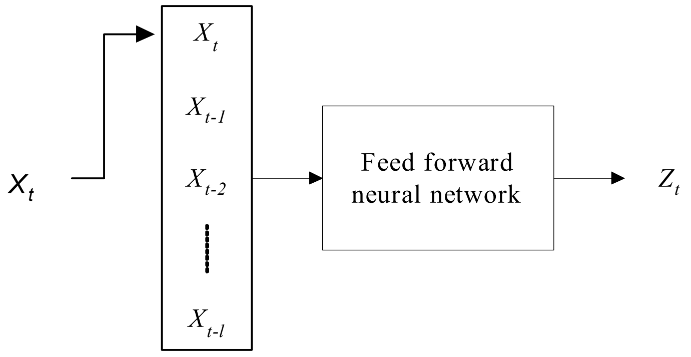

Figure 2-1: Time delay neural network

Past information can be stored in a more implicit manner in recurrent neural network structures such as the Elman network which was first proposed by Elman (1990). The output of the Elman network hidden layer is fedback to itself so that the dynamics of process are captured (Figure 2-2). Theoretically, the first input will affect all the following network outputs and the network therefore gain the ability to process temporal signal. However, this is only ideal situation. In fact, our experiments with Elman neural networks prove that the information is retained in the network for around 20 time steps before being washed out.

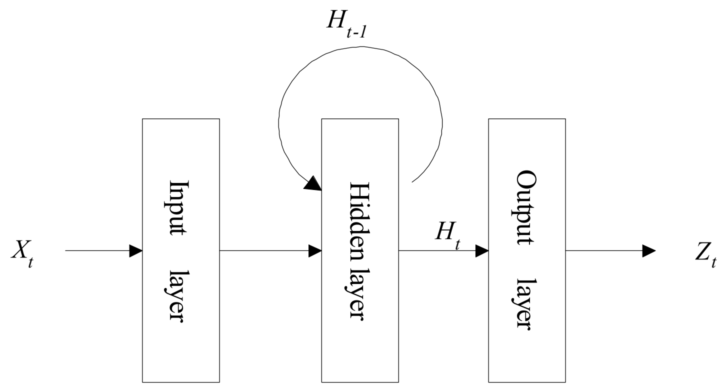

Figure 2-2: Elman neural network

A transform of the input matrix is another method to capture process dynamics. A novel transform proposed by Stiles and Ghosh (1997) is based on the phenomenon called habituation. Primarily, habituation is a means by which biological neurons can filter out repetitive and hence irrelevant information. Neurons achieve this by adjusting their synaptic strength (the counterpart in artificial network is the "connection weights"). If the presynaptic neuron is active for a period of time, habituation tends to reduce the synaptic strength and recovers it only after the activity is over. When the concept is applied to input encoding, it turns out to be an input weights calculation method. The essential idea of habituation transform is to use a set of weights instead of original input X_t^{nd} . After the encoding which is based on time, the input X_t^{nd} is converted to $[W_t, W_{t-1}, \cdots W_{t-l}]$ (Figure 2-3). A discrete time version of the habituation model was first presented by Wang and Arbib (1990) in the following form:

$$\begin{aligned} W_{t+1} &= W_t + \tau(\alpha Z_t W_0 - W_t) - W_t I_t \\ Z_{t+1} &= Z_t + \gamma Z_t (Z_{t-1} - 1) I_t \end{aligned} \qquad [2\text{-}4]$$

where I_t is the output of the presynaptic neuron, τ and α are constants used to vary the habituation and recovery rate and Z_t is a monotonically decreasing function. In the case of multi-dimensional input, encoding each variable in the above manner can give the transferred input matrix. The W_t will decease to zero eventually after a period of time that is determined by τ , α and γ .

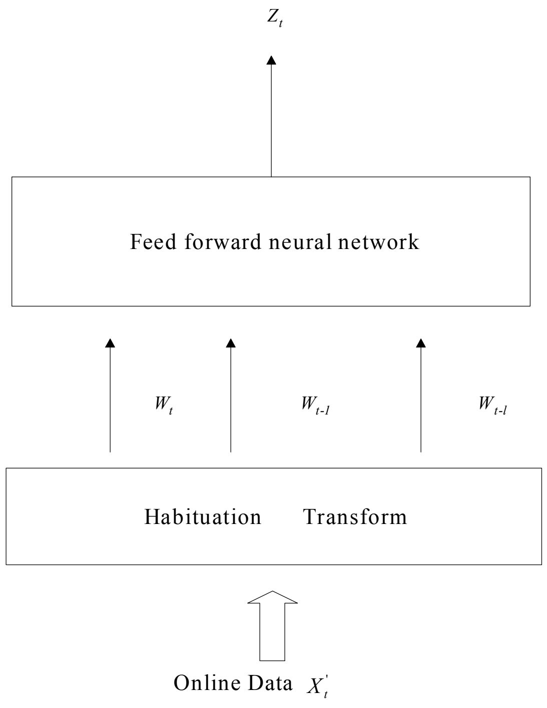

Figure 2-3: Habituation neural network

In the past decade, the discovery that most biological neural systems use spikes to encode the time-related input information has led to the development of the Spiking Neural Network (Wolfgang, 1997). A "spiking neuron" has an excitatory or inhibitory activation function (Jan *et al.,* 2001) and a time variant bias function (Figure 2-4). As a result, temporal information is transferred to a series of "spikes" at the output layer at irregular time intervals. The "spiking neural network" is expected to be capable of learning very large time sequences.

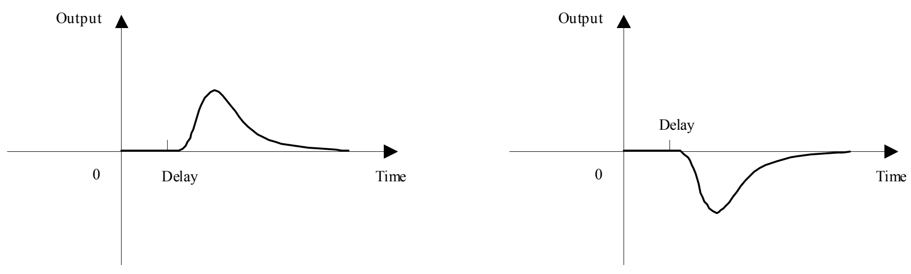

Figure 2-4: Activation functions of spiking neuron (a) Excitatory function (b) Inhibitory function

When a neural network is constructed, its structure is decided not only by the input dimensionality but also by the complexity of underlying classes. When both the input dimensionality and the number of classes are high, as is the situation in industrial temporal signals, the training of network becomes a challenge task. A typical chemical process unit usually has hundreds of sensors each generating thousands of observations every day. These data are noisy and contains patterns corresponding to the different operating states. Although the neural network computes in a parallel manner, its training is a major challenge especially when a single neural network is used. The training takes a considerable computation time even then the resultant network may perform poorly. In this thesis, two novel neural network structures which can identify the temporal patterns accurately with modest training requirement are proposed.

2.3 Context-based Pattern Recognition

Patterns whose interpretation varies across contexts are common in many engineering domains such as natural language processing, databases, communication, electronic documentation, and vision (Brezillon, 1999). Kettebekov and Sharma (1999) discussed the interpretation of a human gesture. The same gesture (primary feature) can imply a number of meanings in different occasions (contexts). Using simultaneous speech information as contextual features is essential to avoid misunderstanding of a gesture. Another example is language interpretation ─ the same word (primary feature) could have very different meanings in different situations (Hunter, 2001; Maskery and Meads, 1992). While these problems are extreme examples where the recognition of the objective is impossible without context information, in other cases, context information can improve the recognition performance. Turney and Halasz (1993) used contextually normalized features to improve aircraft gas turbine engine diagnosis. In this domain, ambient temperature is the context and affects the feature patterns arising from a fault. In their approach, the context effect was first removed from the primary features before pattern recognition was performed. Turney (1993) also presents a survey of a wide range of research problems where context information can improve machine learning performance.

Turney (1993a, 1993b) introduced the general definition of context-sensitive features in any pattern recognition problem. The context-based pattern recognition problem is defined by differentiating between three types of types of features: *primary, contextual, and irrelevant*. Primary features are defined as those features x_i that are useful for classification when considered in isolation, without regard to other features. Contextual feature is not useful in isolation, but can be useful when combined with other (primary) features. Irrelevant features are not useful for classification, either in isolation or when combined with other features.

Once features have been segregated into primary, contextual, and irrelevant ones, a mechanism to incorporate contextual information into pattern recognition has to be formulated. Several strategies have been proposed for this (Katz, *et al.*, 1990; Turney, 1993a; Turney, 1993b). In contextual normalization**,** contextual features are used to normalize the context-sensitive primary features, prior to classification. The intent is to process context-sensitive features in a way that reduces their sensitivity to context. For example, a feature can be normalized so that its ranges in different contexts do not overlap. A related approach to contextual normalization is contextual weighting. There, the contextual features are used to weight the primary features, prior to classification. The intent of weighting is to assign more importance to features that, in a given context, are more useful for classification. An alternative approach is to expand the feature space by including the contextual features as well as the primary features. This is called contextual expansion. The contextual features are handled by the classifier in the same manner as other features. In contextual classifier selection, classification proceeds in two steps. In the first step, different classifiers are trained, each specializing in a particular context. Subsequently, the contextual features are used to select the right classifier for a given situation (context). During contextual classification adjustment, the two steps in contextual classifier selection are reversed. In the first step, all the classifiers are used to predict the class using the primary features. The final identification is through a composition based on the contextual features.

While context-based pattern recognition problem is well-defined and some frameworks are proposed, few structures have been presented so far. In this thesis, a novel structure is presented to solve this problem.

Chapter 3. Dynamic PCA Based Methodology for Clustering Process

3.1 Introduction

An agile chemical plant normally operates in a number of operating states. Plant startup, grade change and shutdown are some common examples. The holy grail of the control community has been to enable optimal control of the process during different operating states. One crucial task towards this is the online identification of a process' operating state. Different control configurations or controller parameters may then be used (Rosen and Yuan, 2001). Model identification, fault detection, and alarm management (Arnold and Darius, 1989) are other applications whose parameters have to be adjusted to fit the current process state. Therefore, the ability to identify the state of operation from the time evolution of the sensor data is essential for process automation in agile chemical plants (Roverso, 2000).

A process unit's state can be classified into modes and transitions (Srinivasan *et al.*, 2001). A mode corresponds to the continuous operation of the unit and a fixed flowsheet configuration, i.e. no equipment is brought online or taken offline. During a mode, the process unit operates in a quasi-steady state. Thus, its constituent variables are at steady states and their values vary within a narrow range. The concept of steady state from the process operation point of view is different from statistical "stationarity" (Cao and Rhinehart, 1995). Statistical stationarity requires that the statistical properties do not change with time whereas steady state only requires that the slope of every key variable *X(t)* is small (Jiang *et al.*, 2003):

$$\left|\frac{X(t)-X(t_0)}{t-t_0}\right| < T_f \quad \forall t \in [t_0-\Delta t, t_0+\Delta t] \qquad [3\text{-}1]$$

where T_f is user-defined threshold. An example is used to illustrate the steady state. Two variables of a typical chemical process are plotted vs. time (10 second sampling rate) in Figure 3-1. From the figure, three modes can be distinguished: M_0 between t=1x10s and t=80x10s, M_1 between t=702x10s and t=1315x10s and M_2 between t=1440x10s and t=1580x10s.

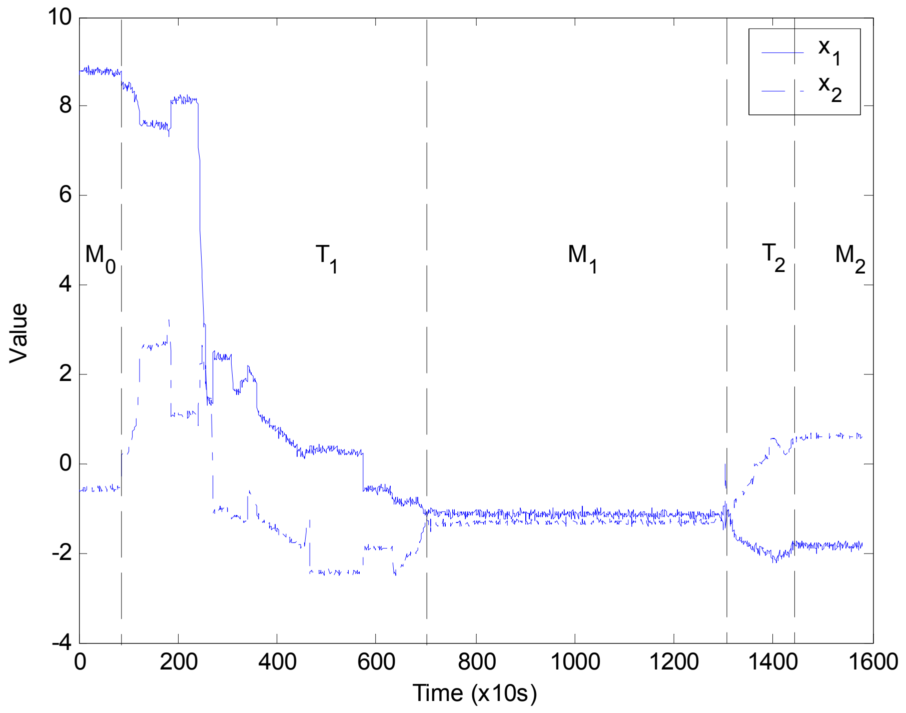

Figure 3-1: Evolution of two variables of a typical chemical process

Transitions correspond to discontinuities in the plant operation such as change of set-point, opening of valve, change of equipment configuration, turning on or idling equipment, etc. These operational events are usually induced by operator action. When a unit undergoes a transition, at least one of its constituent variables would show a significant change. However, all constituent variables do not have to vary during a transition. The time evolutions of the key variables contain distinct signatures that can be used to identify a transition and differentiate it from other transitions in the unit. A transition is characterized by the key variables along with the distinct features in their evolutions, the preceding mode, the subsequent mode, and the start and end times of the transition. Two transitions are shown in the example in Figure 3-1: T_1 between t=81x10s and t=701x10s and T_2 between t=1316x10s and t=1439x10s. As can be seen, the time evolutions of X_1 and X_2 are different in the two transitions. Also, T_1 begins from mode M_0 where X_1 has a value of 9 and X_2 of -0.5 and ends at mode M_1 where X_1 has a value of -1.2 and X_2 of -1.2. In contrast, transition T_2, that begins from mode M_1 and ends at mode M_2 where X_1 has a value of -1.8 and X_2 of 0.5.

As mentioned in the Chapter 1, clustering of process states in historical data can be used for a number of purposes, such as comparing operating condition to improve the understanding of transitions, correlating features of successful runs to product properties, process efficiency, etc.

Some existing automated clustering techniques have been reviewed in Chapter 2. While all the methods mentioned consider the entire process data monolithically, we propose a method which is based on differentiating between the states—modes and transitions in the process. It is possible to segment the multivariate process data by identifying steady state operating regimes. These steady states can therefore be used to segment the data into different operating modes and transitions. The operating states are then grouped into different clusters based on the similarity between them. If the similarity degree between two modes or two transitions is sufficiently large, they will be concluded as belonging to the same cluster. Therefore the proposed method includes two sub-problems: (1) Steady state identification, and (2) Similarity comparison. Existing methods are reviewed next.

The rest of this Chapter is organized as follows. In Section 3.2, the proposed DPCA-based process state clustering method is presented. Next, in Section 3.3 and Section 3.4, the application of the proposed method is illustrated on two simulated case studies – a fluidized catalytic cracking unit and the Tennessee Eastman challenge problem.

3.2 Proposed Method for Clustering Process States

The proposed process state clustering strategy is summarized in Figure 3-2. Historical data are first analyzed to identify periods of steady states using the multivariate method that that will be presented in Section 3.2.1. These steady states are used to segment the data into different operating modes and transitions. The operating states are then clustered using similarity measures described in Section 3.2.2.

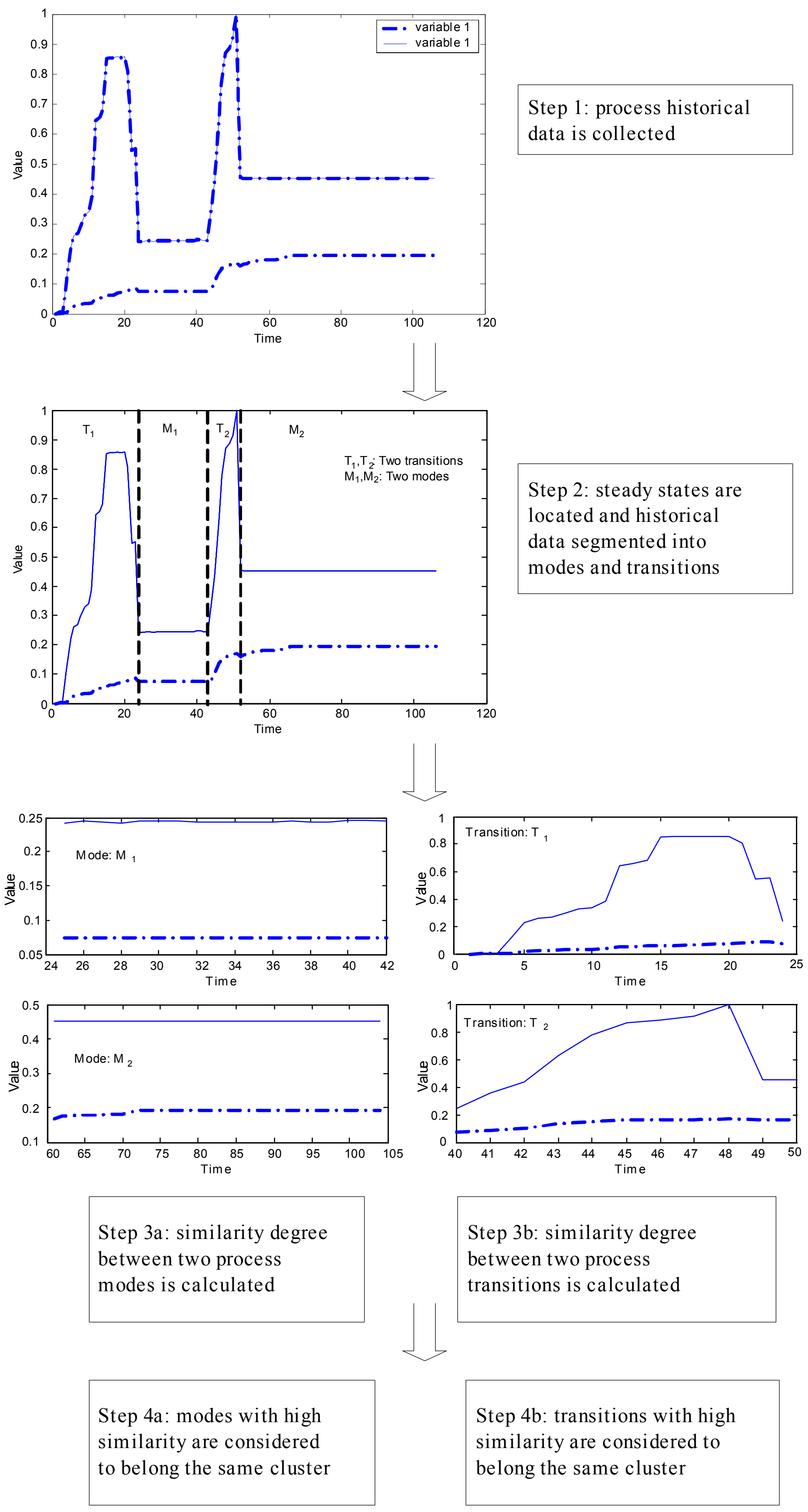
variable 1
variable 1
Value
Time
Step 1: process historical data is collected
T1
M1
T2
M2
T1,T2: Two transitions
M1,M2: Two modes
Step 2: steady states are located and historical data segmented into modes and transitions
Mode: M 1
Transition: T 1
Mode: M 2
Transition: T 2
Step 3a: similarity degree between two process modes is calculated
Step 3b: similarity degree between two process transitions is calculated
Step 4a: modes with high similarity are considered to belong the same cluster
Step 4b: transitions with high similarity are considered to belong the same cluster

Figure 3-2: Proposed process state clustering approach

3.2.1 Identification of Steady States

As we have discussed in Chapter 1, a mode is characterized by near constant values for all process variables in a given time window. However, a chemical plant has hundreds of variables. Considering all the variables one-by-one is computationally expensive and makes combining results from individual variable analysis difficult. To overcome these and also account for the redundancy of sensors, PCA is used here to reduce the dimensionality of the data.

An operating state can be treated as a mode if all variables fluctuate within a limited range. Since the obtained scores from the PCA operation are a linear combination of the original variable values, during a mode, the variation of the scores will also be small. It is illustrated by the score plot for the example in Figure 3-1, which is shown in Figure 3-3. Samples from M_0, M_1 and M_2 are concentrated in small ellipses. In contrast, the samples from T_1, T_2 are scattered over a larger area. We exploit this property to segregate steady states from transitions.

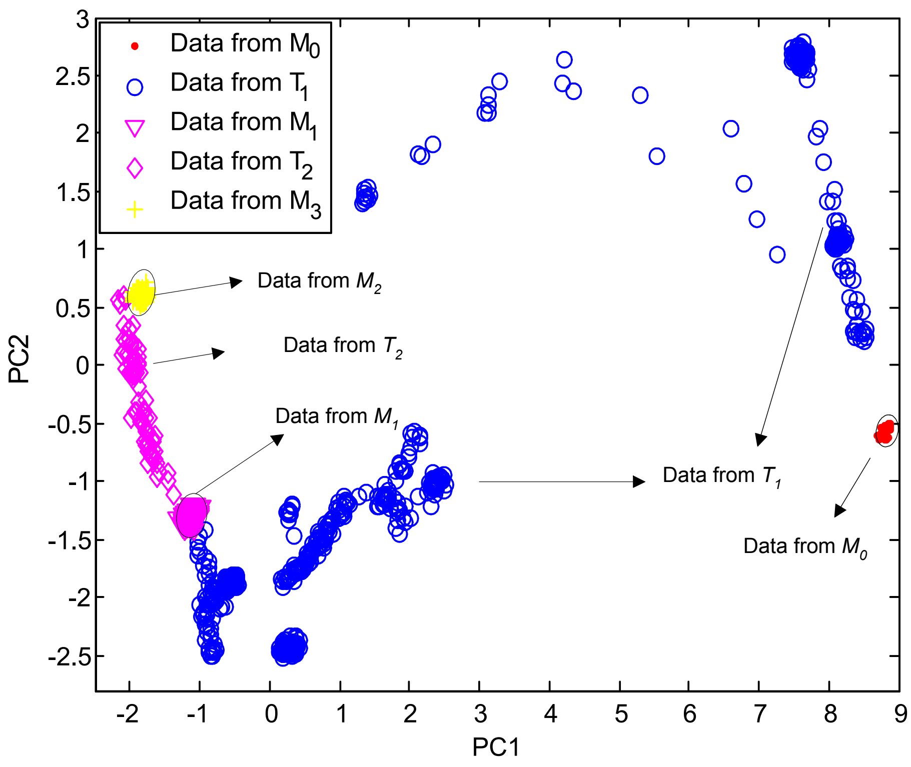

Figure 3-3: Score plot of modes M_0, M_1, M_2 and transitions T_1, T_2

The proposed steady state identification methodology is summarized in Figure 3-4. The historical data X_{nn}^d *:[X₁,,X₂,, …, X_{nn},]* are auto-scaled and PCA performed. The first *k* principal components are retained for the next step. The selection of *k* is based on a plot of P_i vs. *i,*

$$P_i = \frac{\lambda_i}{\sum_{i=1}^{k} \lambda_i} \qquad [3\text{-}2]$$

where λ_i is i^{th} eigenvalue of the covariance matrix of X_{nn}^d. *k* is selected as a cutoff point on a sharp change in the slope of the graph. An alternative is to retain *k* principals such that:

$$\sum_{i=1}^{k} P_i > 93\% \qquad [3\text{-}3]$$

Here, the 93% threshold is a commonly used cutoff and can be replaced by another based on domain knowledge and the expected noise levels (Sebzalli and Wang, 2001).

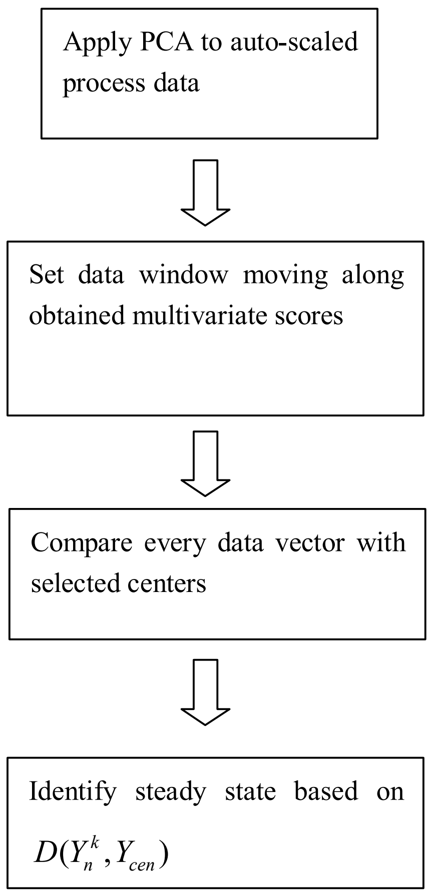
Apply PCA to auto-scaled process data
Set data window moving along obtained multivariate scores
Compare every data vector with selected centers
Identify steady state based on $D(Y_n^k, Y_{cen})$

Figure 3-4: Proposed steady state identification approach

The resulting k x nn dimensional scores Y^k is then analyzed to identify steady states. Steady states are identified by looking at a data window of length T_w along the score matrix. Each k-dimensional vector Y_i^k within the window is compared with randomly chosen centers Y_{cen}^k (described below) and the distance D between Y_i^k and Y_{cen}^k calculated:

$$D(Y_i^k, Y_{cen}^k) = | Y_i^k - Y_{cen}^k | \qquad [3\text{-}4]$$

Here, $D(Y_i^k, Y_{cen}^k)$ is defined as a k-dimensional vector so that each variable can be compared separately. Every Y_i^k is then checked for the following condition:

$$D(Y_i^k, Y_{cen}^k) < \theta_d.(Y_{\max}^k - Y_{\min}^k) \qquad [3\text{-}5]$$

where θ_d is a pre-specified "variation tolerance", $Y_{\max}^k$ and $Y_{\min}^k$ are the vector of the maximum and minimum values of Y^k over the whole process respectively. The "<" in Equation [3-5] is defined as an element-wise comparison requiring that every element satisfies the condition. If at least δ fraction of Y_i^k in a window satisfy Equation [3-5], then the process is concluded to be within a mode during the window. The data window is then moved forward by step size L and the process repeated for the entire Y^k. All steady states in the historical data are thus identified. It should be noted that a process may change gradually and process variables would have a gradient close to zero. In such cases, when the variables are changing within the "variation tolerance", the proposed method will correctly identify such states as modes.

The Y_{cen}^k plays a critical role during steady state identification. In our approach, several data points from within the window are used as centers. This is because if a single Y_{cen}^k is used and an outlier or disturbance point is selected for this, then the identification would be susceptible to misclassification. For instance, Figure 3-5 shows data from steady state operation with a few outliers probably from noise or process disturbance. If one of these disturbance points is selected as Y_{cen}^k, the distance $D(Y_i^k, Y_{cen}^k)$ of most observations will exceed the variation tolerance and the current window will be inaccurately identified as being in a non-steady state. To avoid this without using computationally heavy outlier detection, we use several randomly selected points from within the data window as centers. If the criterion in Equation [3-5] is satisfied by even one Y_{cen}^k, the window is concluded to be in a steady state. Only when none of the centers satisfy the criteria, will the process be concluded as being in a transition state. The reader would note that, through this, outliers would not be mistaken as transitions.

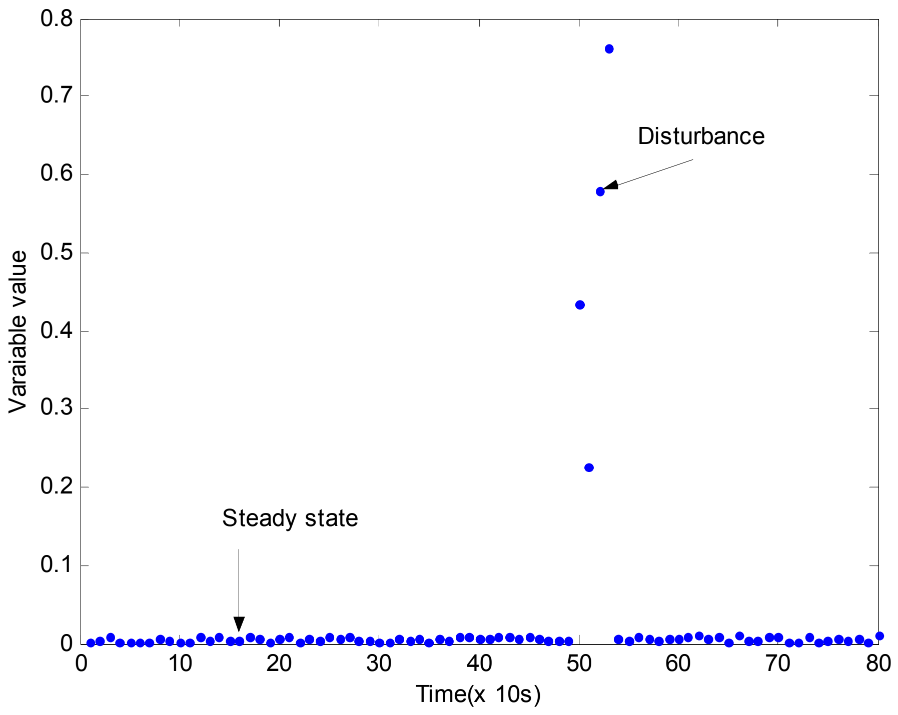

Figure 3-5: A disturbance during steady state operation

A steady state is considered to be a mode only if it lasts for *at least* a certain period. This minimum duration is called the *dwell time* and denoted as TS_{min}. It is needed because even during a transition, the process variable evolution could contain short periods with nearly constant values. The dwell time constraint prevents mis-identification of these short constant periods as modes. A valid mode can thus last for any duration longer than TS_{min}. A proper setting of TS_{min} is important to get correct segmentation of the process data. It is usually possible to specify TS_{min} from process knowledge and historical data.

Edge Detection: When the process is detected to move from a mode to a transition or vice versa, a further check is performed to accurately detect the edge. While the technique is directly applicable to multivariate processes. For the convenience of illustration, we use a uni-variate process. Figure 3-6 shows a part of a mode and a transition. By the method discussed above, the data in the window

$[\alpha - L \;\; \alpha - L + T_w]$ is identified as a mode, and data within the window $[\alpha \;\; \alpha + T_w]$ is a transition. Now for the window $[\alpha \;\; \alpha + T_w]$, the beginning of the transition — that is, change point ***A*** — must be located. To accurately detect the position of ***A***, data window $[\alpha \;\; \alpha + T_w]$ is sub-divided into *r* sub-windows. Data in each sub-window of length T_w/r is compared with the mean of the data in the preceding window $[\alpha - L \;\; \alpha]$ using a condition analogous to Equation [3-8] to check if it is a steady state. If the "variation tolerance" is crossed by a part in the sub-window, that sub-window is deemed to be a transition and its beginning point is declared as the start point of the transition. In the example in Figure 3-6, $\alpha + \frac{2T_w}{r}$ is identified as the start point of the transition. Similar method is used to detect the change point when a process moves from a transition to a mode. The key advantage of this edge detection is that while the resolution of steady state identification is improved from T_w to T_w/r, the total computational burden does not increase much since the edge detection is executed only when a state changing is detected from a mode to a transition or vice versa. Therefore, *L* can be large to maintain a good computational performance without adversely affecting accuracy.

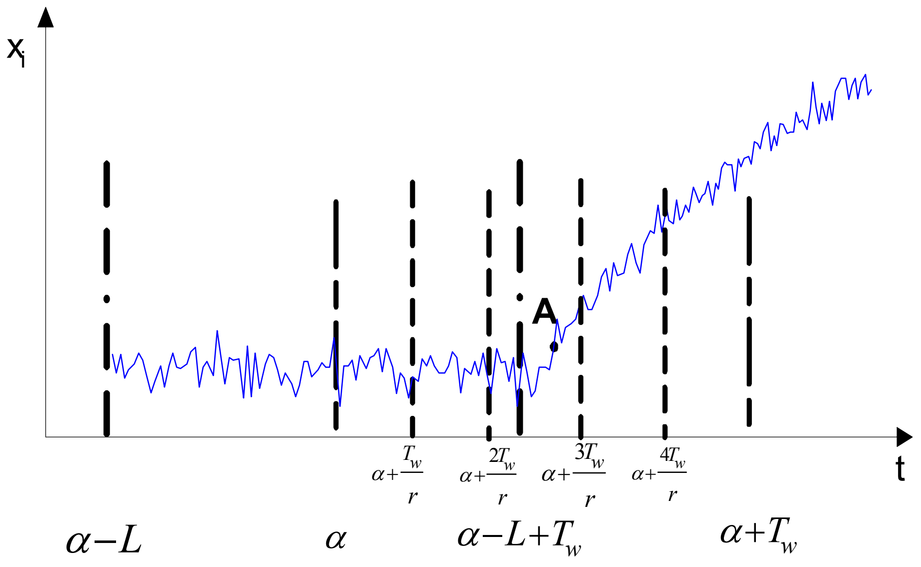

Figure 3-6: Mechanism for edge detection during steady state identification

3.2.2 Similarity Measurement

Once the modes have been identified and their beginning and end points located, the rest of the regions in Y^k are tagged as transitions. The whole historical data X_{nn}^d is also divided based on the segments into modes and transitions. To finish the clustering task, similar modes and transitions in X_{nn}^d have to be clustered together.

3.2.2.1 Similarity between Modes

As described earlier, modes are characterized by small fluctuations in the variable values. Hence, the mean vector is the principal property of a mode and the similarity degree between two modes is defined based on this property. The traditional method for comparing means is the t-test (Douglas and George, 1994). However this is a uni-variate statistic. In this thesis, I propose a simple multivariate method.

Normalization: In order to avoid the effect of different scales during the distance calculation, each variable x_i is first normalized to [0, 1] based on the measurement range of the sensor.

$$\tilde{X}_i = \frac{X_i - X_L}{X_H - X_L} \qquad [3\text{-}6]$$

where, $\tilde{X}_i$ is the normalized process variable, and X_H and X_L are the high and low limits of the sensor range respectively. The corresponding normalized historical data $\tilde{X}_{nn}^d$ is subsequently used for clustering modes.

Suppose M_i and M_j are two modes. Their process mean vectors $\hat{\mu}_{M_i}$ and $\hat{\mu}_{M_j}$ is estimated from $\tilde{X}_{nn}^d$. M_i and M_j are considered to be instances of two different modes if at least one of the process variables has distinctly different mean values in the two modes. Note that this is congruent to Equation [3-5] and in line with the definition of transition in Section 3.2.1. In general, the distance between M_i and M_j is defined as the maximum distance along the elements between the $\hat{\mu}_{M_i}$ and $\hat{\mu}_{M_j}$. That is:

$$D(M_i, M_j) = \max |\hat{\mu}_{M_i} - \hat{\mu}_{M_j}| \qquad [3\text{-}7]$$

Since x_i is normalized and therefore the elements of $\hat{\mu}_{M_i}$ and $\hat{\mu}_{M_j}$ are in [0 1], $D(M_i, M_j)$ is also in [0 1]. The similarity factor between two modes is then defined as:

$$S_M(M_i, M_j) = 1 - D(M_i, M_j) \qquad [3\text{-}8]$$

Following this, two modes M_i and M_j are classified into the same cluster if:

$$S_M(M_i, M_j) > \theta_M \qquad [3\text{-}9]$$

where θ_M is a user-defined threshold and can be estimated from historical data.

3.2.2.2 Similarity between Transitions

The comparison of transitions is based on the comparison of the corresponding variable profiles. We use a DPCA-based similarity factor for this purpose since it can account for the autocorrelation in the variables by augmenting the data matrix with time-lagged variables. Hence, if the current value of a variable *X(t)* is expressed as the weighted sum of past observations $X(t) = A_1.X(t) + A_2.X(t-1) + \cdots + A_l.X(t-l)$, where, A_l is the regression coefficients and l is the regression order of the process, the dynamics of the process and the time-dependent relationships between the variables is contained in the extended matrix, X_{delay} (Rosen and Yuan, 2001).

$$X_{delay}(t) = \begin{bmatrix} X^d(t) & X^d(t-1) & \cdots & X^d(t-nn+l) \\ X^d(t-1) & X^d(t-2) & & X^d(t-nn+l-1) \\ \vdots & \vdots & \ddots & \\ X^d(t-l) & X^d(t-l-1) & & X^d(t-nn) \end{bmatrix} \qquad [3\text{-}10]$$

where $X^d(t) = [x_1(t), x_2(t), \cdots, x_d(t)]^T$ is the *d*-dimensional process data vector at time *t*. The dimension of X_{delay} is $nh \times (nn - l + 1)$, where $nh = d \times (l+1)$.

DPCA consists of applying traditional PCA to the extended data matrix X_{delay}:

$$Y = U^T X_{delay} \qquad [3\text{-}11]$$

where *Y* is the projection of the X_{delay} on a new set of bases *U.* The transformation matrix U is composed of the eigenvectors of the covariance matrix of X_{delay}. Equation [3-11] can also be written as:

$$\begin{aligned} y_k &= a_{k_1} x_1(t) + \cdots + a_{k_d} x_d(t) \\ &\quad + a_{k_{d+1}} x_1(t-1) + a_{k_{d+d}} x_d(t-1) + \cdots + a_{k_{nh}} x_d(t-l) \end{aligned} \qquad [3\text{-}12]$$

where y_k is the value of k^{th} score or latent value, $x_d(t-l)$ is the value of variable X_d at time *(t-l)*, and $\{a_{k_1}, a_{k_2}, \cdots, a_{k_{nh}}\}$ is the value of k^{th} PC. The obtained k^{th} score y_k is now a linear summation of *nh* values, including not only the current process measurements but also the previous ones. Note that two kinds of dynamic relations are taken into consideration, (1) the relationships between the current value of X_d and its past values $[X_{d_{(t-1)}}, X_{d_{(t-2)}}, \cdots, X_{d_{(t-l)}}]$, and (2) the relationships between the value of variable X_d and the other variables and their values from time $t-l$ to *t*. The *nh* eigenvectors thus reflect the essential temporal correlation within a data window.

A DPCA similarity factor $S^{\lambda}{}_{DPCA}$ can now be defined analogous to the PCA similarity factor that been given in literature review. For reader's convenience, I present the equation here again.

$$S^{\lambda}_{PCA} = \frac{\sum_{i=1}^{k} \sum_{j=1}^{k} \lambda_i^S \lambda_j^T \cos^2 \theta_{ij}}{\sum_{i=1}^{k} \lambda_i^S \lambda_i^T} \qquad [3\text{-}13]$$

$S^{\lambda}{}_{DPCA}$ is defined on Equation [3-13] based on the *nh* vectors in U. The DPCA similarity factor compares the temporal profiles as brought out by the following example. Figure 3-7 shows two transitions *A* and *B* in a process with two variables: During transition A, variable x_1 is first ramped up and then ramped down to the original value. During transition B, x_1 is only ramped up. The range of x_1 is the same during both the transitions, while x_2 is constant except for noise. Transition A and B are two different dynamic responses of the process and should be identified as different temporal patterns. Without the use of autocorrelation information, transitions A and B are indistinguishable. The principal components from PCA for both transitions are exactly the same.

$$U_A = U_B = \begin{bmatrix} -1/\sqrt{2} & -1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

The PCA similarity factor S_{PCA}^{λ} between the two transitions is equal to one which implies that two transitions are identical. When the autocorrelation is considered using DPCA, for a lag l=8, the corresponding principal components are:

$$U_A = \begin{bmatrix} \text{-0.3954} & -0.1503 & 0.1179 & \cdots & 0.2246 \\ 0.4935 & 0.3425 & 0.0474 & \cdots & 0.0169 \\ 0.1257 & \text{-0.0348} & \text{-0.1857} & \cdots & \text{-0.4317} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0.0584 & 0.0630 & \text{-0.0329} & \cdots & 0.0070 \end{bmatrix}$$

and

$$U_B = \begin{bmatrix} \text{-0.3307} & \text{-0.0928} & 0.0118 & \cdots & \text{-0.4429} \\ 0.0586 & \text{-0.6375} & 0.0242 & \cdots & 0.0694 \\ \text{-0.3301} & \text{-0.0267} & 0.0308 & \cdots & 0.1577 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \text{-0.0081} & \text{-0.0177} & \text{-0.3269} & \cdots & \text{-0.0102} \end{bmatrix}$$

The 18-dimensional eigenvectors of the two transitions are no longer identical and S_{DPCA}^{λ} is 0.54. The two transitions are now distinguishable based on the difference in their autocorrelations. Another advantage of the S_{DPCA}^{λ} is that it is insensitive to the duration of the transition; only the underlying direction of change is considered. Therefore, the need to synchronize transitions before they can be compared is obviated. The multivariate relationships captured by the DPCA ensures that equipment failure or mal-operation during a transition will be reflected in the eigenvectors. This property can be used for detecting faults during the execution of a known transition. The selection of k (<< *nh*) also make this method less susceptible to noise and to other small variations in the variable evolution which would not recur in every instance of a transition.

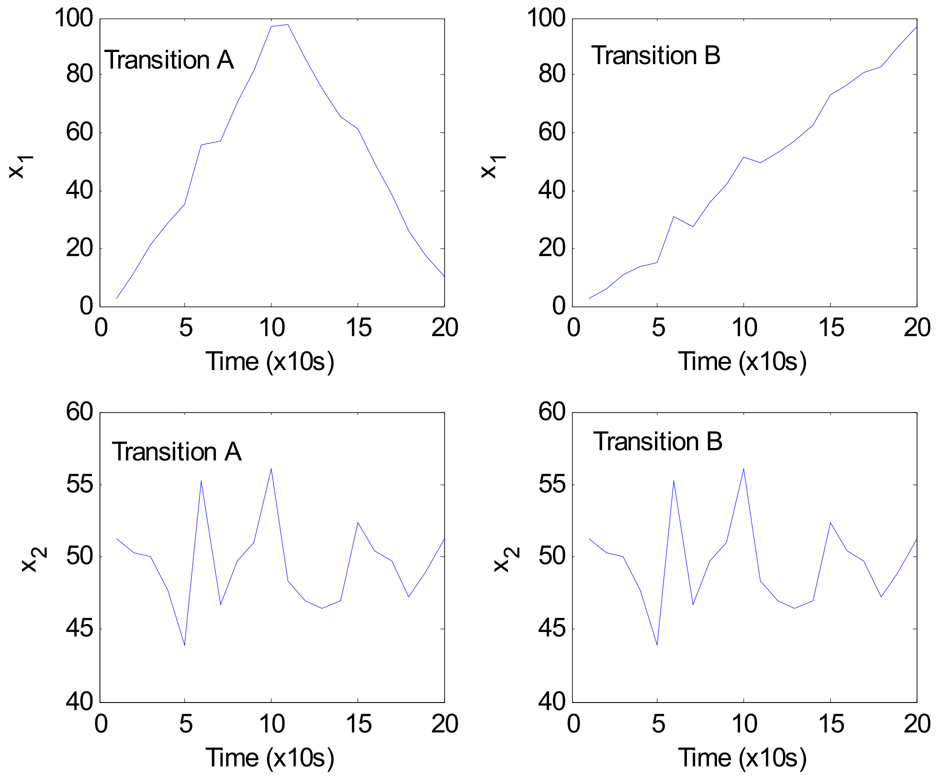

Figure 3-7: Transitions in a two-variable example

The proposed transition comparison method can be summarized as follows: Suppose T_i and T_j are two transitions. The data from T_i and T_j are auto-scaled to zero mean and unit variance first. The process autoregression order l is determined based on process characteristics and knowledge of time constants. Then DPCA operation is performed on the auto-scaled data and the first k principal components are retained. The DPCA similarity factor S_{DPCA}^{λ} is calculated based on the k PCs. The resulting similarity factor takes values between zero and one, where zero denotes no similarity between the two transitions and one indicates that the two are identical. Without considering the magnitude information, based on principal components along, T_i and T_j are classified into the same cluster if:

$$S^{\lambda}_{DPCA}(T_i, T_j) > \theta_T \qquad [3\text{-}14]$$

where, θ_T is a "trend-deviation" threshold. S^{λ}_{DPCA} is sufficient when only the profiles or the underlying dynamics are to be matched without regarding the magnitude. However, when a process state classification is required taking into account the magnitude of the transition as well, the S^{λ}_{DPCA} factor has to be complemented with a magnitude comparison. The mode transition segmentation provides a natural means to achieve this through the S_M factor discussed above. Since any transition T_i is sandwiched between two modes, say M_p at the beginning and M_q at the end, the $\hat{\mu}_{M_p}$ and $\hat{\mu}_{M_q}$ provide an adequate information for comparing two transitions' magnitudes. So when comparing T_i with a transition T_j that is sandwiched between M_g and M_h, $S_M(M_p,M_g)$ and $S_M(M_q,M_h)$ can be considered in addition to $S^{\lambda}_{DPCA}(T_i, T_j)$. In case when a comparison between both profile and magnitude is required, two transitions are similar if all following three criteria hold:

$$S^{\lambda}_{DPCA}(T_i, T_j) > \theta_T \qquad [3\text{-}15a]$$

$$S_M(M_p, M_g) > \theta_M \qquad [4\text{-}15b]$$

$$S_M(M_q, M_h) > \theta_M \qquad [4\text{-}15c]$$

The proposed process state clustering method involves several parameters — $TS_{\min}, T_w$, L, θ_d, l, θ_M and θ_T. While different processes may require different values for these parameters, we have found that most of the settings can be used across case studies (see Section 3.3 and 3.4). If necessary, fine-tuning can be performed based on the clustering results and process knowledge, some basic guidelines are given below:

- The dwell time TS_{min} can be easily specified by a process engineer based on the shortest steady state expected in the process.

- T_w and L can be specified based on TS_{min}. If T_w is too big, the modes will be merged into the transitions and become undetectable. On the other hand, if T_w is too small, short segments of transition would be misidentified as modes. It will also increase the computational burden. Similarly, too small L will aggravate the computational burden while a large one may miss short mode. We therefore use

$$T_w = TS_{\min},\ L = \frac{1}{2} TS_{\min} \tag{3-16}$$

- θ_d is selected based on the noise level in the process. If θ_d is too small, noise in the measurement would lead to the misclassification of modes as transitions. If θ_d is too large, large changes will be allowed within a mode therefore transitions will be merged into modes.

- The value of l is a measure of the nonlinearity expected in the process. Large values of l will be needed in highly nonlinear process while simple processes will require a small l. A large value of l increase the size of X_{delay} and hence the computational complexity.

- θ_M and θ_T can be estimated from historical data. If *a priori* knowledge reveals two instances of the same modes or transitions, then S_M and S_{DPCA}^{λ} from these states can be calculated and used as a lower bond. This is discussed in details in Section 3.3.

These procedures are illustrated based on the case studies in Section 3.3 and 3.4.

3.3 Fluidized Catalytic Cracking Case Study

The proposed two-step clustering strategy has been tested on the startup of a simulated Fluidized Catalytic Cracking Unit (FCCU). FCCU converts a mixture of heavy oils into more valuable light products. The control of the FCCU, especially transition control, has attracted extensive attention recently because of its complexity (Khalilian and Dhib, 2001; Pramit and Raghunathan, 2000). A high-fidelity dynamic simulator of a FCCU, called ShadowPlant, is used for this purpose (Honeywell, 2000). The ShadowPlant consists of five main sections as shown in Figure 3-8:

1. Feed preheater
2. Riser/Regenerator
3. Main Fractionator
4. Waste heat boiler
5. Air-preheater

The main operational states of the ShadowPlant are:

1. Cold startup
2. Steady state operation at various feed grades
3. Transition between grades
4. Shutdown

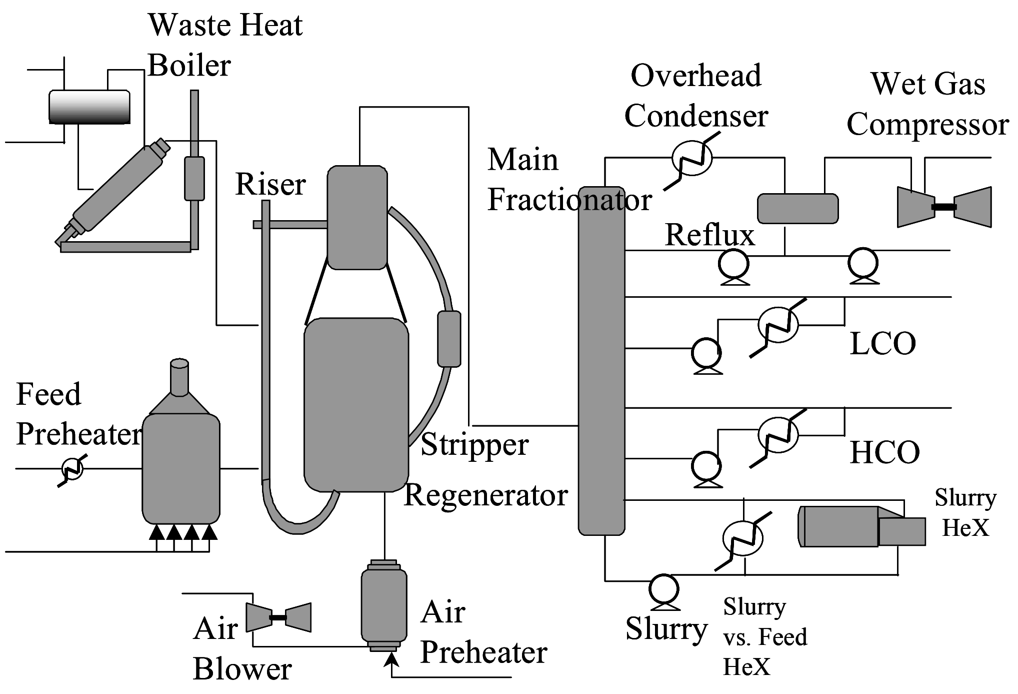

Figure 3-8: Schematic of FCCU Process

Of these, the startup is one of the most difficult transitions and requires experienced operators. One of the first steps in the startup of the ShadowPlant is the startup of the air-preheater section. The air blower is started slowly and operated at a steady state speed of about 4000 rpm. Air is heated in the preheater furnace to about 370°C and is used to initially increase the temperature of reactor/regenerator. Once this operation is stabilized, torch oil flow is started in the regenerator. This supplies the necessary heat and the fuel gas supply to the air-preheater is stopped. Catalyst is then added to the regenerator gradually and the temperature maintained around 400°C by manipulating the torch oil flow rate. When the catalyst level in the riser reaches 60%, the regenerator slide valve is opened and the catalyst moves through the riser to the reactor. Once the regenerator/reactor section has reached an intermediate steady state, with the regenerator temperature around 600°C, the Fractionator is flushed with kerosene and started up with fresh feed. When the Fractionator reaches a steady state, it is connected to the reactor/regenerator. Feed is started through the reactor and cracked products are fed to the main Fractionator. Throughput is gradually increased and steady state flow established. Details of the unit and the startup transition are reported by Honeywell (2000) and Sundarraman and Srinivasan (Srinivasan *et al.*, 2001).

Data Generation: The startup operation described above takes 40 to 60 hours. Several runs of the startup were performed following the standard operating procedure and data collected at 10 second intervals. While the procedure for starting up the FCCU was the same in all the runs, minor differences in the magnitudes and duration were introduced between the runs. Two runs G_1 and G_2 are considered in details here. The average duration of the startup transition in these runs was 54 hours. Random noise was also added to the measured variables to simulate measurement noise in the real process. A compressed database with 100-second sampling (that is 1 data point saved out of every 10) was generated from the original data in order to emulate the data compression in historians in real plants. The entire process has 335 measured variables. In each section, superfluous variables were first eliminated to decrease the computational requirements. The subset of the variables was selected based on the following criteria:

1. Only process variables (PV) such as pressure, temperature, or flow-rate were selected. Control signals and set points affect the section but do not directly reflect its state. Also, often there is a lag between a change in these variables and the process' response. Hence, these were removed.

2. Redundant variables were removed. For example, if two pressure measurements at adjacent positions along a pipeline were available, only one of them was selected if the information content in the difference is negligible.

For ease of explanation, in the following, the states are compared for each section separately.

3.3.1 Clustering of Regenerator States

Sixteen measurements are available in the regenerator section. Figure 3-9 shows the evolutions of three important variables 16PC108, 16PDC112, and 16TC116 from G_1. As can be seen in the figure, the section operates in several states.

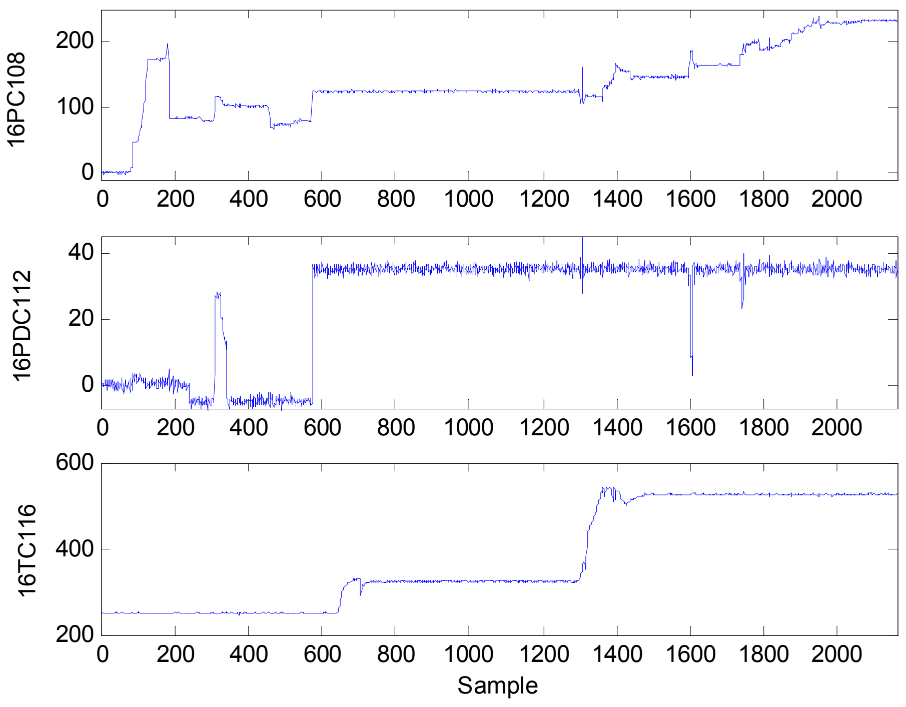

Figure 3-9: Three variables of regenerator section of ShadowPlant

State Identification: The PCA-based steady state identification method described in Section 3.2.1 was used to segregate the states. Figure 3-10 shows the variance contributions of the 16 PCs for G_1. The sixth PC was found to be an inflection point, so $k=6$ is used in the subsequent step to identify the modes. Figure 3-11(a) shows the time evolution of the first two scores and Figure 3-11(b), the identified modes and transitions. The value of TS_{min} was set to 90 min and the value of T_w and L

calculated from Equation [3-16]. Six modes, M_1^R to M_6^R and five transitions, T_1^R to T_5^R were identified. Analysis of these states and comparison with the operator log and operating procedures reveal that all the transitions are unique within a run – T_1^R corresponds to the regenerator startup operation, T_2^R to catalyst circulation, T_3^R, to the introduction of the feed, T_4^R, to the startup of the wet-gas compressor in the main Fractionator section, and T_5^R, to the increase in the feed flowrate to the riser. T_1^R includes several actions for starting up the section such as warming up the regenerator using hot air, introducing diesel for further temperature increase, and catalyst loading. T_2^R corresponds to opening the slide valve to enable catalyst circulation between riser and regenerator. Subsequently, during T_3^R, feed is introduced to the regenerator and reaction starts. M_1^R is the initial “cold-start” state of the regenerator section, and M_6^R the final state corresponding to steady state operation. M_2^R to M_5^R are intermediate modes during which the operator stabilizes the regenerator, starts up other sections of the FCCU, and links them to the regenerator section. Similar results were identified for data set G_2 as well.

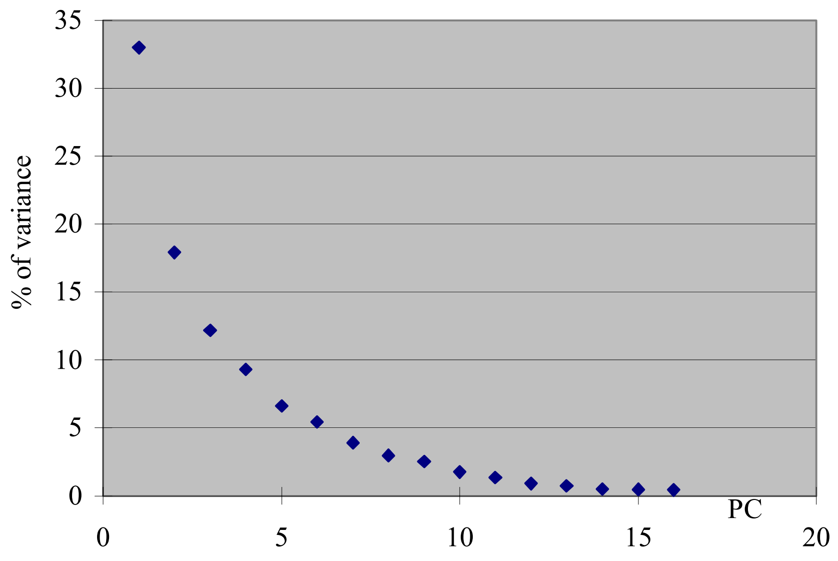

Figure 3-10: Plot of variance represented by each PCs in regenerator section

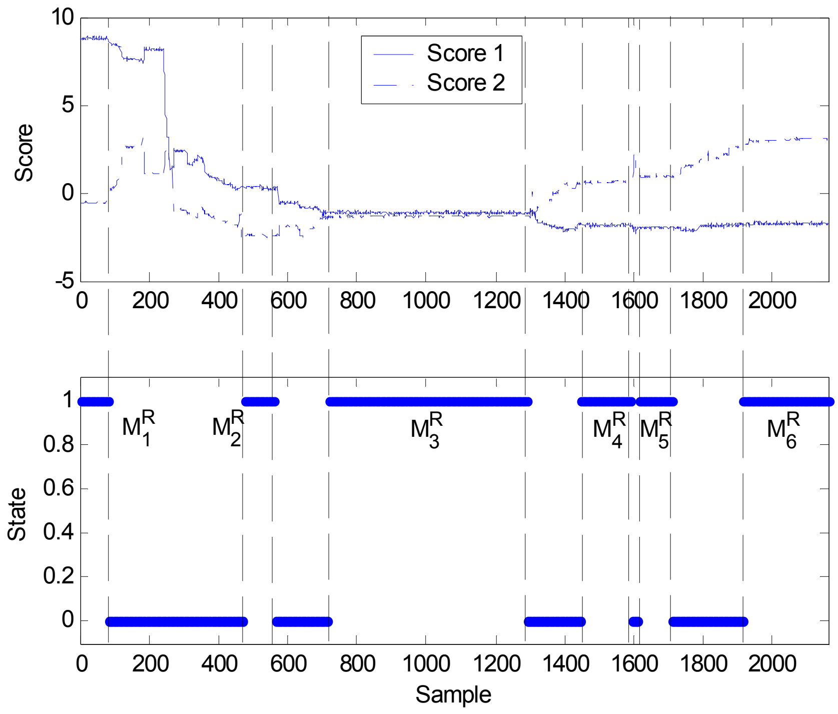

Figure 3-11: Eleven operating states identified in regenerator section based on 6 PCs, TS_{min} =90min (a) Evolution of first two (b) Durations of modes and transitions

Comparisons between the start and end times of the states from the operator log and those from the state identification are shown in TABLE 3-1. As can be seen, the two set of state timings are in close match. The only difference is at the edge of the states. On an average, this difference is 2.36 samples ─ a small error acceptable for steady state identification. It can therefore be concluded that the proposed method can identify steady states that corresponding to actual operating philosophy with high accuracy.

TABLE 3-1: Operating state identification error in regenerator section

Operating State	*a priori* Knowledge (x100s)	Identification Results (x100s)	Num. of Mis-classified samples
M_1^R	[1 82]	[1 80]	0
T_1^R	[83 470]	[81 472]	2
M_2^R	[471 573]	[473 568]	2

T_2^R	[574 718]	[569 712]	5
M_3^R	[719 1298]	[713 1296]	6
T_3^R	[1299 1439]	[1297 1436]	2
M_4^R	[1440 1594]	[1437 1592]	3
T_4^R	[1595 1613]	[1593 1612]	2
M_5^R	[1614 1732]	[1613 1732]	1
T_5^R	[1733 1949]	[1733 1952]	0
M_6^R	[1950 2160]	[1953 2160]	3
Average			2.36

State Clustering: Once the process data has been segmented into modes and transitions, similarity degrees between the modes and between the transitions can be calculated. During regenerator startup, none of the states is repeated within a run and all the 6 modes as well as the 5 transitions are unique. This is confirmed using the similarity calculations described in Section 3.3.2. TABLE 3-2 shows the S_M similarity degree among the modes. It can be seen that among the six modes, only M_4^R and M_5^R are similar (S_M = 0.915). This is verified by the process operation since the intermediate transition T_4^R is caused by an external disturbance that does not affect the dynamics of the regenerator. It can also be confirmed visually in Figure 3-11(a). The similarity degrees among the five transitions in G_1 calculated using S_{DPCA}^{λ} and S_{PCA}^{λ} are summarized in TABLE 3-3 and TABLE 3-4, respectively. Low similarities are calculated among all the transitions by both DPCA and PCA thus implying that the underlying dynamics in all the transitions are distinct. The highest similarity, observed between T_4^R and T_5^R, indicates that the two belong to different clusters. This can also be visually confirmed from the evolutions of 16PC108 during these transitions shown in Figure 3-12. Thus, the transition clustering results from the DPCA similarity factor is consistent with the overall evolutions as evident from the trends.

TABLE 3-2: S_M for modes in regenerator section during G_1

S_M	M_1^R	M_2^R	M_3^R	M_4^R	M_5^R	M_6^R
M_1^R	1	0.052	0.007	0.053	0.052	0.048
M_2^R		1	0.262	0.072	0.067	0.067
M_3^R			1	0.320	0.315	0.007
M_4^R				1	0.915	0.479
M_5^R					1	0.482
M_6^R						1

TABLE 3-3: DPCA similarity factors for transitions in regenerator section during G_1

Transition	T_1^R	T_2^R	T_3^R	T_4^R	T_5^R
T_1^R	1	0.145	0.139	0.179	0.246
T_2^R		1	0.476	0.131	0.101
T_3^R			1	0.114	0.269
T_4^R				1	0.357
T_5^R					1

TABLE 3-4: PCA similarity factors for transitions in regenerator section during G_1

Transition	T_1^R	T_2^R	T_3^R	T_4^R	T_5^R
T_1^R	1	0.148	0.229	0.251	0.456
T_2^R		1	0.548	0.227	0.231
T_3^R			1	0.292	0.473
T_4^R				1	0.538
T_5^R					1

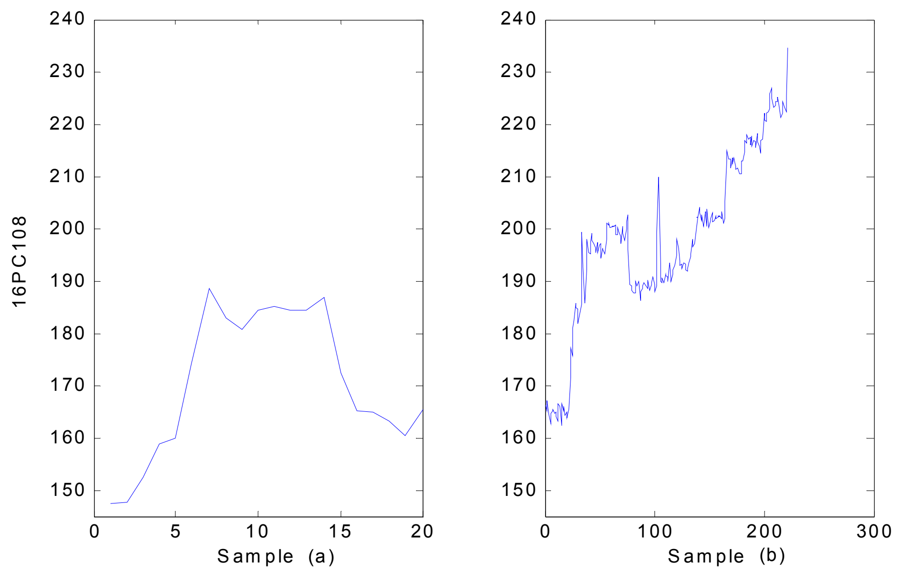

Figure 3-12: Evolution of 16PC108 in regenerator section startup (a) Transition T_4^R (b) Transition T_5^R

Next, the similarities among the states in two different regenerator startup runs – G_1 and G_2 are investigated. A comparison among all the modes from the two runs yielded an average S_M=0.973 between the corresponding ones, and an average $S_M=0.482$ for the second closest match. This indicates that the modes can be exactly matched despite run-to-run differences. A comparison of the transitions in G_1 and G_2 showing the best match and the second closest (in brackets) is shown in TABLE 3-5. Using the transitions in G_1 as the templates, all the corresponding transitions have been correctly identified from G_2. Figure 3-13 shows that there are significant differences (especially spikes at t=1305x100s and t=1360x100s) in the evolution of 16PC108 in G_1 and G_2, but these are correctly disregarded by the similarity factor since they do not indicate any change in the underlying dynamics.

TABLE 3-5: Comparing transitions from G_1 and G_2 in regenerator section

Factor	T_1^R	T_2^R	T_3^R	T_4^R	T_5^R
S_{DPCA}^{λ}	T_1^R : 0.997	T_2^R : 0.998	T_3^R : 0.933	T_4^R : 0.941	T_5^R : 0.981
	(T_5^R :0.257)	(T_3^R :0.329)	(T_2^R : 0.414)	(T_5^R : 0.350)	(T_4^R : 0.374)
S_{PCA}^{λ}	T_1^R : 0.998	T_2^R : 0.999	T_3^R : 0.923	T_4^R : 0.979	T_5^R : 0.989
	(T_5^R :0.451)	(T_3^R :0.385)	(T_5^R : 0.379)	(T_5^R : 0.644)	(T_4^R : 0.633)

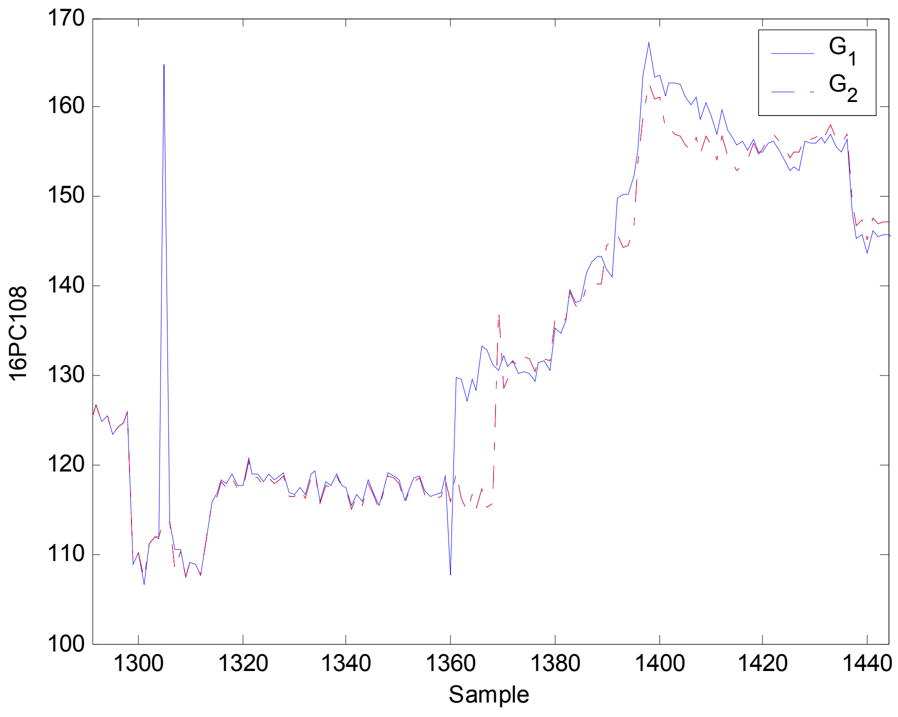

Figure 3-13: T_3^R from different runs in regenerator section

3.3.2 Clustering the Waste Heat Boiler Data

State Identification: The two-step clustering is performed on data from the Waste Heat Boiler section. The same parameter settings T_w=90 min, $\theta_d = 0.05$ and l=5 are used here as well. Based on the plot of P_i vs. i, 3 PCs are retained for state identification ($k=3$). As seen in Figure 3-14, the data was segmented into ten operating states – five modes, M_1^W to M_5^W and five transitions, T_1^W to T_5^W. The evolution of the first two scores is also shown for comparison. An analysis of the operations log reveals that T_1^W is caused by the regenerator warm-up action, T_2^W by catalyst loading, and T_3^W by the introduction of the feed. The feed introduction during T_3^W is accompanied by a ramp increase in fuel gas temperature 16TI120 as shown in Figure 3-15. T_4^W and T_5^W are characterized by sudden drop in temperature due to activities in other sections of the plant. In both cases, the temperature returns quickly to a steady state as seen in Figure 3-15. M_1^W is the initial "cold start" steady state of the unit and M_5^W the final state corresponding to steady state operation. M_2^W is a long intermediate state when other sections of the plant are started up. M_3^W and M_4^W are intermediate modes punctuated by T_4^W and T_5^W respectively.

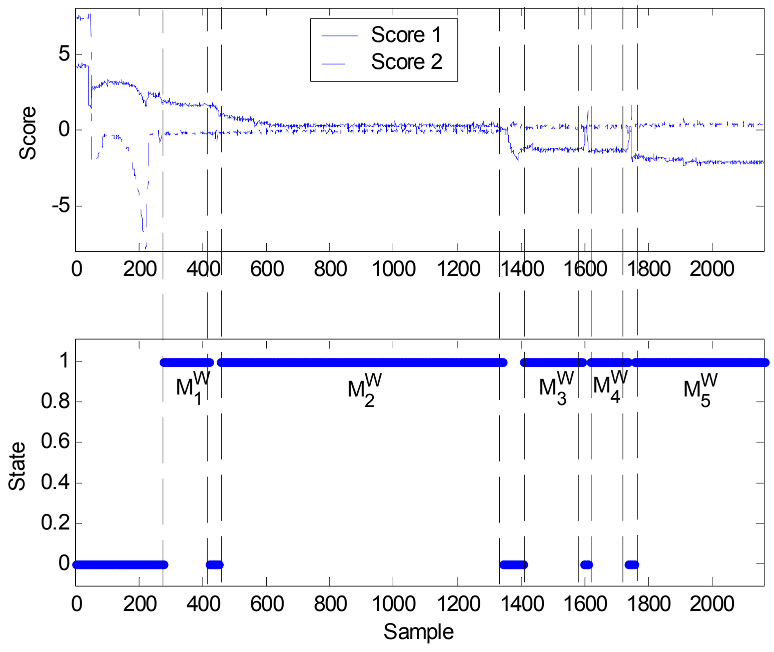

Figure 3-14: Ten operating states identified in waste heat boiler section based on 3 PCs, *TS*$_{min}$ =90min (a) Evolution of first two scores (b) Durations of modes and transitions

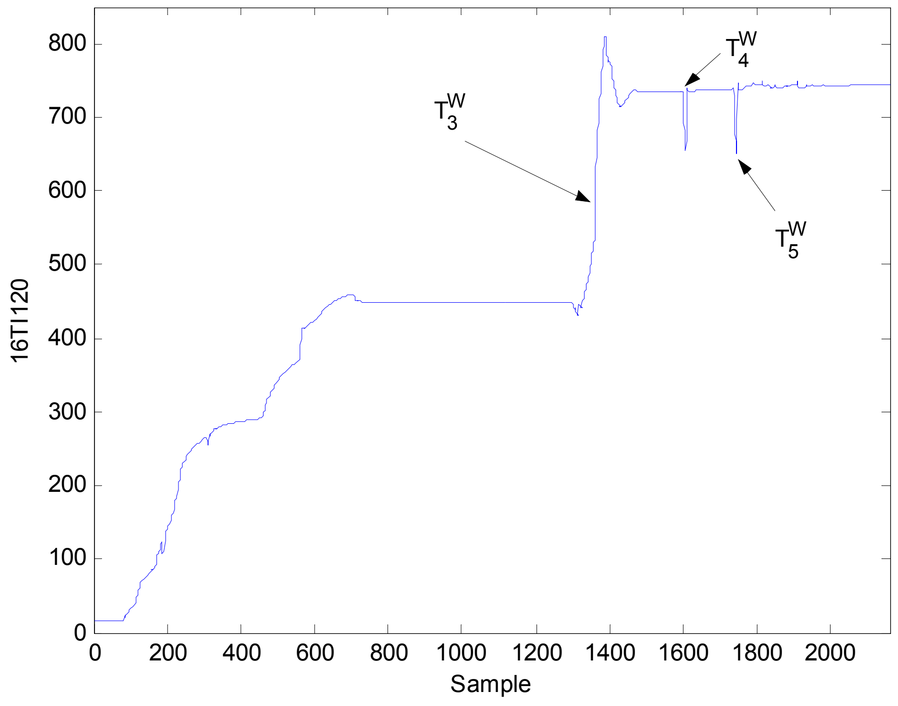

Figure 3-15: Two disturbances that lead to T_4^W and T_5^W in waste heat boiler section

State Clustering: Similarity degrees between the modes and between the transitions are calculated next. The close similarity between M_3^W and M_4^W discussed above is revealed by the high value of $S_M(M_3^W, M_4^W)$=0.976 in TABLE 3-6. The similarity degrees among the five transitions in G_1 are shown in TABLE 3-7. A high match $S_{DPCA}^{\lambda}(T_4^W, T_5^W) = 0.971$ is found between T_4^W and T_5^W accurately reflecting their similar dynamics. However, as seen in Figure 3-15, there is a small difference in their magnitudes and the fuel gas temperature 16TI120 settles down to a higher level in M_5^W. If it is necessary to differentiate between T_4^W and T_5^W for operational purposes, Equation [3-15] which includes the additional comparison of the terminal modes can be used. In this example, $S_M(M_4^W, M_5^W)$=0.834 indicates that the ending states for T_4^W and T_5^W are different. The operational difference of these transitions with T_3^W is also adequately captured by the lower $S_{DPCA}^{\lambda}(T_3^W, T_4^W)$ and $S_{DPCA}^{\lambda}(T_3^W, T_4^W)$ values. The similarities among transitions from G_1 and G_2 was also investigated. Among the total 10 transitions from the two runs, as summarized in TABLE 3-8, the corresponding transitions were always found to have high similarity values (between 0.931 and 0.996).

TABLE 3-6: S_M of modes in waste heat boiler section during G_1

S_M	M_1^W	M_2^W	M_3^W	M_4^W	M_5^W
M_1^W	1	0.533	0.455	0.432	0.267
M_2^W		1	0.607	0.584	0.419
M_3^W			1	0.976	0.811
M_4^W				1	0.834
M_5^W					1

TABLE 3-7: DPCA similarity factors for transitions in waste heat boiler section during G_1

Transition	T_1^W	T_2^W	T_3^W	T_4^W	T_5^W
T_1^W	1	0.092	0.780	0.531	0.567
T_2^W		1	0.08	0.151	0.145
T_3^W			1	0.763	0.826
T_4^W				1	0.971
T_5^W					1

TABLE 3-8: Comparing transitions from G_1 and G_2 in waste heat boiler section

Factor	T_1^W	T_2^W	T_3^W	T_4^W	T_5^W
S_{DPCA}^{λ}	T_1^W :0.990	T_2^W :0.943	T_3^W :0.996	T_4^W :0.970	T_5^W :0.931
	(T_3^W :0.790)	(T_4^W :0.312)	(T_5^W :0.867)	(T_5^W :0.927)	(T_5^W :0.945)
S_{PCA}^{λ}	T_1^W :0.990	T_2^W :0.997	T_3^W :0.998	T_4^W :0.999	T_5^W :0.999
	(T_4^W :0.864)	(T_1^W :0.647)	(T_5^W :0.937)	(T_5^W :0.998)	(T_4^W :0.999)

The waste heat boiler dataset also brings out the advantage of the S^{λ}_{DPCA} over S^{λ}_{PCA} for transition comparison. The S^{λ}_{PCA} among T^W_1 to T^W_5 is shown in TABLE 3-9. Here, while the largest value $S^{\lambda}_{PCA}(T^W_4, T^W_5) = 0.999$ correctly indicates the similarity between T^W_4 and T^W_5, the other large values, $S^{\lambda}_{PCA}(T^W_3, T^W_4) = 0.968$ and $S^{\lambda}_{PCA}(T^W_3, T^W_5) = 0.957$ confirm that S^{λ}_{PCA} cannot adequately distinguish between transitions based on their dynamics, that is, ramping up cannot be distinguished from spikes.

TABLE 3-9: PCA similarity factors for transitions in waste heat boiler section during G_1

Transition	T^W_1	T^W_2	T^W_3	T^W_4	T^W_5
T^W_1	1	0.539	0.856	0.859	0.851
T^W_2		1	0.431	0.539	0.551
T^W_3			1	0.968	0.957
T^W_4				1	0.999
T^W_5					1

Tuning Parameters: The effect of the tuning parameters on the clustering results is discussed here. The value of *k* required to adequately describe a dataset has received considerable attention in literature (Valle *et al.*, 1999). Figure 3-16 shows the effect of *k* on mode identification results for the regenerator section. When *k* is increased, more PCs corresponding to smaller eigenvalues are considered in the steady state identification. Due to the ensuing increase of noise in the scores, the durations of the transitions increase and modes can be merged into transitions. However, as can be seen, the effect of *k* is not significant and the majority of modes and transitions are consistently identified regardless of the value of *k*.

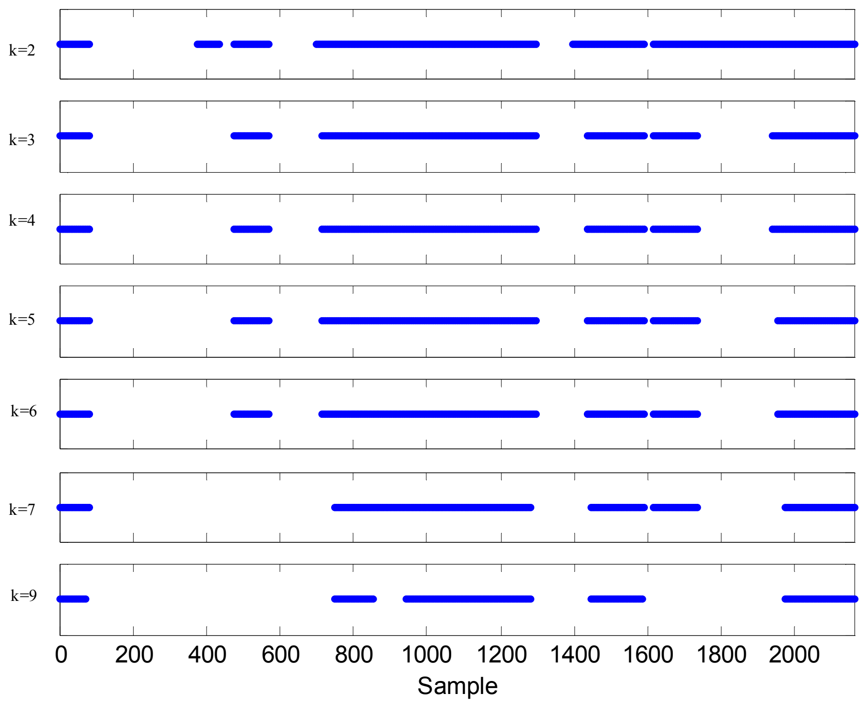

Figure 3-16: Steady states identification in regenerator section based on different *k*

The TS_{min} is an important parameter that significant affects the resolution of the states as well as the computational capacity. It can normally be specified using process knowledge, specifically the process operation, time constant and the expected duration of modes. Mode identification results for several values of TS_{min} are summarized in TABLE 3-10. When TS_{min} is increased from 60 min to 120 min, short periods of steady states within a transition are not flagged as modes and the total number of modes reduces from 10 to 6. The durations of the transitions consequently increase and the number of transitions decreases.

TABLE 3-10: Number of states identified for different TS_{min}

TS_{min}	Number of Modes	Number of Transitions
60min	10	9
90min	6	5
120min	6	5

The variation tolerance threshold θ_d also affects the length and number of modes identified. However, as shown in Figure 3-17 this effect is marginal. For a wide range of θ_d, from 3% to 12%, the number of modes identified is the same but only their start and end times vary slightly. When θ_d is increased, new modes are identified (see θ_d increase from 11% to 13%) and some modes are amalgamated (see θ_d increases from 14% to 17%). The number of modes identified will increase to a maximum and then reduce to one for large values since in the latter case, the whole dataset will be deemed to be in a mode. We have found θ_d =5% to be a suitable choice for most cases.

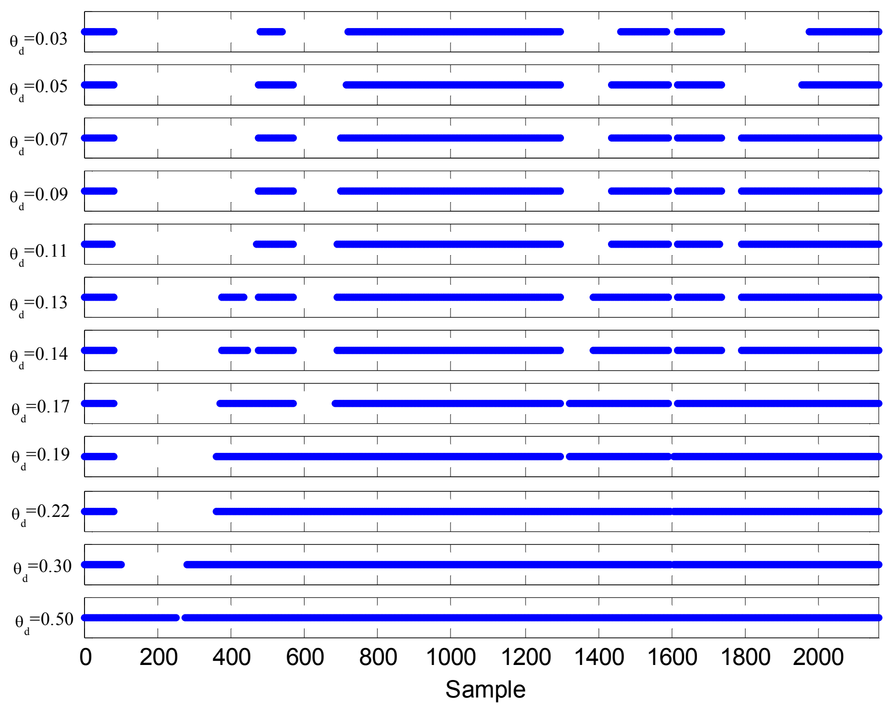

Figure 3-17: Steady states identification in regenerator section based on different θ_d

θ_M and θ_T can be estimated from historical data based on *a priori* knowledge of the process. They should reflect the noise level and the run-to-run variations expected in the process. We have found the value of *0.9* to be suitable for both and used this value in all case studies. As seen in TABLE 3-2 and TABLE 3-6 for inter-mode similarities, TABLE 3-3 and TABLE 3-7 for inter-transition similarity, and TABLE 3-5 and TABLE 3-8, for intra-transition similarity can all be adequately identified with this threshold.

The regression order *l* plays a significant role in the ability of DPCA to differentiate between transitions. Ku *et al.* (1995) proposed an iterative procedure for determining *l*. While *l=2* has been reported to be adequate for many processes, Figure 3-18 reveals that, to completely capture the nonlinearities in the ShadowPlant, a value of *l=5* is necessary. A larger lag *l* is better to differentiate transitions; however, it leads to increased computational requirements. As shown in the figure, when *l* is increased, the similarity factors between the different transitions (such as T_3^W and T_4^W and T_3^W and T_5^W) declines faster while the similarity factor between analogous transitions (such as T_4^W and T_5^W) reduces slowly. For clustering therefore, an intermediate value such as *l=5,* presents a good tradeoff between accuracy and calculation speed. In the Section 3.4, I present the results from clustering process states in the Tennessee Eastman process.

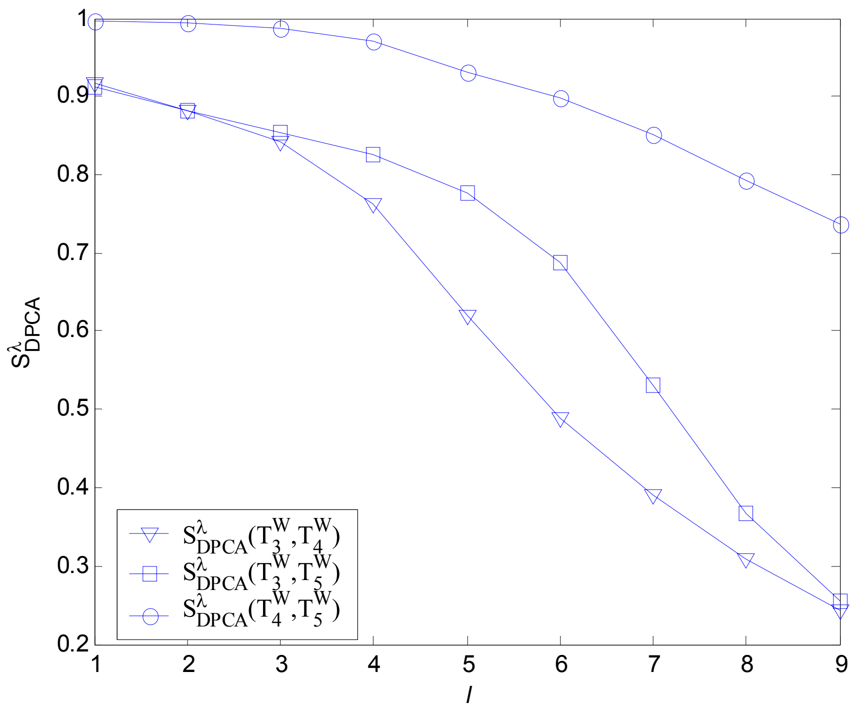

Figure 3-18: Effect of lag *l* on S^{λ}_{DPCA} in waste heater boiler section

3.3.3 Comparison of Proposed Method with Existing Approaches

Several existing clustering algorithms have been discussed in the literature review. Static clustering techniques such as k-means and c-means clustering investigate only the relative position between feature vectors and centers. The auto-correlation information in temporal signal is lost. Therefore, these methods are inapplicable for process states which are characterized by the temporal evolution of the process variables.

Klaus *et al.* (1996) proposed a neural network system consisting of q single networks, and $q>m$, where m is the estimated number of clusters. The system is trained so that each network approximates the underlying regression function f of a single cluster. After training, clustering of a new feature vector is achieved through the comparison of q prediction errors from the q networks. This method is tested on waste heat boiler section. The procedure of Klaus' method is summarized here:

1. Initializing a number of *q* neural networks. There is no requirement for the type of neural network. The number *q* is selected randomly, but it must be larger than the number of clusters. That is, the number of clusters has to be estimated *a priori*
2. Training *q* networks simultaneously by using gradient decreasing $\Delta w_i \propto -\sum_t p_i^t \frac{\partial \varepsilon_i^t}{\partial w_i}$. w_i is the network weights. The weighting coefficient p_i^t corresponds to the relative probability for the contribution of network *i* and the $\sum_i p_i = 1$. p_i^t will reinforce the training of one of the networks to approximate the particular function, while at the same epoch other networks remain unchanged or very small change.
3. After training, finding out the networks segmented the time series almost exactly at the switching points, while others will be drifted off.
4. Using these networks to do winner-take-all clustering in term of their prediction error.

Klaus' method clusters the process data based on the neural network prediction error. It is therefore only suitable to uni-variate case. In waste heat boiler section, a single variable 16FC118 is selected for the test. The evolution of the variable is shown in Figure 3-19.

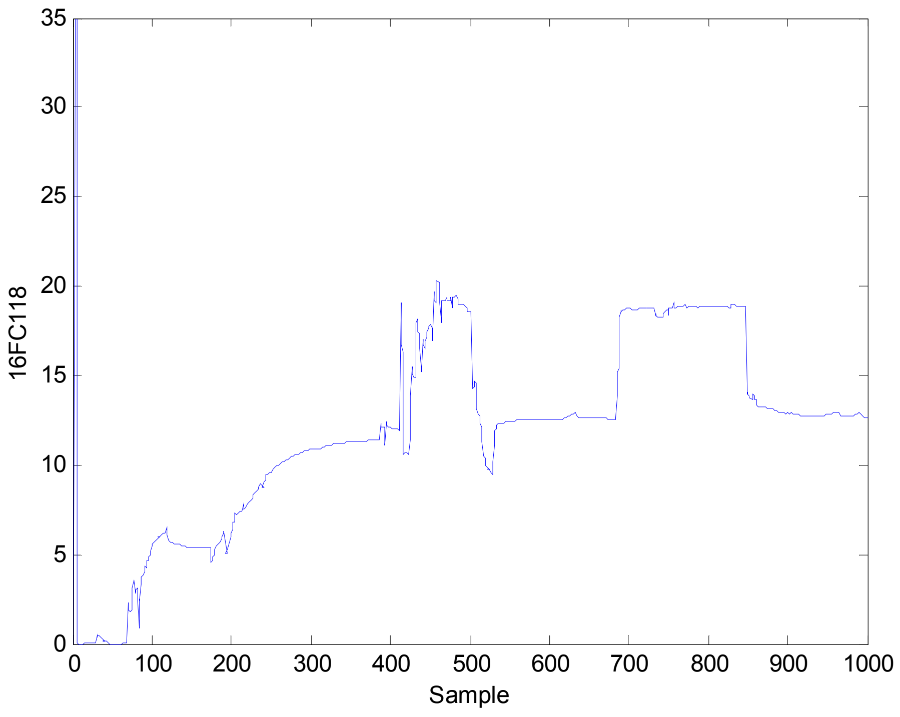

Figure 3-19: Evolution of 16FC118 in waste heat boiler section

Six RBF networks with same centers and random initial weights are built. After training, the data from G_2 are fed into the network and the prediction errors are investigated. At each time instance, the smallest prediction error is picked out and the corresponding network is chosen to mark the data cluster. The final validation results are shown in Figure 3-20. In the figure, the solid line represents the operating states from 1 to 6 identified by the networks. It can be seen from the figure, the underlying dynamics of the steady state is uniform. In these steady states, the identification results of mode are acceptable such as M_1 and M_2. However, during process transition T_1 and T_2, the process presents rich dynamics that can not be categorized by a single function. Therefore, in these states, the clustering results are very confused. The method proposed by Klaus required that the process does not switch dynamics frequently. This is normally not satisfied in real process especially during transition. In addition, it is difficult to know the number of clusters *a priori* and select the number of networks accordingly.

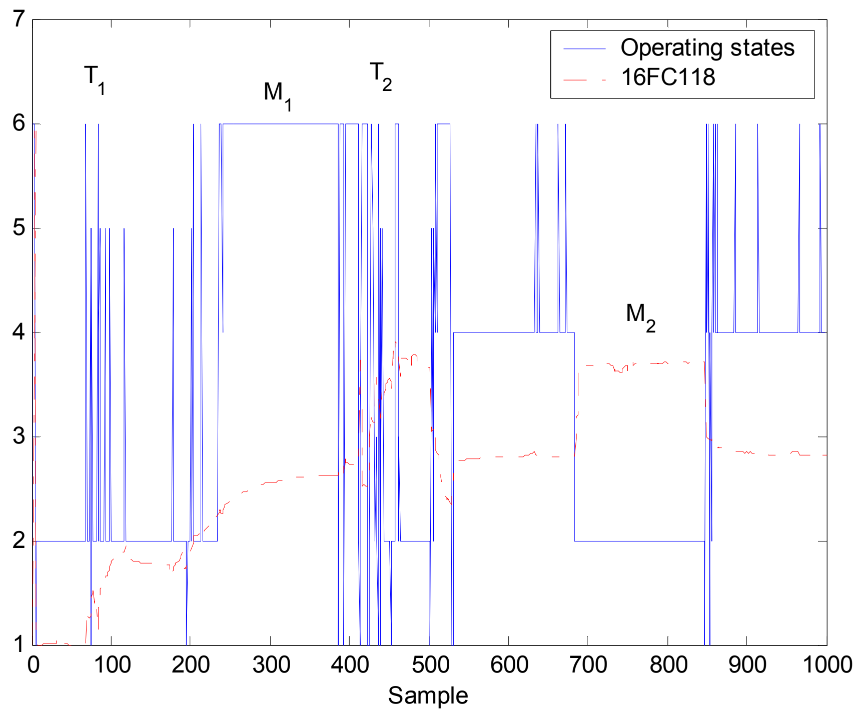

Figure 3-20: Six operating states identified in waste heat boiler section by Klaus's method

As discussed in literature review, a trend-based approach to mode and transition identification has been previously proposed by Sundarraman *et al.* (2003). A comparison of the results obtained from the trend-based approach with the proposed method indicates that both methods identify essentially the same modes and transitions. The trend-based approach identified seventeen operating states comprising nine modes and eight transitions for the regenerator section and fifteen operating states including eight modes and seven transitions for the waste heater boiler section. There is a one-to-one match in the long duration modes and transitions found by the two methods. Examples of this are shown in Figure 3-21 and Figure 3-22. Figure 3-21 (a) shows a mode identified by the trend-based approach in waste heat boiler section and Figure 3-21 (b), the corresponding state identified by the PCA method. It can be noted that both methods identify almost the same beginning and end positions of these two steady states. Both the approaches find the same two instances of this mode during the entire startup. Similarly, Figure 3-22 shows a transition identified in waste heat boiler section by the two approaches. Differences were found between the two approaches in the characterization of the short-duration states (both modes and transitions) and are to be expected because of the differences in the comparison schemes and the parameter settings.

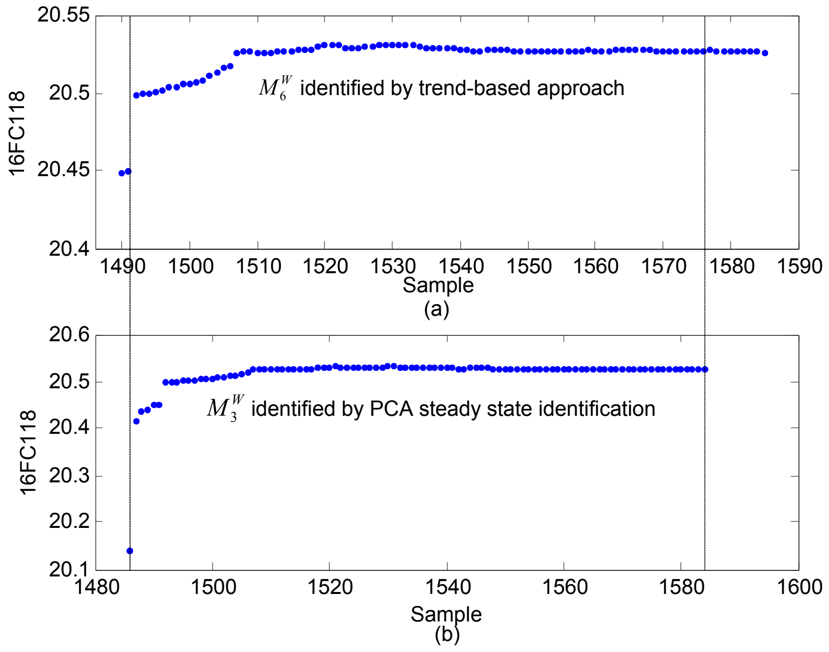

Figure 3-21: Steady state identified in waste heat boiler section (a) Steady state identified by trend-based approach (b) Steady state identified by proposed PCA approach

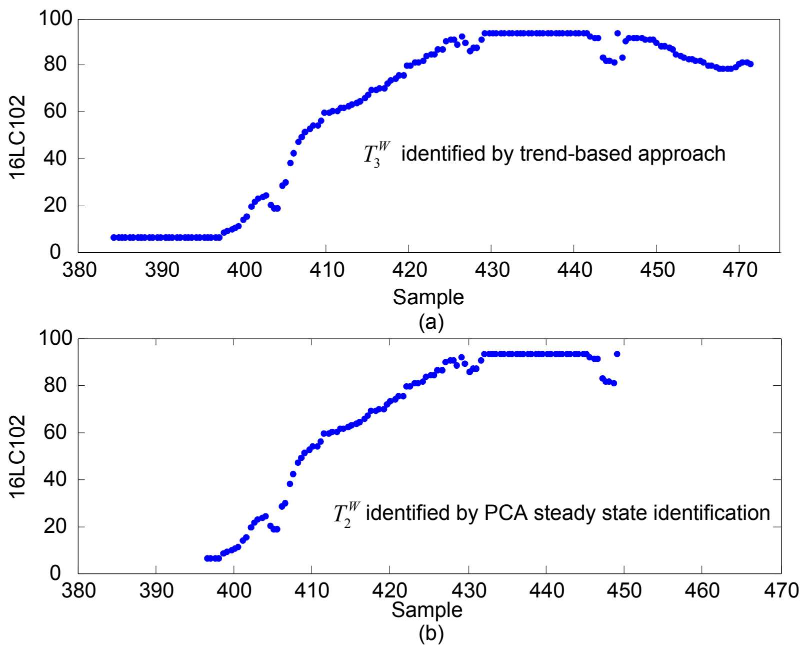

Figure 3-22: Transition identified in waste heat boiler section (a) Transition identified by trend-based approach (b) Transition identified by proposed PCA approach

From the above comparison, one key advantage of the proposed PCA clustering approach is its inherent multivariate nature whereas the Klaus and trend-based approaches require that each variable is investigated one by one. Also, the trend and the DTW-based transition comparison are computationally expensive and are therefore performed on selected key variables instead of all variables in a process unit. The selection of key variables for a process unit is a nontrivial problem which is eliminated in the PCA-based approach. The analogous problem of selecting a subset of the PCs is a simpler one with clear guidelines. The PCA-based approach used here is based on the normalization of process variables; transition comparison therefore focuses only on process trends and not on the magnitudes of the variable. In contrast, the trend-based approaches use additional quantitative process information such as the starting value of a variable during transition and hence perform a stricter comparison.

The PCA operation in the proposed clustering method can also filter the process measurement noise. The steady state identification and transition similarity comparison therefore are not affected significantly by the disturbance. However, Klaus algorithm examines the underlying function of the process data by monitoring the prediction error. This is comparatively sensitive to the process noise which can cause the process unobservable during transitions.

3.4 Tennessee Eastman Process

The Tennessee Eastman (TE) process reported by Downs and Vogel (1993) is a popular test bed for process systems applications such as plant-wide control, optimization, predictive control, fault diagnosis and signal comparison. The process produces two products (G and H) and a byproduct (F) from reactants A, C, D, and E. The process flowsheet is shown in Figure 3-23. The process has five unit operations: a two-phase reactor, a product condenser, a flash separator, a recycle compressor, and a product stripper. The simulation in this work uses the closed-loop process Matlab simulator developed by Singhal (2001) based on the base control structure of McAvoy and Ye (1994). There are 53 variables in the process: 22 of these are process measurements variables, 19 are component compositions, and 12 process manipulated variables. The agitator speed is left constant in the simulation, hence it is not considered in the analysis. The remaining 52 variables are investigated to evaluate the DPCA-based method robustness. A sampling time of 3 minutes is used here.

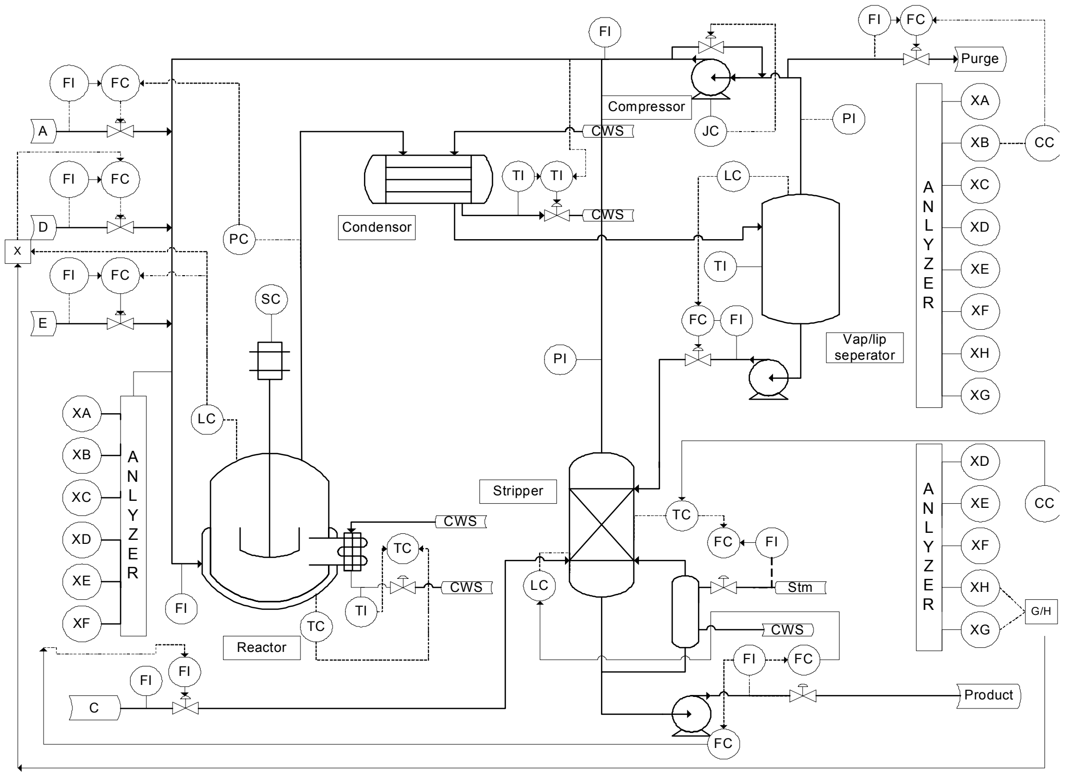

Figure 3-23: Schematic of Tennessee Eastman process with control system

In order to simulate agile, multi-mode operation, I have introduced five set point changes in the process, which resemble multi-mode operations in real industrial processes. The five new classes XD1 – XD5 affect the A feed flowrate, reactor pressure, reactor level, reactor temperature, and compressor work. Further, different instances (runs) of the same class have different start times, duration, and magnitude. For example, during XD1-A, the flowrate of A feed from upstream is increased from the base case value of 0.25052 kscmh to 0.3902 kscmh (a 60% change) at t=180 min. After the process recovers from these, the inverse change, decreasing the A feed flow is introduced at t=780 min. The process is then allowed to return to a steady state. The effect on the A flow rate (XMEAS(1)) and the downstream pressures (XMEAS(13) and XMEAS (16)) is shown in Figure 3-24. Two other instances XD1-B and XD1-C with magnitude of 55% and 50% are also introduced as shown in Table 3-11. As summarized in TABLE 3-12, similar changes were introduced to bring forth classes XD2–XD5.

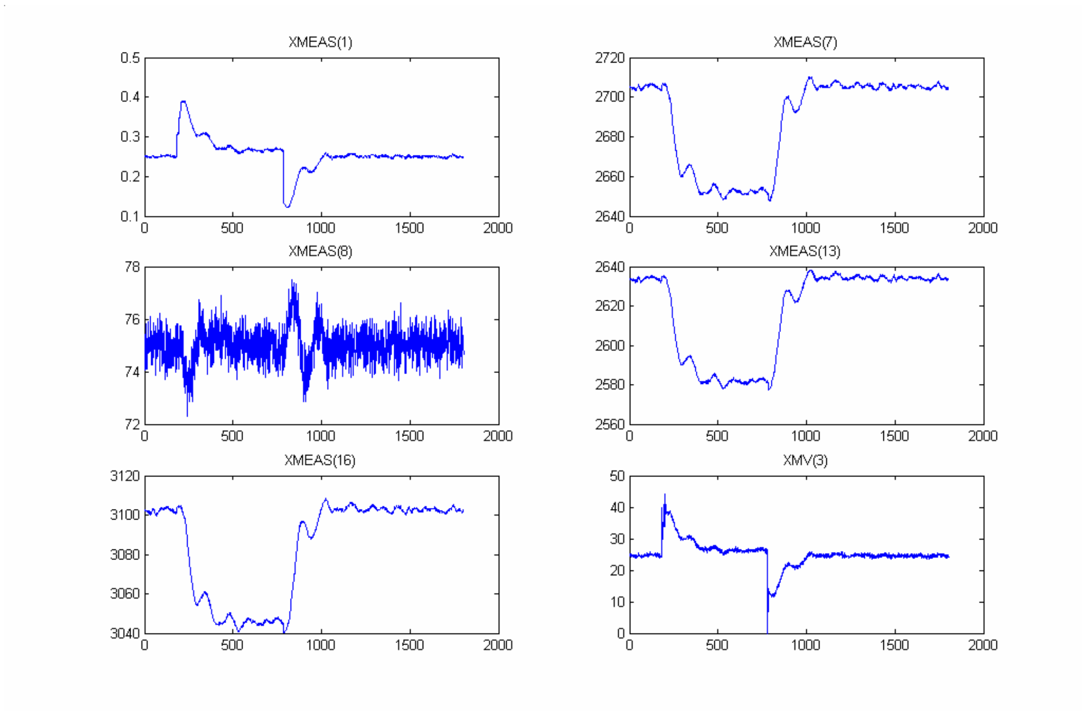

Figure 3-24: Process signals for XD1

Table 3-11: Disturbance profile for XD1

	Target	Time (min)	Target	Time (min)	Target	Time (min)	Target	Time (min)
XD1-A	1.20*Base value	180	1.40*Base value	190	1.60*Base value	200	1.0*Base value	780
XD1-B	1.15*Base value	240	1.35*Base value	254	1.55*Base value	268	1.0*Base value	900
XD1-C	1.10*Base value	300	1.30*Base value	318	1.50*Base value	336	1.0*Base value	1020

TABLE 3-12: Operating states for Tennessee Eastman process

Case	Description	Base Value	Change
XD1	Increase ‘A’ feed flowrate	0.25052 kscmh	50% 55% 60%
XD2	Increase of Reactor pressure	2705 kPa gauge	6% 6.5% 7%
XD3	Increase in the reactor level	75%	12% 13.5% 15%
XD4	Reactor temperature increase	120.04°C	12% 13.5% 15%
XD5	Decrease of compressor work	341.43 kW	12% 13.5% 15%

Each instance of the five classes has five operating states – two transitions corresponding to the two transients and three modes. The two-step clustering method was applied to identify and compare these states. Steady state identification was performed using k=5, T_w=120 min and $\theta_d = 0.1$. In all cases, the five operating states were successfully identified with an average change point detection error of 1.1%. The similarity factors S_M among the 15 modes for the instances are shown in TABLE 3-13. As can be seen, the second mode (M_2) in any run can be clearly differentiated from the first (M_1) and the third (M_3) with an average S_M less than 0.55. M_1 and M_3 are clustered together in four classes with an average S_M of about 0.92. In these cases, M_1 and M_3 are similar since the process is returned near the original state by the controllers after transition T_2. During XD4, the stripper steam valve is 47% open during M_1 but only 34% during M_3. This difference in the process state is reflected by the low similarity (S_M= 0.58) between them.

TABLE 3-13: Average euclidean distances among modes in XD1 — XD5

Case	$S_M(M_1,M_2)$	$S_M(M_2,M_3)$	$S_M(M_1,M_3)$
XD1	0.806 ± 0.010	0.800 ± 0.010	0.954 ± 0.003
XD2	0.518 ± 0.020	0.523 ± 0.030	0.877 ± 0.004
XD3	0.607 ± 0.040	0.606 ± 0.040	0.910 ± 0.000
XD4	0.329 ± 0.070	0.359 ± 0.070	0.586 ± 0.01
XD5	0.483 ± 0.055	0.550 ± 0.050	0.926 ± 0.006

A regression order of *l=25* was found to be optimal for this process. Within the same class, all six transitions show similar dynamics. The average S_{DPCA}^{λ} for intra-class comparison is 0.958. This high similarity factor reflects the fact that the transitions from the same class, even of different magnitudes, are clustered together. The similarity comparison is thus robust to run-to-run magnitude differences.

The DPCA factor S_{DPCA}^{λ} is then used to cluster the 30 transitions and the results are shown in TABLE 3-14. The average value of S_{DPCA}^{λ} between the 10 (that is, C_2^5) pairs of transitions is 0.733 and illustrates the ability to differentiate between operating states resulting from different sources and exhibiting different dynamics. Two pairs of operating states are found to have similar dynamics, (XD1,XD2)=0.871 and (XD3,XD5)=0.880. As discussed before, if differentiation between XD1 and XD2, or XD3 and XD5 is required for an application, it can be achieved by using the supplementary mode comparisons in Equation [3-15]. In this case, SM(XD1-M2, XD2-M2)=0.319 and SM(XD3-M2, XD5-M2)=0.466, and the magnitude-sensitive comparison reveals that XD1 and XD2, XD3 and XD5 are from different classes.

The transition comparison method proposed here is general and can be used for other applications such as fault diagnosis. The twenty disturbances proposed by Downs and Vogel [26] are used for the purpose. Twenty disturbances (IDVs) for the TE plant involving inlet-cooling temperature, A/C feed ratio, sticking valve, etc. These disturbances have been built into the simulation and have been popularly used in literature for testing process control and monitoring algorithms. In this work, IDV6 is not considered. Each disturbance can be initiated randomly and the process evolution observed. In the closed-loop process, usually the controller would reject the disturbance and eventually bring the process back to the original state. For the twenty IDVs, this period ranges from 12 to 24 hours.

Any pattern comparison algorithm is evaluated through its ability to differentiate among the disturbances. A PCA similarity factor based comparison of the 19 disturbances is shown in

TABLE 3-15 and brings forth the difficulty in differentiating among them due to the nonlinearity and noise in the process. Among the 210 disturbance-pairs, 39 pairs corresponding to eleven disturbances are similar ($S_{PCA}^{\lambda} \geq 0.9$) indicating that these faults are closely clustered in the feature space. This is also illustrated by the spread in the similarity factor (mean 0.67 and the standard deviation of 0.2) among the 190 non-diagonals. Another shortcoming of the PCA similarity factor is inconsistency in the clusters (that is a lack of the transitive property, as explained below). Similar difficulties were pointed out by Huang [28] where nonlinear extensions to PCA could correctly isolate in the order of 50% of the cases.

The DPCA-based transition similarity factor was used to differentiate between the IDVs. The comparison results using l=25 are shown in TABLE 3-16. We can see that there are 22 disturbance-pairs that are similar ($S_{DPCA}^{\lambda} \geq 0.9$). The S_{DPCA}^{λ} based similarity comparison consequently cannot differentiate among 9 faults, a clear improvement over the above results. The lower mean (0.60) and the larger standard deviation (0.27) also highlight the efficacy of the S_{DPCA}^{λ} metric. In addition, the results from S_{DPCA}^{λ} are consistent, that is if IDV3 is similar to IDV4 and IDV4 is similar to IDV5 then IDV3 and IDV5 are also flagged as similar. Exploiting this, for faults such as IDV3 where a single cause cannot be pinpointed, the comparison results can be used to narrow the search to only the faults in that cluster (for example IDV4,IDV5,IDV9).

TABLE 3-14: DPCA similarity factors among transitions in XD1 ─ XD5

Transition	XD1-T_1	XD1-T_2	XD2-T_1	XD2-T_2	XD3-T_1	XD3-T_2	XD4-T_1	XD4-T_2	XD5-T_1	XD5-T_2
XD1-T_1	0.926 ± 0.040	0.896 ± 0.010	0.878 ± 0.040	0.832 ± 0.090	0.712 ± 0.010	0.742 ± 0.007	0.458 ± 0.006	0.608 ± 0.008	0.778 ± 0.009	0.822 ± 0.004
XD1-T_2		0.996 ± 0.001	0.900 ± 0.004	0.876 ± 0.060	0.822 ± 0.005	0.812 ± 0.007	0.545 ± 0.001	0.686 ± 0.008	0.840 ± 0.008	0.839 ± 0.009
XD2-T_1			0.927 ± 0.040	0.855 ± 0.050	0.625 ± 0.008	0.629 ± 0.007	0.426 ± 0.020	0.562 ± 0.008	0.696 ± 0.008	0.714 ± 0.006
XD2-T_2				0.955 ± 0.050	0.847 ± 0.040	0.860 ± 0.007	0.556 ± 0.020	0.749 ± 0.008	0.848 ± 0.010	0.864 ± 0.010
XD3-T_1					0.998 ± 0.002	0.980 ± 0.001	0.632 ± 0.050	0.820 ± 0.005	0.871 ± 0.008	0.862 ± 0.010
XD3-T_2						0.998 ± 0.001	0.617 ± 0.003	0.814 ± 0.004	0.899 ± 0.008	0.889 ± 0.001
XD4-T_1							0.997 ± 0.003	0.917 ± 0.004	0.511 ± 0.006	0.504 ± 0.009
XD4-T_2								0.992 ± 0.000	0.689 ± 0.006	0.647 ± 0.016
XD5-T_1									0.995 ± 0.005	0.952 ± 0.003
XD5-T_2										0.996 ± 0.001

TABLE 3-15: S_{PCA}^{λ} among twenty IDVs

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1.000	0.231	0.447	0.456	0.460	1.000	0.530	0.851	0.463	0.439	0.477	0.562	0.726	0.447	0.450	0.431	0.556	0.572	0.429	0.597
2		1.000	0.509	0.504	0.496	0.231	0.329	0.238	0.505	0.466	0.426	0.394	0.374	0.490	0.501	0.425	0.364	0.141	0.483	0.165
3			1.000	0.971	0.997	0.447	0.461	0.591	0.996	0.900	0.874	0.902	0.549	0.973	0.999	0.915	0.628	0.612	0.980	0.626
4				1.000	0.971	0.456	0.472	0.591	0.972	0.872	0.924	0.889	0.567	0.984	0.972	0.886	0.636	0.605	0.959	0.633
5					1.000	0.460	0.480	0.613	0.996	0.901	0.881	0.918	0.557	0.973	0.998	0.919	0.641	0.639	0.980	0.652
6						1.000	0.530	0.851	0.463	0.439	0.477	0.562	0.726	0.447	0.450	0.431	0.556	0.572	0.429	0.597
7							1.000	0.653	0.477	0.427	0.477	0.593	0.628	0.466	0.465	0.461	0.557	0.574	0.468	0.617
8								1.000	0.606	0.563	0.656	0.783	0.692	0.592	0.593	0.590	0.754	0.894	0.605	0.898
9									1.000	0.900	0.884	0.915	0.564	0.972	0.996	0.915	0.639	0.626	0.979	0.640
10										1.000	0.789	0.839	0.512	0.872	0.899	0.938	0.580	0.584	0.893	0.593
11											1.000	0.878	0.548	0.911	0.875	0.807	0.710	0.674	0.885	0.697
12												1.000	0.598	0.887	0.902	0.876	0.762	0.819	0.910	0.834
13													1.000	0.555	0.552	0.490	0.684	0.562	0.522	0.598
14														1.000	0.975	0.888	0.635	0.604	0.958	0.636
15															1.000	0.916	0.625	0.614	0.979	0.627
16																1.000	0.610	0.648	0.913	0.652
17																	1.000	0.841	0.655	0.857
18																		1.000	0.642	0.994
19																			1.000	0.658
20																				1.000

TABLE 3-16: S_{DPCA}^{λ} with *l=25* among twenty IDVs

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	0.210	0.245	0.322	0.247	1.000	0.419	0.730	0.244	0.245	0.337	0.264	0.677	0.251	0.238	0.202	0.385	0.319	0.231	0.313
2		1.000	0.234	0.284	0.217	0.210	0.285	0.187	0.228	0.225	0.222	0.149	0.341	0.222	0.217	0.159	0.319	0.075	0.203	0.087
3			1.000	0.938	0.992	0.245	0.332	0.524	0.991	0.863	0.812	0.839	0.368	0.976	0.998	0.868	0.566	0.567	0.967	0.592
4				1.000	0.937	0.322	0.376	0.531	0.937	0.800	0.865	0.802	0.429	0.955	0.938	0.804	0.570	0.556	0.911	0.580
5					1.000	0.247	0.350	0.559	0.992	0.870	0.825	0.875	0.365	0.971	0.995	0.878	0.582	0.601	0.969	0.625
6						1.000	0.419	0.730	0.244	0.245	0.337	0.264	0.677	0.251	0.238	0.202	0.385	0.319	0.231	0.313
7							1.000	0.508	0.339	0.306	0.406	0.415	0.537	0.343	0.331	0.313	0.459	0.497	0.344	0.487
8								1.000	0.542	0.492	0.622	0.732	0.553	0.521	0.524	0.524	0.714	0.840	0.558	0.838
9									1.000	0.865	0.833	0.868	0.363	0.973	0.992	0.873	0.579	0.588	0.967	0.614
10										1.000	0.710	0.791	0.336	0.838	0.863	0.893	0.525	0.567	0.859	0.582
11											1.000	0.836	0.406	0.860	0.810	0.732	0.673	0.653	0.819	0.678
12												1.000	0.332	0.824	0.841	0.834	0.723	0.796	0.862	0.820
13													1.000	0.367	0.358	0.280	0.546	0.350	0.338	0.351
14														1.000	0.976	0.842	0.567	0.565	0.943	0.589
15															1.000	0.869	0.558	0.568	0.967	0.592
16																1.000	0.554	0.602	0.874	0.622
17																	1.000	0.840	0.593	0.848
18																		1.000	0.603	0.988
19																			1.000	0.632
20																				1.000

3.5 Conclusions and Discussion

Operating states in multi-mode operation evolve over long periods of time. In batch operation as well as during startup, grade change, and shutdown activities in continuous processes, the process is characterized by complex temporal dynamics. Process automation applications for advanced and supervisory control are applicable to specific process states (usually the quasi-steady state). In order to function properly during periods of transition, these applications have to be dynamically reconfigured with state-specific parameters and models. State identification is therefore essential during agile operations. As a step towards that goal, I have developed a multivariate statistics-based methodology to cluster process states in historical operations data.

Process data are first segmented based on regions of steady state operations into modes and transitions. Similar modes are identified by comparing their means. A new dynamic PCA-based similarity factor, which accounts for the autoregressive nature, has been developed to cluster transitions. The technique was applied to data collected from two kinds of agile operation – startup of a simulated FCCU and multi-mode operation in the Tennessee Eastman plant simulation. Application to fault isolation was also demonstrated in the latter case study. In all cases, the method correctly identified and clustered the modes and transitions. These tests thus highlight the applicability of the state segregation and the superiority of the DPCA-based transition similarity factor.

The proposed strategy is compared with several existing algorithm such as Klaus clustering method and trend-based approach. The proposed methodology offers several advantages. It accounts for the multivariate nature of chemical processes naturally and is hence superior to methods like Klaus, DTW and qualitative trend comparison, which are designed for one-dimensional data. Also, the trend and the DTW-based transition comparison are computationally expensive and are therefore performed on selected key variables instead of all variables in a process unit. The selection of key variables for a process unit is a nontrivial problem which is eliminated in the proposed approach. The homologous problem of selecting k, the subset of the PCs, is a simpler one with clear guidelines. The normalization of process variables during transition comparison ensures that the focus is on the underlying dynamics. The magnitude of the change is correctly ignored thus making the comparison more robust to run-to-run differences. However, if a stricter comparison is necessary, magnitudes can also be included through the S_M factors as demonstrated in the TE case study. The DPCA factor results in a quantitative comparison of two transitions instead of a binary judgment. While this makes the similarity results dependent on the tuning parameters, the results are consistent for a range of parameter values. The proposed DPCA factor is also compared with the traditional PCA factor. DPCA factor shows superiority especially when the process presents high auto-correlation. I have also developed clear guidelines for setting the parameter values. The computational requirement of this approach is quite modest and allows large amount of data to be analyzed in a short period.

Chapter 4. Neural Network Systems for Multivariate Temporal Pattern Classification

4.1 Introduction

In Chapter 3, I have proposed a DPCA-based multivariate clustering method, which can segment and group the process operating states according to their temporal features. The resulted operating state knowledge can then be used to train the supervisory classifier to identify the operating state in real time. However, the proposed clustering method is not suitable to univariate process data. In case *a priori* knowledge is needed to train a network monitoring univariate process (like the case of OVON which will be discussed later), a visually checking for clustering is adopted.

Consider a process whose state at time *t* is reflected by the *nd* dimensional vector $X^{nd}(t):[X^d(t), X^d(t-1), \cdots, X^d(t-l)]$, where $X^d(t-l)$ is the value of *process vector* $[x_1, x_2, \cdots, x_d]$ at time *t-l*, *d* denotes the number of variables, *l* represents the size of the data window or the memory of the system, $nd = d \times (l+1)$ and $X^{nd} \in \mathbb{R}^{nd}$. Here the vector X^{nd} that reflects the temporal pattern at a certain time is called the *process feature vector* (note its difference from process vector). The problem of identifying the operating state from the process measurements is a data classification problem of finding a mapping $\hat{S}(t) \leftarrow X^{nd}(t)$, where $\hat{S}(t) \in \{\hat{S}_1, \hat{S}_2, \cdots, \hat{S}_{nk}\}$. $\hat{S}(t)$ is normally an integer that represents the i^{th} operating state. In this chapter, the process feature vector is assumed to be context independent, i.e. each pattern represents a unique operating state (Srinivasan *et al.*, 2004a); so the mapping between $X^{nd}(t)$ and $\hat{S}(t)$ has to be one-to-one or many-to-one. Suppose a set of process vectors generated from a particular operating state $\hat{S}_{\hat{j}}$ from time *(t-q)* to *(t-p)* ($p < q$) is $D_{\hat{j}}^{d}$: $\{X^{d}(t-p), X^{d}(t-p-1), \cdots, X^{d}(t-q)\}$. The corresponding *feature vector set* is $D_{\hat{j}}^{nd}$: $\{X^{nd}(t-p), X^{nd}(t-p-1), \cdots, X^{nd}(t-q)\}$, where every element of $D_{\hat{j}}^{nd}$ is a snapshot of a moving data window of size *l* that is expected to contain sufficient temporal information for state identification.

Several methods for supervised classification have been proposed. Neural networks are widely used for data classification in industrial control due to their robustness and learning capacity as I reviewed in Chapter 2. However, the application of neural network becomes a challenge when the data dimensionality and pattern complexity is very high. This is well-known as "dimensionality curse". In this chapter, I propose two new neural network structures that overcome the curse of dimensionality and the complexities in temporal pattern recognition. The One-Variable-One-Network (OVON) architecture decomposes the problem in the variable dimension – uni-variate temporal patterns, called sub-states, are first identified in each variable; subsequently, the process state is inferred using the variable sub-states. The One-Class-One-Network (OCON) architecture uses a problem decomposition in the class dimension – the presence of specific temporal patterns in a high-dimensional space is first established using a bank of two-class classifiers. The process state is then identified by aggregating the results of the different classifiers. Both the architectures use a set of neural networks – in OVON there is one network for each variable, while in OCON, one network is used for each pattern class to be identified. In comparison to traditional monolithic neural networks, both the proposed architectures improve classification accuracy and minimize the training complexity. In addition, OVON is robust to sensor failures and OCON is well suited for addition of new pattern classes. The structures and training methodologies of the two architectures are presented and their performance compared against traditional neural networks using patterns arising during transitions in a simulated fluidized catalytic cracking unit.

This chapter is organized as follows. Two neural network-based systems that can simplify the classification problem are described in Section 4.2. In Section 4.3, I illustrate the performance of these systems on several case studies from a simulated fluidized catalytic cracking unit. Finally in Section 4.4, a comparison of the two methods is presented and conclusions are drawn.

4.2 Neural Classification Systems for Temporal Pattern Classification

4.2.1 One-Variable-One-Net (OVON) System

System Structure: A neural network architecture, called One-Variable-One-Net (OVON), which is based on the decomposition of the multivariate input into a set of uni-variate ones, is proposed in this section. In OVON, patterns in each variable are first identified based on spatial or temporal features; subsequently these are unified to identify the operating state.

In a multivariate process with *nk* operating states, each variable x_i can be considered to have nk_i sub-states. An example of temporal patterns from a bi-variate process is shown in Figure 4-1. Three distinct sub-states are defined for X_1 from an operations point-of-view ─ sub-state 1 during t=[1 35] when X_1 is at a quasi-steady state around 110, sub-state 2 during t=[36 125] when X_1 increases, and sub-state 3 during t=[126 270] when x_1 settles down to another steady state around 180. X_2 can also be simply classified into two sub-states ─ sub-state 1 during t=[1 125] when X_2 is in a steady state and sub-state 2 when X_2 drops to another steady state after a short transient. Here the sub-states are classified primarily based on the slope of the variables; however, in general, any criterion can be used to demarcate the states.

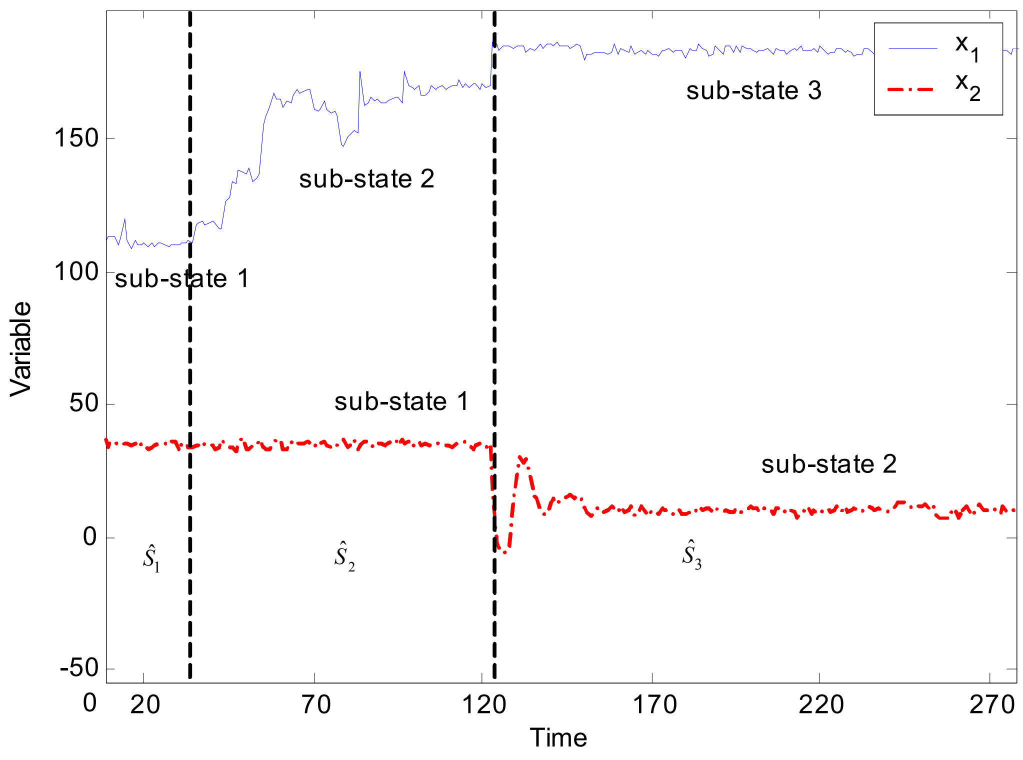

Figure 4-1: Example of sub-states

The OVON is based on demarcating the states at the variable-level. Figure 4-2 shows the typical structure of the OVON system. It comprises of two layers: the sub-state identification layer and the unification layer. The sub-state identification layer consists of *d* neural networks *[VN_1, VN_2, ...,VN_d]* corresponding to the *d* process variables. Each network VN_i identifies the sub-state of variable x_i. The input to VN_i is the value of feature vector $[x_i(t), x_i(t-1), \cdots, x_i(t-l_i)]$ that provides temporal information for pattern identification. The output of VN_i is an integer $S^{x_i}(t)$, the sub-state of variable X_i at time *t*. $S^{x_i}(t) \in [S_1^{x_i}, S_2^{x_i}, \cdots, S_{nk_i}^{x_i}]$. Network VN_i can be designed based on the characteristics of the variable and its features. A Multilayer Perceptron network (MLP), Radial Basis Function network (RBF), Elman recurrent network, Time Delay Neural Network (TDNN), or other neural networks can be used in the sub-state identification layer.

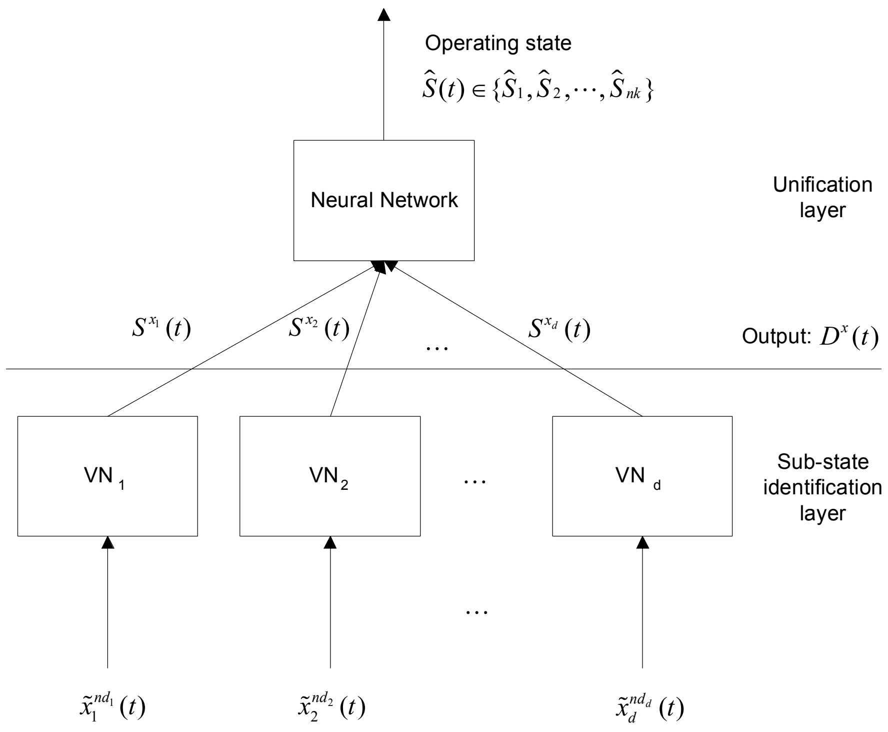

Figure 4-2: Structure of OVON

The unification layer is used to infer the operating state of the process based on the outputs of the sub-state layer networks. The operating state of the multivariate process is thus obtained by the mapping: $\hat{S}(t) \leftarrow D^x(t) : [S^{x_1}(t), S^{x_2}(t), \cdots, S^{x_d}(t)]$ where $S^{x_i}(t) \in [S_1^{x_i}, S_2^{x_i}, \cdots, S_{nk_i}^{x_i}]$ and $\hat{S}(t) \in [\hat{S}_1, \hat{S}_2, \cdots, \hat{S}_{nk}]$. Different combinations of the outputs from the sub-state identification layer indicate different process operation states. Note that, $nk \le \prod_{i=1}^{d} nk_i$. The unification layer can be any neural network. We have found RBFs to be efficient due to the local nature of their approximation and the ease of training them with integer inputs, as is the case here.

Training: The training of the OVON begins with the training of the sub-state identification networks. Process data are first normalized to [0, 1] based on the measurement range of the sensor to avoid the effect of different scales (Equation[3-6]). Each VN_i is trained individually using *a priori* knowledge of the nk_i operating states. The structure of each VN_i is defined based on the complexity of the patterns. When a variable contains intricate dynamics, temporal information is crucial for classification. A large lag l, more hidden neurons, and several hidden layers would be needed in such cases. Each VN_i is then trained to output:

$$S_j^{x_i} = f_{VN_i}(\tilde{X}_i^{nd_i}(t)) \quad if \quad \tilde{X}_i^{nd_i}(t) \in D_j^{nd_i} \quad \forall j \qquad [4\text{-}1]$$

where f_{VN_i} is the mapping embedded in VN_i. The training set $\{(D_1^{nd_1}, S_1^{x_i}), (D_2^{nd_2}, S_2^{x_i}), \cdots, (D_{nk_i}^{nd_i}, S_{nk_i}^{x_i})\}$ thus comprises of $D_j^{nd_i}$, the feature matrix with the values of delayed feature vectors $\tilde{x}_i^{nd_i}(t) : [\tilde{x}_i(t), \tilde{x}_i(t-1), \cdots, \tilde{x}_i(t-l_i)]$ corresponding to each sub-state $S_j^{x_i}$. The dimensionality nd_i of the input feature vector could be different for different VN_i since the input time delay l_i is based on the characteristics of the patterns in variable $\tilde{X}_i$. The training algorithm is selected based on the network structure, for example, back propagation for MLP, Elman network, and TDNN or linear least squares for RBF. Each VN_i in the sub-state identification layer is trained in this manner.

The unification layer is then trained using the outputs of the trained sub-state networks. The trained VN_i are simulated with the normalized training data $\tilde{X}^{nd_i}$. Each element in the d-dimensional output $D^x(t) : [S^{x_1}(t), S^{x_2}(t), \cdots, S^{x_d}(t)]$ is rounded to the nearest integer to form $\widehat{D}^x(t) : [\widehat{S}^{x_1}(t), \widehat{S}^{x_2}(t), \cdots, \widehat{S}^{x_d}(t)]$, where $\widehat{S}^{x_1}(t) = \text{round}(S^{x_1}(t))$. The purpose of rounding is to minimize the propagation of noise in the process data to the unification layer (See Section 4.4). The training set $\{(\widehat{D}_1^x, \widehat{S}_1), (\widehat{D}_2^x, \widehat{S}_2), \cdots, (\widehat{D}_{nk}^x, \widehat{S}_{nk})\}$

thus comprises of $\hat{D}_1^x$, the matrix with the output vectors from the sub-state identification layer corresponding to each sub-state

Advantages: The main advantage of the hierarchical structure in OVON is that temporal classification is reduced to a low *(l+1)*-dimensional space instead of the original, much higher $d \times (l+1)$ one. It also provides the flexibility to select a variable-specific value for l_i in the sub-state identification networks. This is in contrast to a monolithic neural network where *l* has to be assigned based on the longest time constant among the *d* variables in order to capture the process dynamics adequately. Consequently, more complex networks such as Elman networks can be used to improve sub-state identification accuracy if necessary while maintaining modest computational requirements. Another important advantage is that due to the independent and distributed nature of sub-state identification, this architecture is more robust to measurement faults. Sensor faults are common in chemical plants and can compromise state identification results. For proper functioning of a traditional monolithic network in such situations, the network would need to be completely retrained using only the correctly functioning sensors. In the case of OVON, since only one sub-state identification network would be affected for each measurement fault, the resulting errors are localized and can be eliminated by retraining only the unification layer. Thus minimizes the computational burden. It should be noted that if the state of the system becomes unobservable due the sensor failure, the results from any state identification approach, including the monolithic network and OVON, will be unreliable. Conversely, when a new sensor measurement becomes available, only one new sub-state identification network has to be developed and the unification layer retrained to take advantage of the additional information. This is also an easier task compared to retraining a monolithic network.

4.2.2 One-Class-One-Net System

System Structure: The One-Class-One-Net (OCON) architecture is based on the decomposition of the multi-pattern classification problem into a set of two-pattern classification sub-problems. This simplification strategy has been used in face recognition (Jou *et al.,* 1995), handwriting recognition (Baltzakis and Papamarkos, 2001; Jou *et al.,* 1992; Tsay *et al.,* 1992) and speech recognition (Sekhar and Yegnanarayana, 1996) and is reported to provide a good trade-off between classification accuracy and training complexity.

Figure 4-3 shows the typical structure of the OCON system. It comprises of two layers: the state identification layer and the regulator layer. When a single neural network is applied to a multi-pattern classification problem, it has to establish a number of up to *(nk-1)* surfaces to separate the input $\mathbb{R}^{nd}$ space into *nk* sub-spaces. This is a challenge especially when the patterns are complex and *nk* is large. The OCON overcomes this challenge by reducing it into simpler two-class separation sub-problems which require only one surface in $\mathbb{R}^{nd}$. As shown in Figure 4-3, the state identification layer of OCON consists of *nk* sub-nets *[CN$_1$, CN$_2$, ...,CN$_{nk}$]* corresponding to the *nk* operating states. All the networks share the same input variables $\tilde{X}^{nd_{\hat{j}}}(t)$. Each $CN_{\hat{j}}$ is trained so that it outputs one when the input vector $\tilde{X}^{nd_{\hat{j}}}(t)$ is generated from $\hat{S}_{\hat{j}}$ and zero when the input is from any other operating state. That is:

$$Z_{\hat{j}} = f_{CN_{\hat{j}}}(X^{nd_{\hat{j}}}(t)) = \begin{cases} 1 & if \quad \tilde{X}^{nd_{\hat{j}}}(t) \in D_{\hat{j}}^{nd_j} \\ 0 & if \quad \tilde{X}^{nd_{\hat{j}}}(t) \notin D_{\hat{j}}^{nd_j} \end{cases} \qquad 1 \le \hat{j} \le nk \qquad [4\text{-}2]$$

where $Z_{\hat{j}}$ is the output of sub-network $CN_{\hat{j}}$, $D_{\hat{j}}^{nd_{\hat{j}}}$ is the feature matrix containing delayed feature vectors from operating state $\hat{S}_{\hat{j}}$, and $f_{CN_{\hat{j}}}$ is the mapping embedded in the $\hat{j}$-th neural network $CN_{\hat{j}}$. The dimensions $nd_{\hat{j}}$ of the input feature vector can be modified to reflect the pattern complexity by varying the input time delay $l_{\hat{j}}$ for different $CN_{\hat{j}}$. MLP, RBF, Elman recurrent network, TDNN, or other neural networks can be used in the state identification layer.

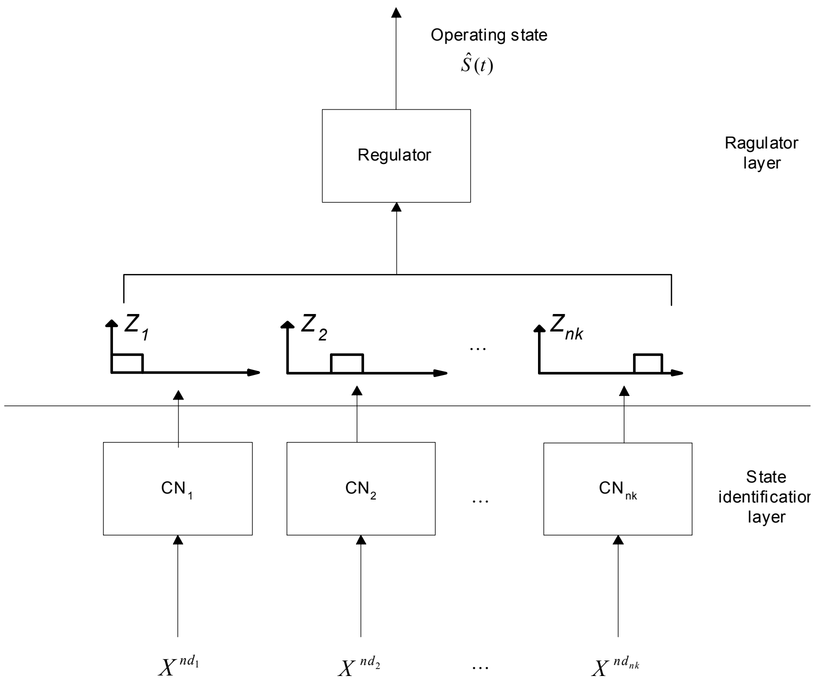

Figure 4-3: Structure of OCON

The role of the regulator layer is to infer the final operating state $\hat{S}(t)$ based on the outputs of the *nk* state identification sub-networks, i.e., $\hat{S}(t) = f_s(Z_1, Z_2, \cdots, Z_{nk})$. Several functions for f_s have been proposed in literature (Josef and Fabio, 2000). One common method is the winner-takes-all criterion where the network with the largest output is considered the winner.

$$\hat{S}(t) = \hat{S}_{\hat{j}},\ if\ \max(Z_1, Z_2, \cdots, Z_{nk}) = Z_{\hat{j}} \qquad [4\text{-}3]$$

An important shortcoming of the winner-takes-all criterion is that it requires that at any given instant t, there is only one sub-network whose output is close to one. The outputs of all other *(nk-1)* networks have to be close to zero. In chemical plants, due to process disturbances and operating condition deviations, sometimes there may be no clear winner, that is, more than one sub-network may give an output close to one, or conversely, no network may clearly dominate. To overcome this, we have complemented the winner-takes-all criterion with the following hold-strategy. In our hold-strategy, if the outputs of the state identification layer are ambiguous, the regulator would consider them as arising due to excess noise in the process data and would continue to output the previous operating state. The outputs of classifier system are regarded as ambiguous under two situations: (1) If all the networks give an output lower than a threshold θ_r or (2) if more than one network gives an output greater than θ_r.

$$\hat{S}(t)=\hat{S}(t-1),\ \ if\ \ \{Z_{\hat{j}}|_t<\theta_r,\forall\hat{j}\}\ or\ \{(Z_{\hat{i}},Z_{\hat{j}})>\theta_r,\hat{i}\neq\hat{j}\} \qquad [4\text{-}4]$$

Training Algorithms: Similar to the training of sub-networks in OVON, each $CN_{\hat{j}}$ is first trained individually using historical data. Normalized process data are used for this purpose. The training set is in the form of $\{(D_1^{nd_1},0),\cdots,(D_{\hat{j}}^{nd_{\hat{j}}},1),\cdots,(D_{nk}^{nd_{nk}},0)\}$. The determination of the structure of each $CN_{\hat{j}}$ and the corresponding time delay $l_{\hat{j}}$ is based on features of the process and the patterns indicating each operating state. The guidelines for structure selection described in Section 4.2.1 are also applicable here.

While $CN_{\hat{j}}$ can be any kind of network, RBF is attractive because of the localized nature of its activation function. The training of each $CN_{\hat{j}}$ enables it to recognize the following pattern:

$$z_{\hat{j}}=f_{CN_{\hat{j}}}(X^{nd_{\hat{j}}}(t))=\begin{cases}1 & if\ \ X^{nd_{\hat{j}}}(t)\in D_{\hat{j}}^{nd_j}\\ 0 & if\ \ X^{nd_{\hat{j}}}(t)\in D_{\hat{i}}^{nd_i}\end{cases} \qquad \hat{i}\neq\hat{j} \qquad [4\text{-}5]$$

Instead of using data from every state to train each $CN_{\hat{j}}$, to reduce the computational effort in training, only data from two operating states are used to train each $CN_{\hat{j}}$. Data from $\hat{S}_{\hat{j}}$ provide the positive examples from the state to be identified while data from any other state $\hat{S}_{\hat{i}}$ with $\hat{i} \neq \hat{j}$ provide the negative examples to enable the network to learn the decision boundary. The localized activation function of RBF can then adequately map other classes (even those that were not used for training) to zero. In this chapter, the Gaussian basis function has been used in all cases. It was observed that the choice of the basis function does not affect the RBF's performance (Chen, *et al.*, 1991). In contrast to the unification layer of OVON, the regulator in OCON does not need to be trained. The only parameter to be tuned — θ_r — can be set by experimental evaluation.

Advantage: The advantage of the OCON system is that multi-pattern classification has been decomposed into a number of simpler sub-problems. Each network deals with a two-class classification problem in the d-dimensional input space. This simplifies the classification task and improves the network's generalization capability. In addition, similar to OVON, the value of *l* does not need to be the same for each $CN_{\hat{j}}$. This makes it easier to design a sub-network to balance network structure complexity and pattern complexity. Another important advantage of the RBF-based OCON is that the local activation functions in $CN_{\hat{j}}$ are robust to new patterns. Thus, even patterns that are not used during training would be rejected by $CN_{\hat{j}}$ (through an output $z_{\hat{j}} \cong 0$) if they adequately differ from $\hat{S}_{\hat{j}}$. Chemical plants commonly operate in new states when new raw materials are to be processed or new product grades have to be manufactured. In such cases where the new pattern does not significantly overlap with the previous ones, all the $CN_{\hat{j}}$-*s* in the OCON do not need not to be retrained; only the training of one additional sub-network to identify the new state is necessary. These results in time-saving compared to monolithic networks which would need to be completely retrained when the set of states has to be modified. In the following Sections, we illustrate these features of OVON and OCON using a simulation of a large-scale chemical process.

4.3 Testing on Industrial-Scale FCC Unit

The classification-ability and performance of the OVON and OCON systems for industrial-scale applications was tested using data from a high-fidelity simulation of a refinery Fluidized Catalytic Cracking Unit (FCCU). The ShadowPlant that has been introduced in Chapter 3 is used here. As mentioned earlier, several runs of the startup were performed following the standard operating procedure. Two of them G_3 and G_4 are used here. In each section, superfluous variables were first eliminated to decrease the computational requirements.

Equation [3-6] is then applied to normalize both groups of the multivariate data to [0, 1]. A manual clustering based on operator logs is carried out to identify the target operating states for training. The boundary between contiguous operating states is recorded and used to define the training dataset.

Training criteria: The two network architectures have been trained and tested for all the sections in the FCCU. For comparison purpose, results from TDNN, RBF and Elman are also shown. In all cases, the training is stopped when any of the following conditions is satisfied: (1) 1000 epochs for sub-networks in OVON or OCON and 3000 epochs for traditional monolithic networks; (2) If the validation error is much larger than training error, which indicates overfitting, the number of epoch is reduced; (3) training mean square error ≤ 0.1; (4) training error does not decrease for a number of epochs.

Definition of Error: The training and testing performance of each network is evaluated using two metrics: (1) the classification error ε; (2) the training time.

$$\varepsilon = \frac{N_{mis}}{N_{total}} = \frac{\text{number of mis-classified validation samples}}{\text{total number of validation samples}} \qquad [4\text{-}6]$$

The training time (calculated on an Intel PIII workstation with 1.2GHz CPU and 1GB RAM) is the sum of training times of every individual sub-network for OVON and OCON. For detailed analysis, we categorize the classification error into two types:

- Type-1 errors which occur in the midst of an operating state. This type of error is largely due to the disturbances and noise during process operation.
- Type-2 errors which occur at the beginning or end of an operating state. This type of error is mainly caused by complex dynamics of the process not adequately captured by the network.

4.3.1 Air Pre-heater Section

The air pre-heater section consists of two sub-sections — air blower and pre-heater as shown in Figure 4-4. During startup, when the air blower reaches the steady state speed and flow rate, it is connected to the regenerator through the pre-heater. Air preheating is used only during startup for initially bringing up the cracker section.

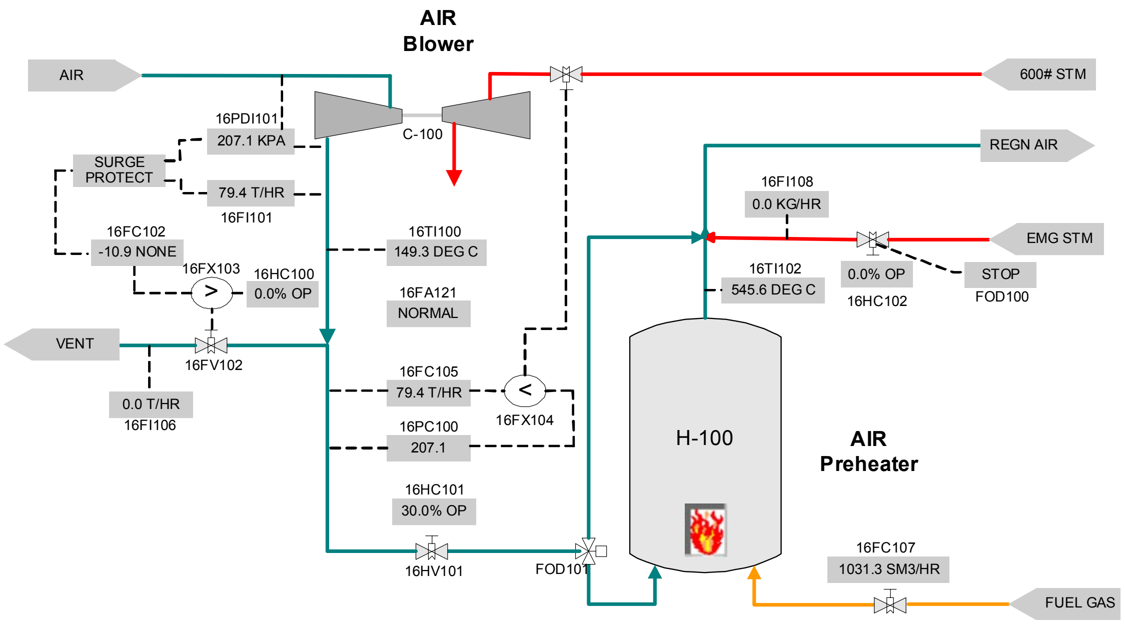

Figure 4-4: Overview of air pre-heater section

4.3.1.1 Pre-heater Sub-Section

In the pre-heater section, only 4 of 10 variables are relevant. As can be seen from Figure 4-5, the process can be clustered into four states. Therefore, both OVON and OCON have four sub-networks. In this case study, RBF is used as sub-nets of OVON and OCON.

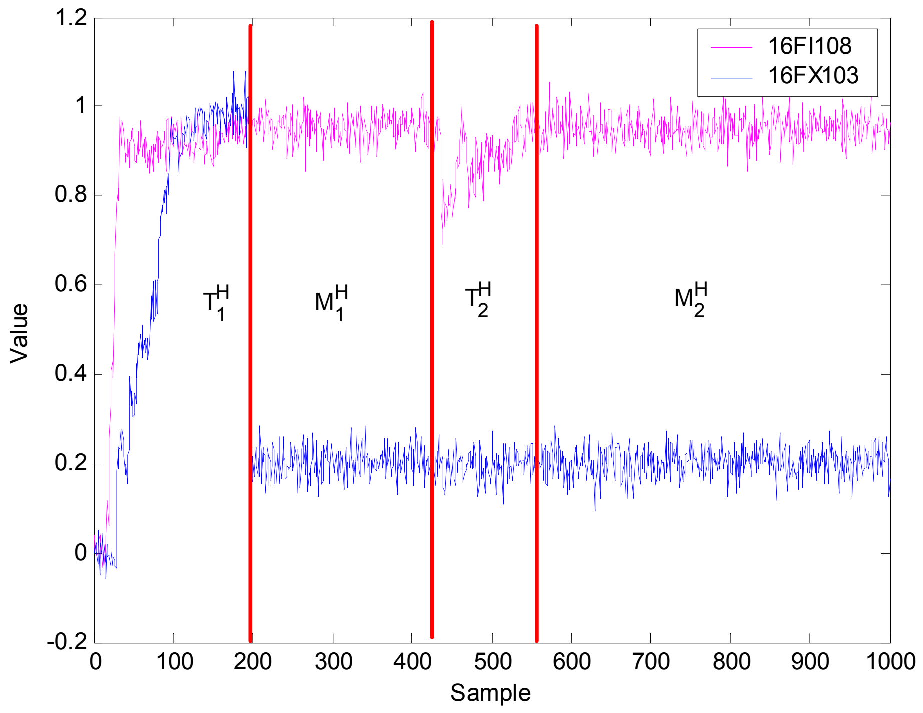

Figure 4-5: Evolution of two process variables of pre-heater

Testing results: The sub-state identification layer of OVON for this case study comprises of four separate RBFs. The classification errors calculated using Equation [4-6] are shown in TABLE 4-1. The overall validation error of this OVON is found to be 5.3%.

TABLE 4-1: OVON sub-state identification networks for pre-heater sub-section

Sub-network	**Variable name**	**No. of sub-states**	**RBF Structure**	**Training ε**	**Validation ε**
VN_1	16FC105	2	5-15-1	2.0%	2.0%
VN_2	16FC107	2	5-7-1	1.0%	1.8%
VN_3	16FI108	2	5-20-1	6.0%	7.0%
VN_4	16FX103	3	5-10-1	3.6%	3.7%

An OCON system was also constructed and tested. Four RBFs corresponding to the four states discussed above were initialized with the structures listed in TABLE 4-2. The final identification error of OCON is similar to OVON (about 5.3%).

TABLE 4-2: OCON sub-state identification networks for pre-heater sub-section

Sub-network	RBF Structure	Training ε	Validation ε
CN_1	20-10-1	1.3%	1.6%
CN_2	20-15-1	2.6%	1.7%
CN_3	20-10-1	3.0%	3.2%
CN_4	20-30-1	5.2%	4.4%

For comparison purposes, TDNN, RBF and Elman networks with structures 5-20-1, 5-30-4 and 1-20-1 respectively were also trained and tested. A summary of their performances is shown in TABLE 4-3. TDNN and Elman networks have higher validation errors which are above 10%. In addition, Elman network has a very long training time compared all of other networks. The classification error of a single RBF is about 6% which is a little higher than OVON and OCON. In this case study, OVON and OCON has the best performance in terms of classification error. However, as shown in the table, it is not superior much to a single RBF. In the following case study, more complicated cases will be used to test the performance of the networks.

TABLE 4-3: Performances of neural networks for pre-heater subsection

	Training Time (s)	Validation ε
OVON	Sub-networks: 20	5.3%
	Unification layer: 2.5	
OCON	25	Winner-takes-all: 5.3%
		Regulator: 5.1%
TDNN	117	13.7%
RBF	15	6.0%
Elman Network	1323	10.5%

4.3.1.2 Air Blower Sub-Section

Seven measured variables are used for pattern classification in the air blower sub-section. Figure 4-6 shows the evolution of two important variables. As seen in the figure, the process data can be divided into six segments ─ three transitions (T_1^B, T_2^B, and T_3^B) and three modes (M_1^B, M_2^B, and M_3^B). T_1^B is caused by the regenerator warm-up action, T_2^B by the introduction of the feed and T_3^B by a sudden surge. M_1^B is intermediate mode during which the operator stabilizes the process. M_2^B and M_3^B are the steady states. For operation purposes, T_1^B, T_2^B, and T_3^B need not be distinguished and can be considered to be transition states. M_2^B and M_3^B are also similar. Thus, three operating state $\hat{S}_1$, $\hat{S}_2$ and $\hat{S}_3$ are to be identified by the networks as shown in Figure 4-6. The beginning and ending time of each state is used as *a priori* knowledge to train the neural networks.

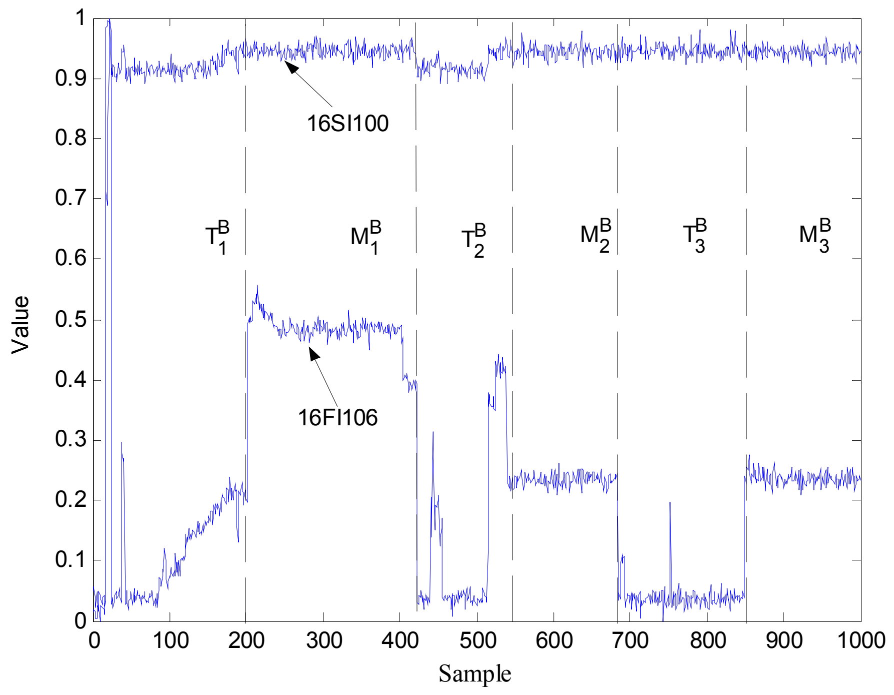

Figure 4-6: Evolution of two process variables of air blower sub-section of *G₃*

TABLE 4-4: Operating states of air blower sub-section of G_3

Operating state	Sample range
$\hat{S}_1$	[1 24]
$\hat{S}_2$	[241 404]
$\hat{S}_1$	[405 539]
$\hat{S}_3$	[540 684]
$\hat{S}_1$	[685 847]
$\hat{S}_3$	[848 1000]

Testing Results: The sub-state identification layer of OVON for this case study comprises of seven separate TDNNs. The structure of each sub-network is designed based on the complexity of the patterns in the input variable. The classification errors calculated using Equation [4-6] are shown in TABLE 4-5. The largest classification error occurs in 16FI106 because its evolution has overlapping patterns (for example, at ends of T_1^B and M_2^B, and ends of T_2^B and M_1^B) as can be seen in Figure 4-6. Another sub-network that gives large classification error is *VN₃*. As seen in Figure 4-7, the difference in 16PDI101 between $S_1^{x_3}$ to $S_2^{x_3}$ is small, subsequently, the sub-network *VN₃* generates large classification error. Similar situations arise in *VN₅* and *VN₆* as well. A RBF with 10 hidden neurons was then trained (using the *newrb* function in Matlab's neural network toolbox (Mathworks, 2002)) as the unification layer to map the sub-states of the seven variables to the process operating state. The rounded outputs from the sub-state identification networks are used as training inputs. The training target consist of three states $\hat{S}_1$, $\hat{S}_2$ and $\hat{S}_3$. Finally, the trained sub-state identification networks are integrated with the trained unification layer RBF to construct the hierarchical OVON. This combined network now takes the process variables as input and outputs the tag of process operating state. The validation error of this OVON is found to be 6.3%.

TABLE 4-5: OVON sub-state identification networks for air blower sub-section

Sub-network	Variable name	No. of sub-states	TDNN Structure	Training ε	Validation ε
VN_1	16SI100	2	7-10-1	0.07%	0.05%
VN_2	16FI100	2	7-10-1	0.07%	0.05%
VN_3	16PDI101	4	10-25-1	8.2%	8.4%
VN_4	16FI101	2	5-10-1	1.4%	1.5%
VN_5	16FC102	4	10-20-1	6%	6%
VN_6	16TI100	4	10-20-1	7%	6.8%
VN_7	16FI106	6	10-10-1	13%	13.2%

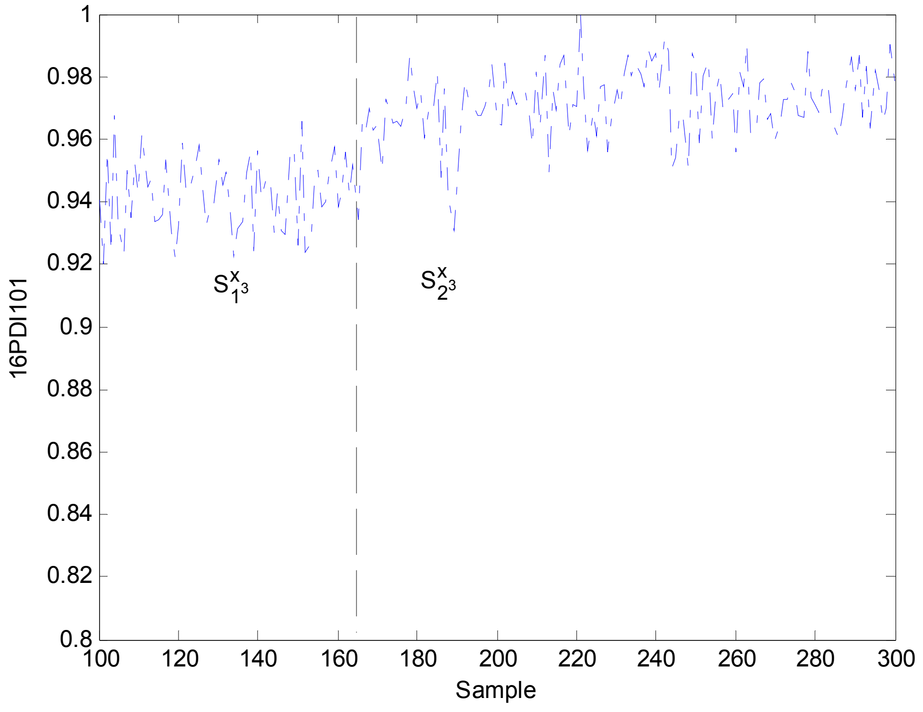

Figure 4-7: Sub-state of 16PDI101 of air blower sub-section of G_3

An OCON system was also constructed and tested. Three RBF corresponding to the three states discussed above were initialized with the structures listed in TABLE 4-6. The network $CN_{\hat{j}}$ with output $z_{\hat{j}}$ is trained to identify the $\hat{j}$-th operating state, where $\hat{j} \in [1\ 2\ 3]$. The test results of three state identification networks are shown in

Figure 4-8 and the classification errors listed in TABLE 4-6. As can be seen from the figure, each CN_j can detect its corresponding state correctly and thus has a low classification error. The final identification error of OCON is about 5.5%.

TABLE 4-6: OCON state identification networks for air blower sub-section

Sub-network	Network Structure	Training ε	Validation ε
CN_1	70-20-1	5.4%	5.6%
CN_2	70-5-1	2.3%	2.3%
CN_3	70-10-1	4.6%	3.8%

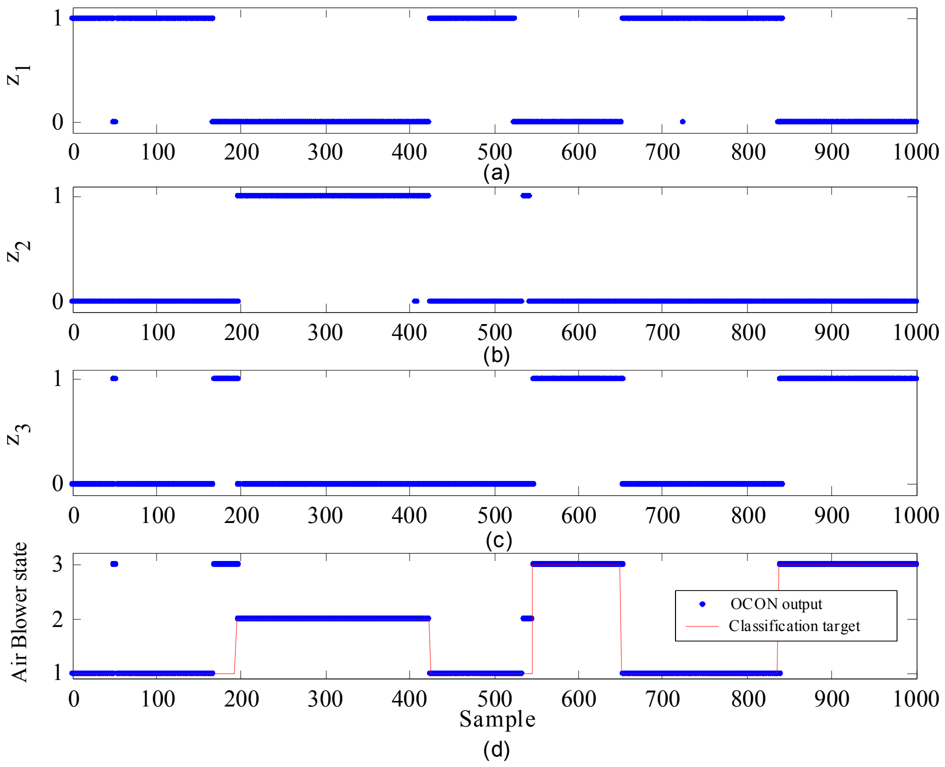

Figure 4-8: Output of CN_1 (b) Output of CN_2 (c) Output CN_3 (d) Output of OVON for air blower sub-section on G_4

For comparison purposes, TDNN, RBF and Elman networks with structures 70-70-3, 70-10-3 and 7-20-3 respectively were also trained and tested. A summary of their performances is shown in TABLE 4-7. Both OVON and OCON perform better than any single neural network with errors that are about 34% less than that of Elman which is the best performer among the traditional networks. The training of Elman network takes 19 times longer than OVON and illustrates the complexity of the problem. In addition, while the monolithic RBF and the OCON use similar network structures, OCON performs better due to the different training philosophy. The OVON and OCON thus achieve better performance through the decomposition of the original complex problem into a set of simpler sub-problems.

TABLE 4-7: Performances of neural networks for air blower sub-section

	Training Time (s)	Validation ε
OVON	Sub-networks: 185	6.3%
	Unification layer: 2.85	
OCON	29.5	Winner-takes-all: 5.5%
		Regulator: 5.4%
TDNN	476	12.7%
RBF	30	12.8%
Elman Network	3517	9.6%

Tuning Parameters: The effect of the tuning parameters on the classification results is discussed here. The time lag l_i (for OVON) and l_j (for OCON) are selected based on process dynamics. If l is too small, operating states with long periods of dynamics may be misclassified. In contrast, if l is too long, it will increase the computational burden. Also, there will be a delay in identifying short states. For OVON, the selection is based on the time constant of the individual variables. However the performance is relatively robust within a broad range of values. For example, during training of VN_7, the validation error is less than 15% for $l_i \in [8\ 15]$. For OCON, the selection of l is based on the dynamics of the operating states. In this case study since the process displays a long ramping pattern in T_1^B, we specified

$l_j = 10$. A short l will not be able to detect this pattern. For example, if $l_j < 8$, the validation error for T_1^B jumps to 6.3%.

The regulator is quite robust to the value of θ_r since the output of RBF is only a binary. In the above case study, varying θ_r from 0.3 to 0.8 does not have any significant effect on performance. We have found θ_r=0.5 to be a suitable choice for most cases.

Effectiveness of OVON unification layer: The unification layer of OVON has the ability to filter out classification errors of individual sub-networks and improve the final classification accuracy. Figure 4-9 shows the rounded output of the sub-network *VN₅* (16FC102) and the unification layer. In REGION A, while the sub-network has Type-I classification errors, there is no corresponding classification error from the unification layer. The errors from the sub-network are filtered out by the unification layer because of the localized nature of the RBF. Suppose, during the training of the unification layer, the value of output vector $[z_1, z_2, \cdots, z_{nk}]$ from the sub-state identification layer is trained to be mapped to the operating state $\hat{S}_{\hat{j}}$. During validation, if the Euclidean distance between the output vector of the sub-state identification layer and $[z_1, z_2, \cdots, z_{nk}]$ is within the width of the Gaussian function, the unification layer will still map it to the state $\hat{S}_{\hat{j}}$ even if a few sub-networks generate classification errors. The unification layer thus accounts for sub-state identification errors and identifies the state correctly. If many sub-networks mis-classify the same sub-state, the unification layer will fail to filter the combined errors as illustrated in REGION B of Figure 4-9.

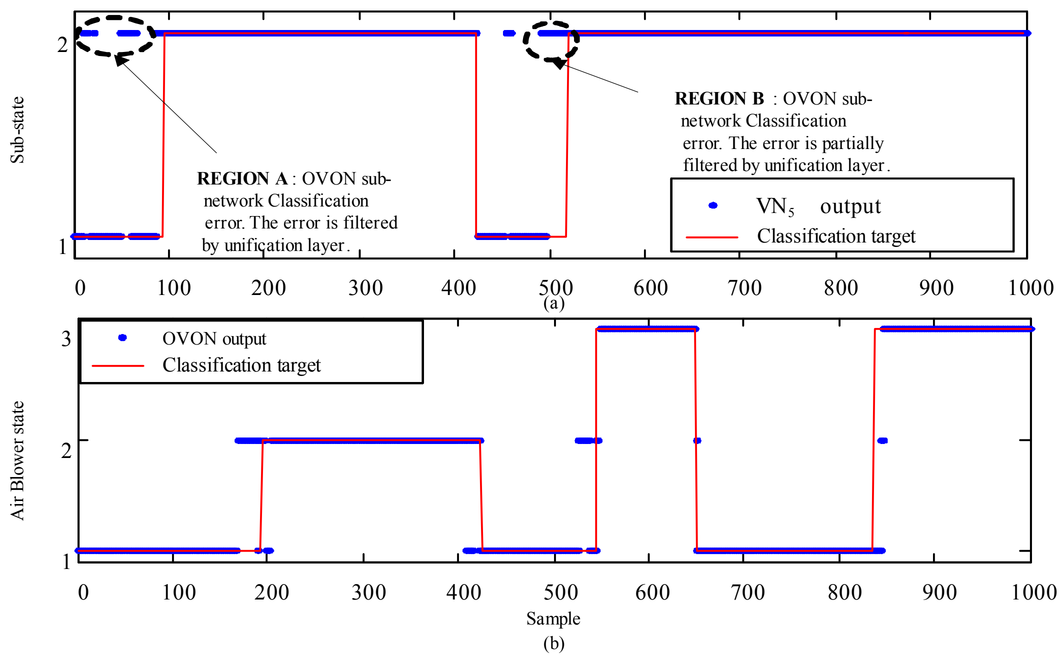

Figure 4-9: (a) Output of *VN$_5$* (16FC102) (b) Output of OVON for air blower sub-section

Effectiveness of OCON regulator layer: A similar property is exhibited by the regulator layer in OCON. To illustrate the robustness of the regulator, a disturbance is added to samples of G_4 from 300 to 350 as shown in Figure 4-10. When the OCON trained on G_3 is tested on this new dataset, classification errors are generated by the sub-networks as shown in Figure 4-11. During the period of disturbance (REGION C), CN_2 fails to identify the pattern and incorrectly outputs zero; the other sub-networks are not affected and correctly output zero. In this situation, there is no clear outcome and the winner-takes-all strategy could give a wrong result. The regulator strategy however would reject the disturbance and correctly conclude that the current operating state is $\hat{S}_2$. REGION D (see Figure 4-11) illustrates another situation where the winner-takes-all strategy erroneously identifies the operating state as $\hat{S}_3$ since $Z_3 > Z_2$. In contrast, the regulator correctly identifies the operating state as $\hat{S}_2$. These illustrate that the regulator can reject Type-1 errors and is superior to winner-takes-all strategy in this situation. Regulator can also correct Type-2 errors caused by process transitions as illustrated in Figure 4-12. In REGION E, during the process state change from $\hat{S}_2$ to $\hat{S}_1$, the three outputs z_1, z_2 and z_3 are lower than θ_r (=0.5) and no sub-network dominates. In this case, the regulator filters the false negative of CN_2 and correctly concludes that the current operating state is $\hat{S}_2$ (previous operating state). Thus, Type-1 and Type-2 errors can be corrected by the regulator layer of OCON. However, the regulator cannot always improve the final OCON performance. Investigating the Type-2 errors in REGION F of Figure 4-13, during the process state change from $\hat{S}_1$ to $\hat{S}_2$, CN_1 gives a false negative while CN_3 gives a false positive. Because CN_3 is the only network which gives an output of one, the regulator incorrectly concludes that the current operating state is $\hat{S}_3$. Thus the regulator fails when a false negative in a state-identification network occurs in conjunction with *exactly* one false-positive from another network.

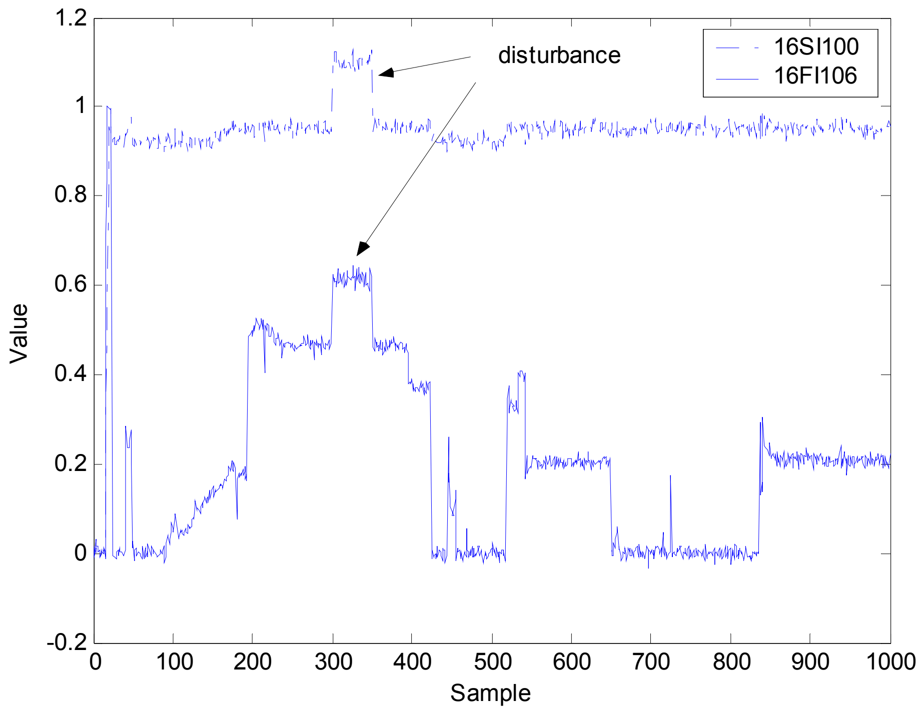

Figure 4-10: Two variables of air blower sub-section on G_3 with disturbance

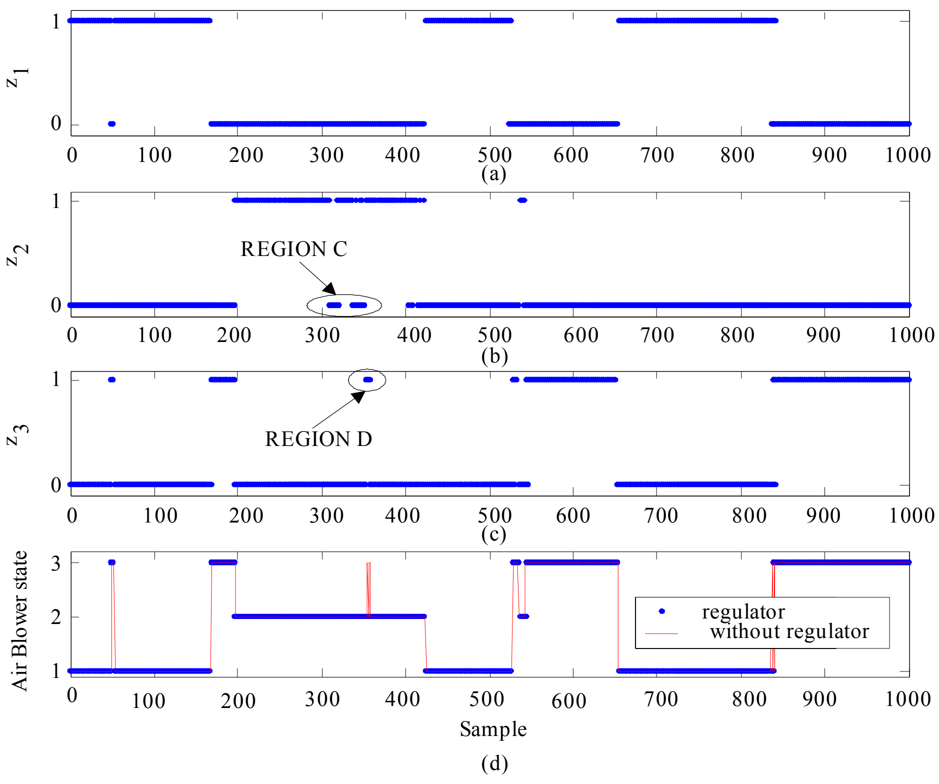

Figure 4-11: (a) Output of CN1 (b) Output of CN2 (c) Output CN3 (d) Output of OCON for air blower sub-section on disturbance-added dataset

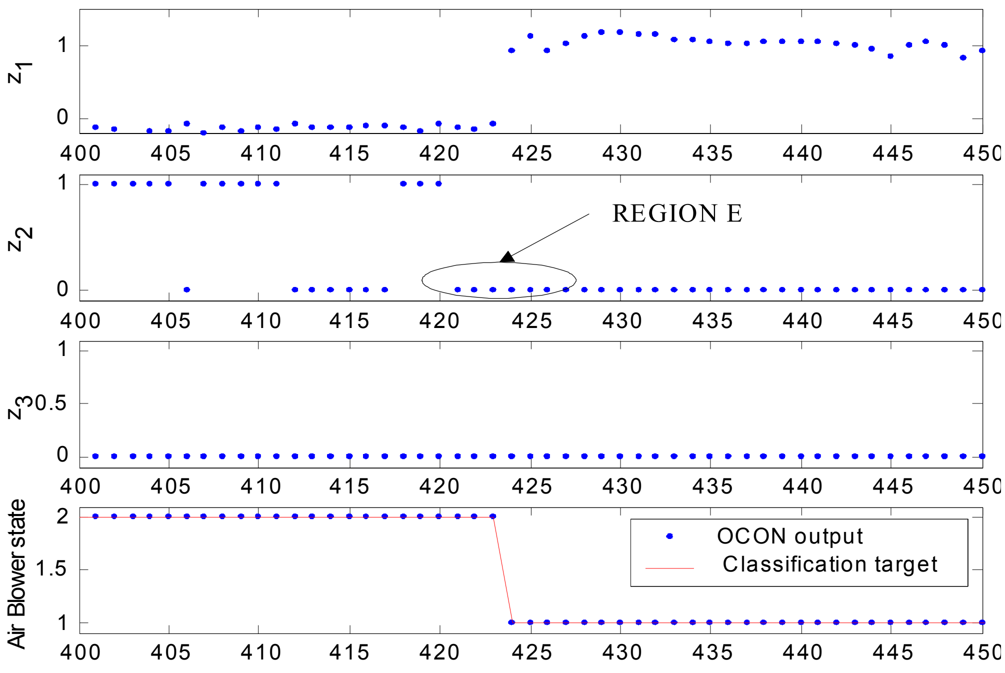

Figure 4-12: Output of sub-networks and regulator during state change from $\hat{S}_2$ to $\hat{S}_1$ for air blower sub-section

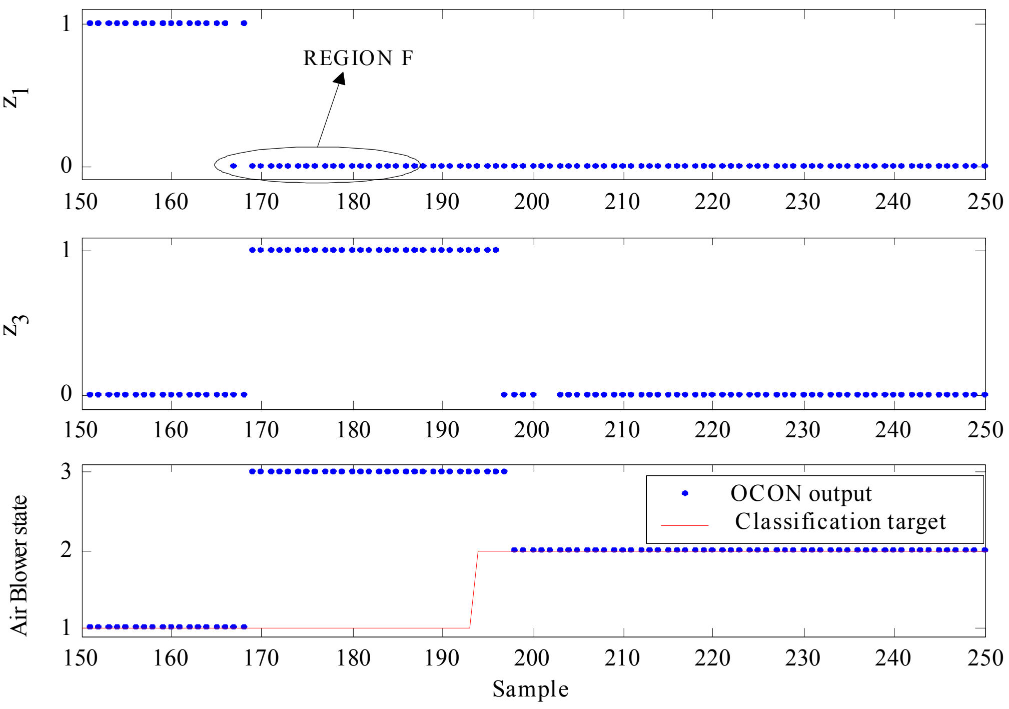

Figure 4-13: Output of sub-networks and regulator during state change from $\hat{S}_1$ to $\hat{S}_2$ for air blower sub-section

4.3.2 Regenerator Section

Figure 4-14 shows the schematic of the regenerator section. The regenerator is a large cylindrical vessel where coke deposited on the catalyst surface as the result of cracking reactions is burned off using air. Eighteen variables of the section are used for state identification. Figure 4-15 shows the profile of two variables. As seen in the figure, there are two operating modes M_1^R and M_2^R, and two transitions T_1^R and T_2^R. T_1^R corresponds to the regenerator startup operation and includes warming up the regenerator using hot air, introducing diesel for further temperature increase, and catalyst loading. Subsequently, during T_2^R, feed is introduced to the regenerator and reaction starts. M_1^R is the intermediate mode during which the operator stabilizes the regenerator and starts up other sections of the FCCU. M_2^R is the final state of the section. Accordingly, for this section four states $\hat{S}_1$, $\hat{S}_2$, $\hat{S}_3$ and $\hat{S}_4$ are to be identified.

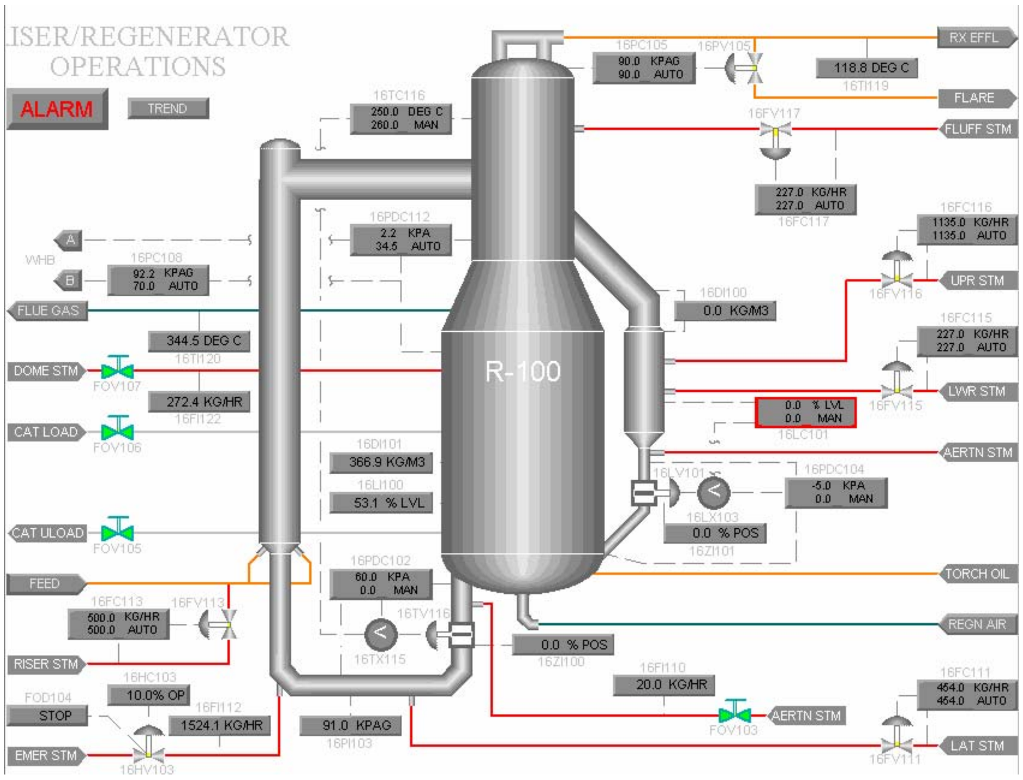

Figure 4-14: Overview of regenerator section

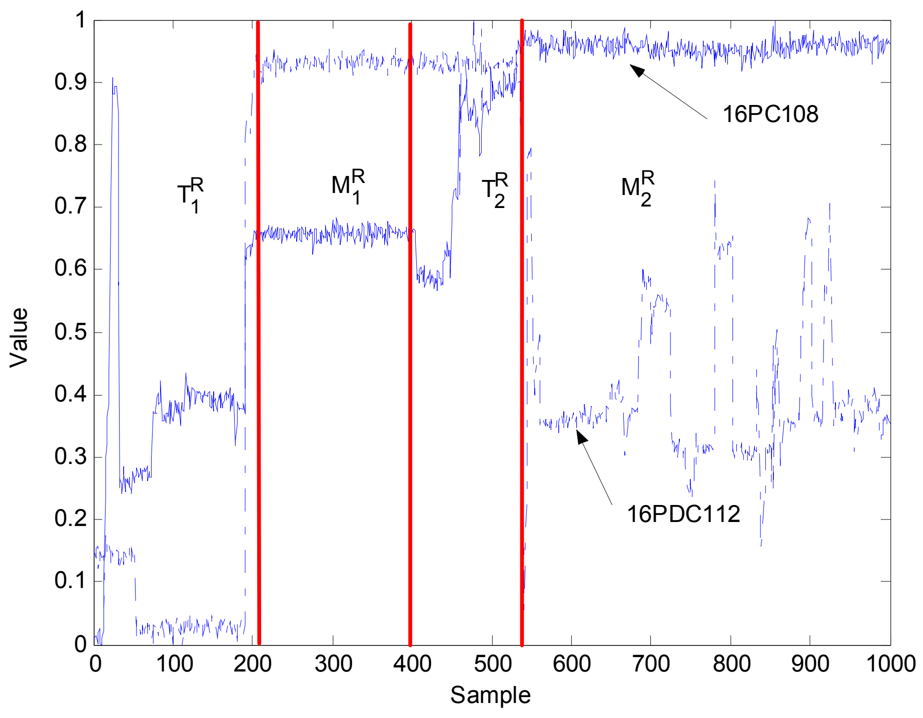

Figure 4-15: Evolution of two process variables of regenerator section of G_2

The OVON and OCON structures were trained as summarized in TABLE 4-8 and TABLE 4-9 respectively. For comparison purposes, TDNN, RBF and Elman networks with structures 180-30-4, 180-10-4, and 18-30-4 respectively were also trained. Their testing performances are summarized in TABLE 4-10. OCON and OVON have the least classification errors among the five structures. OVON requires a larger training time since there are 18 variables in this section. In this case, boundaries among two operating states are comparatively clearer and traditional network structures yield good classification performances. Although the differences among the performances are not significant, some improvement is still achieved by using OVON and OCON. In addition, the performance of OCON with regulator is substantially better than any other networks.

TABLE 4-8: OVON sub-state identification networks for regenerator section (18 variables; 4 states)

Sub-network	Variable	No. of sub-states	TDNN Structure	Training ε	Validation ε
VN_1	16PC108	4	9-10-1	3.7%	3.5%
VN_2	16PDC112	3	6-5-1	1.3%	1.4%
VN_3	16TC116	4	6-5-1	0.9%	1.0%
VN_4	16FI112	2	6-5-1	0%	0%
VN_5	16LI100	6	6-5-1	0.9%	0.8%
VN_6	16DI101	3	8-10-1	3.6%	3.9%
VN_7	16PDC102	2	6-3-1	0.4%	0.5%
VN_8	16PC105	3	6-10-1	3.8%	4.0%
VN_9	16TI119	4	8-15-1	3.9%	3.91%
VN_{10}	16PDC104	4	6-10-1	2%	2.1%
VN_{11}	16FC117	2	5-3-1	0.3%	0.29%
VN_{12}	16FC116	2	6-3-1	0.1%	0.1%
VN_{13}	16FC115	2	6-3-1	0.3%	0.3%
VN_{14}	16PDC104	3	6-10-1	3.4%	3.3%
VN_{15}	16FC111	2	6-3-1	0.5%	0.51%
VN_{16}	16FI110	2	6-3-1	0.3%	0.3%
VN_{17}	16FC109	4	9-10-1	1%	1.3%

VN_{18}	16TI114	5	6-5-1	0.6%	0.7%

TABLE 4-9: OCON state identification networks for regenerator section (18 variables; 4 states)

Sub-network	Network Structure	Training ε	Validation ε
CN_1	180-20-1	1.4%	1.5%
CN_2	180-20-1	0.7%	0.9%
CN_3	180-20-1	0.9%	1.1%
CN_4	180-20-1	0.5%	0.5%

TABLE 4-10: Performances of neural networks for regenerator section (18 variables; 4 states)

	Training Time (s)	Validation ε
OVON	Sub-networks: 280	1.0%
	Unification layer: 10	
OCON	34	Winner-takes-all: 1.0%
		Regulator: 0.2%
TDNN	65	1.3%
RBF	24	2.6%
Elman Network	1157	1.5%

Sensors faults: As discussed above, in chemical processes, sensors may become unavailable due to faults. When measurement from a faulty sensor is used for state identification classifier, the classifier result will be erroneous. Figure 4-16 shows an example of the degradation of the operating state identification by RBF when the sensors for 16FC109 and 16TI114 in regenerator section are stuck. Even when all the other sensors provide accurate measurements, the classifier performance degrades completely and the classification error increases (6% for TDNN, 10% for Elman network, and 35% for RBF). Similar results can be observed from other sensor faults as well. To overcome this, the networks will have to be retrained using the remaining available measurements. The retraining usually takes as long as the original training. However, for OVON, since the sub-networks are specific to individual variables, the system can be used without retraining the sub-state identification layer. Previously trained sub-networks can be used directly and only the unification layer needs to be retrained by using of the other variables. The performance of the retrained networks is shown in TABLE 4-11. While there is no significant change in the accuracy, the retraining of OVON takes only 8s which is 68% lesser that the next best network. This is an important advantage of OVON.

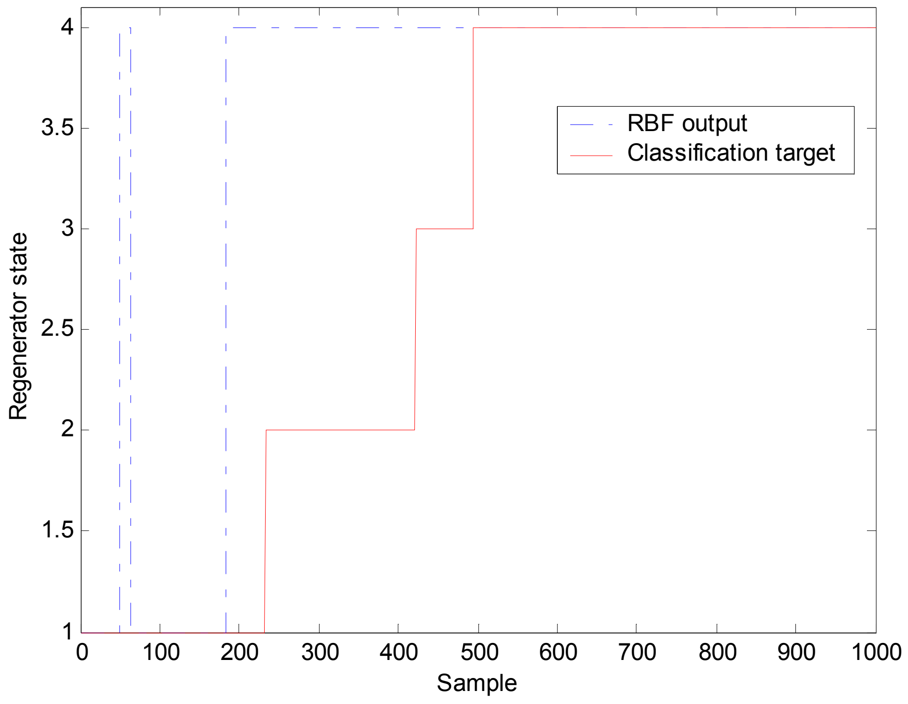

Figure 4-16: Operating state identification results of RBF for regenerator section with faulty sensors

TABLE 4-11: Performances of neural networks for regenerator section (16 variables; 4 states)

	Re-Training Time (s)	**Validation ε**
OVON	Sub-networks: 0	1.0%
	Unification layer: 8	
OCON	35	Winner-takes-all: 0.9%

		Regulator: 0.1%
TDNN	63	1.5%
RBF	25	2.3%
Elman Network	1160	1.4%

New Patterns: The OCON architecture is beneficial when additional states (that is new patterns) need to be introduced such as when new raw materials are to be processed or new product grades have to be manufactured. In such situations, OVON, TDNN, RBF, and Elman networks have to be retrained to recognize the new states. However, for OCON, the existing sub-networks need not to be retrained since their output will be close to zero even when data from the new state is input because of the localized activation function of the RBF networks used in the state identification layer. Only one additional RBF needs to be trained to recognize the new operating state as illustrated next. Suppose in the regenerator section, a new state resulting from the shutdown transition T_3^R (see Figure 4-17) has to be included. For this purpose, the traditional networks and OVON are completely retrained. In contrast, in OCON, a new state identification RBF with structure 180-20-1 is trained using data from M_2^R and T_3^R and included in the previously trained structure. As seen in TABLE 4-12, while there is no significant change in the accuracy, the retraining of OCON takes only 10s which is substantially smaller than the other networks.

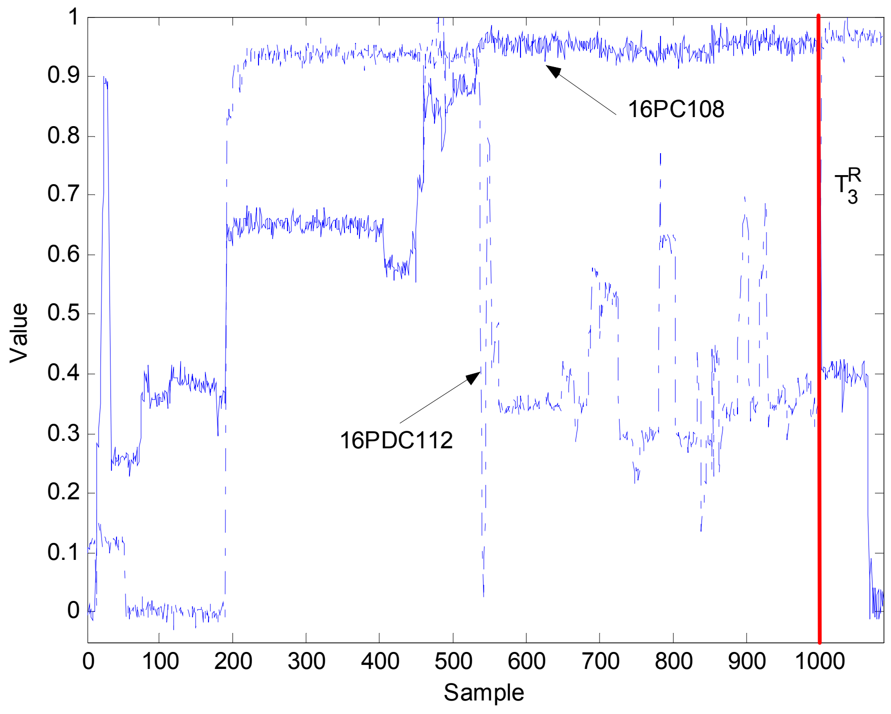

Figure 4-17: Evolution of two process variables of regenerator section with new operating state

TABLE 4-12: Performances of neural networks for regenerator section (18 variables; 5 states)

	Re-Training Time (s)	**Validation ε**
OVON	Sub-networks: 285	1.2%
	Unification layer: 10	
OCON	10	Winner-takes-all: 0.5%
		Regulator: 0.1%
TDNN	65	1.6%
RBF	30	1.9%
Elman Network	1167	1.5%

OVON and OCON are also tested for other sections of the ShadowPlant. They are briefly presented in the following sections.

4.3.3 Fractionator Section

Figure 4-18 shows the schematic of the Fractionator section. Twelve variables of the section are used for state identification. Figure 4-19 shows the profile of two variables. As seen in the figure, there are five segments — two transitions (T_1^F and T_2^F)

and three modes (M_1^F, M_2^F and M_3^F). However, T_1^F and T_2^F are similar, also are M_2^F and M_3^F. Therefore, there are only three distinguishable states.

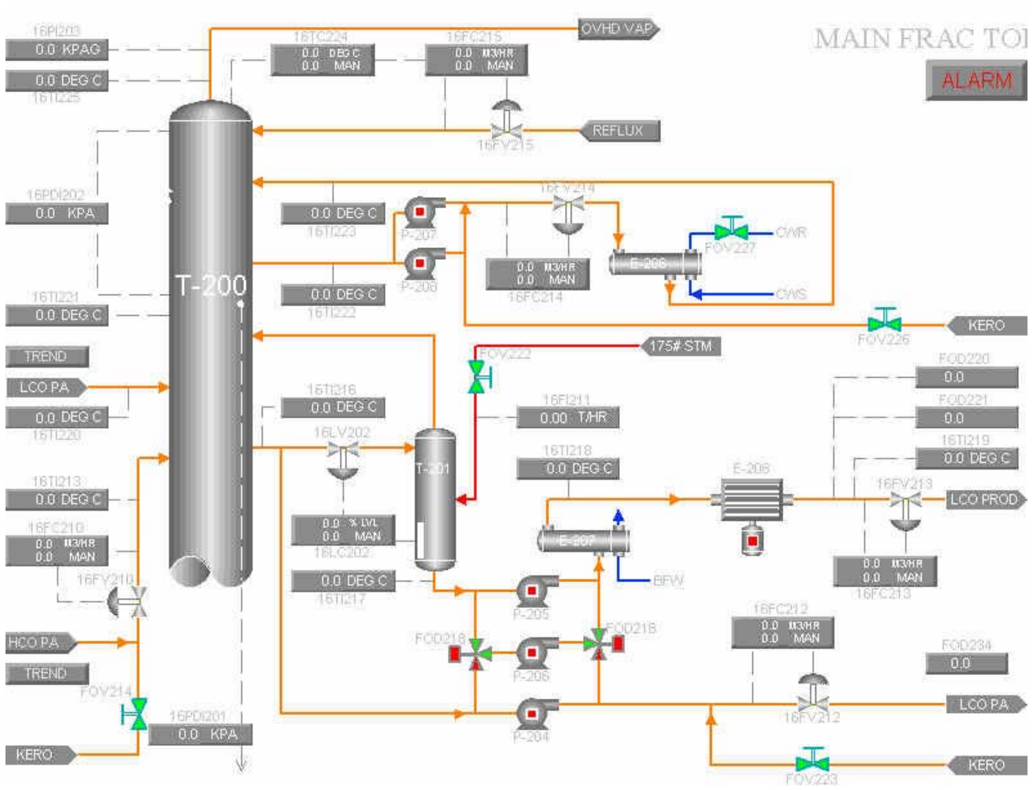

Figure 4-18: Overview of Fractionator section

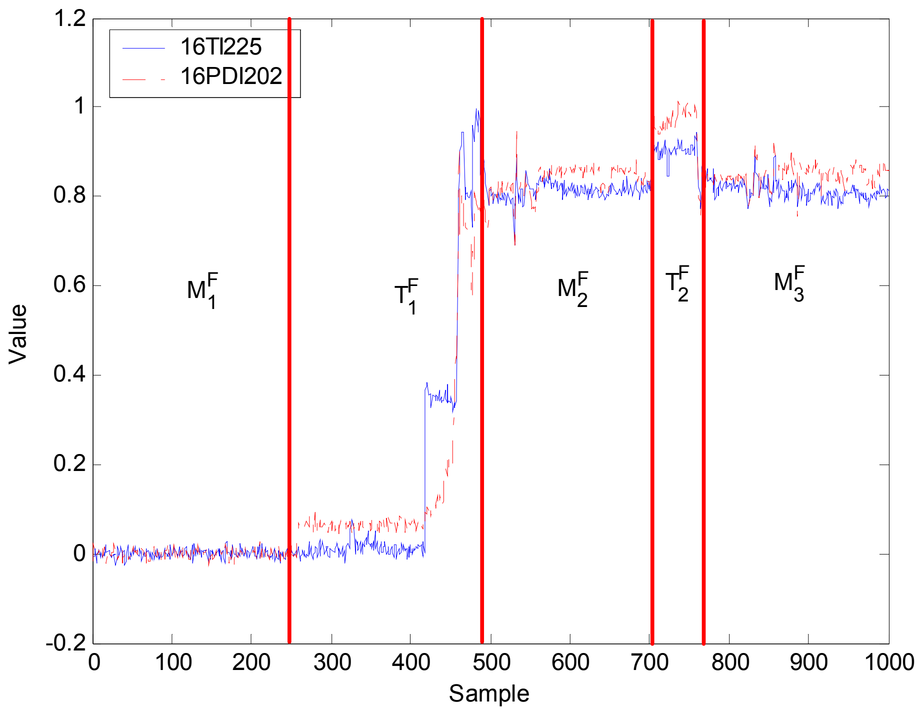

Figure 4-19: Evolution of two process variables of Fractionator section

Testing results: The sub-state identification layer for this case study comprises of 12 separate TDNNs. The classification errors are shown in TABLE 4-13. The overall validation error of this OVON is found to be 6.5%.

TABLE 4-13: OVON sub-state identification networks for Fractionator section

Sub-network	Variable name	No. of sub-states	TDNN Structure	Training ε	Validation ε
VN_1	16TI225	3	60-10-1	2.4%	2.6%
VN_2	16PDI202	5	60-10-1	1.5%	1.5%
VN_3	16TI221	4	60-10-1	2.7%	2.6%
VN_4	16FC215	4	24-10-1	0.4%	0.7%
VN_5	16TI216	3	48-10-1	1.8%	1.7%
VN_6	16FI211	2	12-5-1	0%	0.01%
VN_7	16FC212	3	12-5-1	0%	0%
VN_8	16TI214	3	24-5-1	0.1%	0.15%
VN_9	16TI206	3	24-5-1	1.0%	1.1%
VN_{10}	16FC207	3	84-10-1	1.7%	1.5%
VN_{11}	16TI212	2	36-10-1	0.8%	0.9%
VN_{12}	16FC221	4	36-10-1	1.5%	1.7%

An OCON system was also constructed and tested. Three RBFs corresponding to the three states discussed above were initialized with the structures listed in TABLE 4-14. The final identification error of OCON is about 5.4%.

TABLE 4-14: OCON sub-state identification networks for Fractionator section

Sub-network	RBF Structure	Training ε	Validation ε
CN_1	20-10-1	2.3%	2.6%
CN_2	20-15-1	4.7%	4.3%
CN_3	20-10-1	1.0%	1.2%

For comparison purposes, TDNN, RBF and Elman networks with structures 60-50-1, 60-20-1 and 12-50-1 respectively were also trained and tested. A summary of their performances is shown in TABLE 4-15. As can be seen from the table, OVON and OCON has the highest classification accuracy.

TABLE 4-15: Performances of neural networks for Fractionator section

	Training Time (s)	Validation ε
OVON	Sub-networks: 196	5.3%
	Unification layer: 4	
OCON	25	Winner-takes-all: 5.4%
		Regulator: 5.2%
TDNN	268	6.8%
RBF	8	7.1%
Elman Network	3658	8.3%

4.3.4 Waste Heat Boiler Section

Figure 4-18 shows the schematic of the waste heat boiler section. Five variables of the section are used for state identification. Figure 4-19 shows the profile of two variables. As seen in the figure, there are five segments ─ two transitions (T_1^S and T_2^S) and three modes (M_1^S, M_2^S and M_3^S). However, T_1^S and M_2^S, M_1^S and M_3^S are indistinguishable. Therefore, 3 operating states are identified.

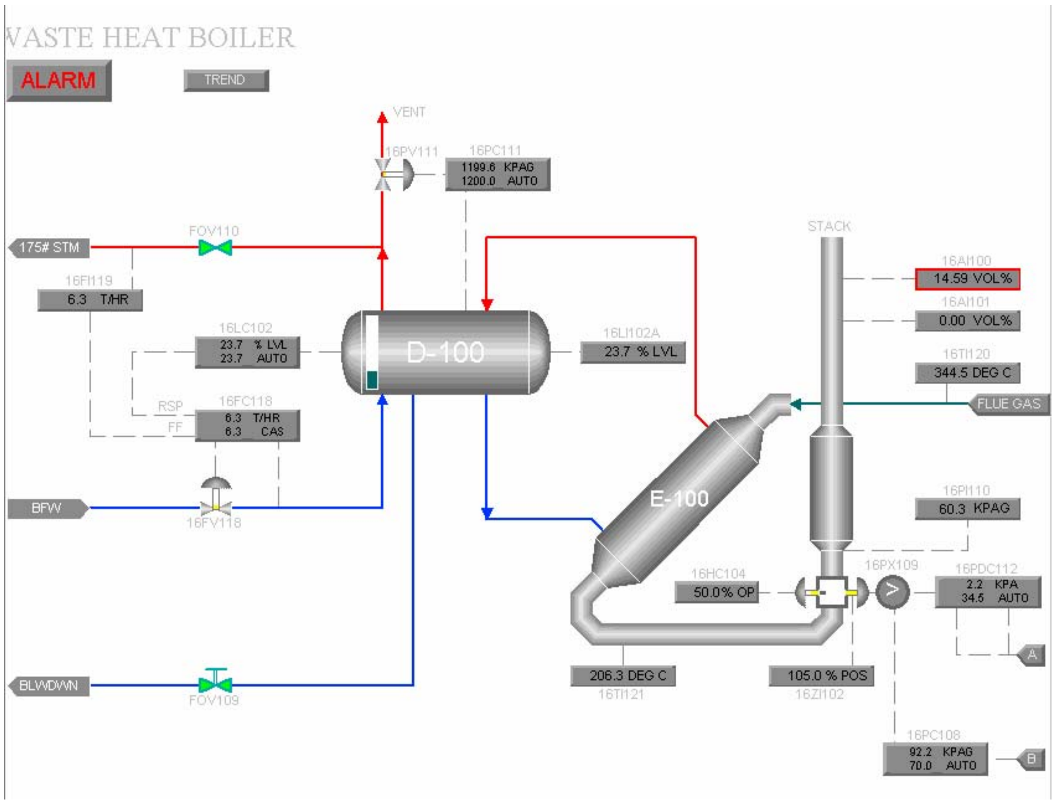

Figure 4-20: Overview of waste heat boiler section

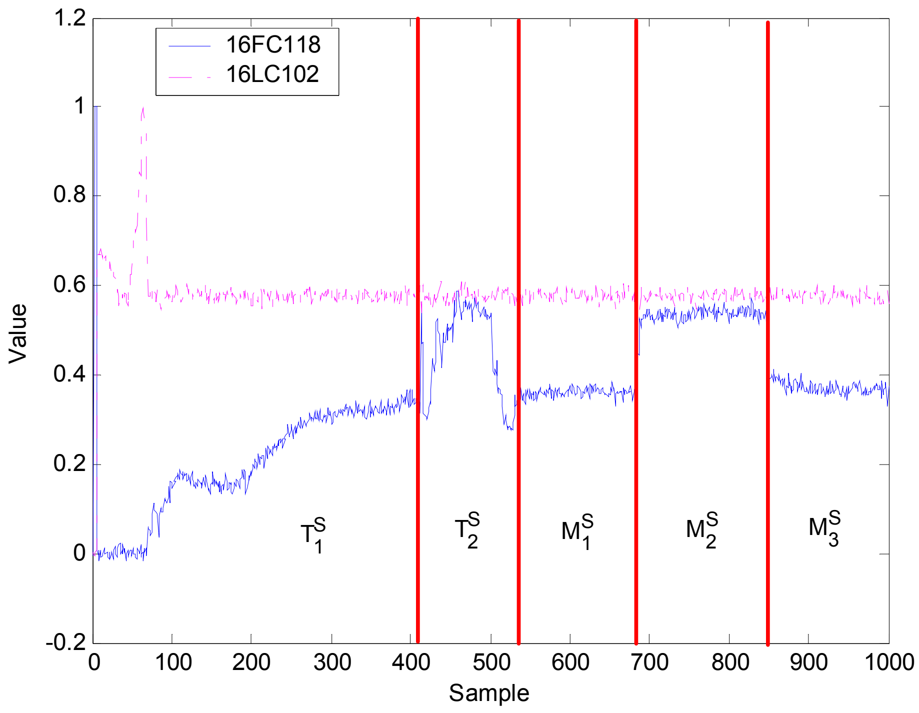

Figure 4-21: Evolution of two process variables of waste heat boiler section

Testing results: The sub-state identification layer for this case study comprises of five separate TDNNs. The classification errors are shown in TABLE 4-13. The overall validation error of this OVON is found to be 7.5%.

TABLE 4-16: OVON sub-state identification networks for waste heat boiler section

Sub-network	Variable name	No. of sub-states	TDNN Structure	Training ε	Validation ε
VN_1	16FC118	3	25-8-1	9.8%	11.6%
VN_2	16FI119	3	25-10-1	7.5%	6.7%
VN_3	16LC102	2	25-10-1	1.3%	1.2%
VN_4	16LI102A	2	25-8-1	1.3%	1.2%
VN_5	16PC111	2	25-8-1	1.7%	1.5%

An OCON system was also constructed and tested. Three RBFs corresponding to the three states discussed above were initialized with the structures listed in TABLE 4-14. The final identification error of OCON is about 5.4%.

TABLE 4-17: OCON sub-state identification networks for waste heat boiler section

Sub-network	RBF Structure	Training ε	Validation ε
CN_1	25-20-1	6.3%	6.4%
CN_2	25-15-1	7.5%	7.5%
CN_3	25-20-1	5.4%	5.6%

For comparison purposes, TDNN, RBF and Elman networks with structures 25-40-1, 25-20-1 and 5-20-1 respectively were also trained and tested. A summary of their performances is shown in TABLE 4-15. Although, the identification error of Elman network is slightly less than OVON and OCON, the training time is considerably large, thus illustrating the good tradeoff between accuracy and training time of OVON and OCON.

TABLE 4-18: Performances of neural networks for waste heat boiler section

	Training Time (s)	Validation ε

OVON	Sub-networks: 100	7.5%
	Unification layer: 5	
OCON	25	Winner-takes-all: 8.6%
		Regulator: 8.4%
TDNN	250	12.4%
RBF	14	11.5%
Elman Network	3634	7.2%

4.4 Conclusions and Discussion

Multi-dimensional temporal pattern recognition is an industrially important but challenging problem. Neural networks can be used for this purpose, however their training is computationally intensive especially when the number of variables is large and the patterns are complex. Two new neural network architectures are proposed in this chapter. In both structures, the original classification problem is decomposed into a number of simpler classification problems. The OVON uses a sub-state identification layer where a set of neural networks are used to identify simpler uni-variate, temporal patterns. A unification layer is subsequently used to infer the process state based on the sub-states, through multi-dimensional, static pattern recognition. The OCON incorporates a different philosophy: a state-identification layer is used to identify the presence or absence of a temporal pattern in multi-dimensions; the state of the process is inferred by analyzing the static, multi-dimensional outputs from the state-identification layer.

OVON and OCON are usually superior to traditional monolithic networks in terms of classification accuracy and training time. Each sub-network in OVON and OCON has a simpler structure than a monolithic neural network for the same problem. Generally, the OVON unification layer and the OCON regulator layer can filter out errors from the sub-layer and provides robustness to noise and disturbances. The OVON is superior in terms of training time, if retraining is necessary to accommodate sensor faults. The OCON is superior in terms of training time, if retraining is necessary to accommodate new operating states.

The improved classification accuracy of OVON is derived at an additional cost. In order to train the sub-state identification layer, a prior knowledge of sub-states of each variable is necessary. The uni-variate nature makes this a straightforward step when the boundaries between clusters can be located accurately. However, for problems with a large number of variables, this analysis is cumbersome. The OCON does not have any such additional requirement. In both OVON and OCON (and in traditional networks), misclassification occurs during state change where there is no clear separation between the states or the sub-states. For instance, in the air blower section, about 50% of the errors occur when the process moves from T_1^B to M_1^B. A better method for accurate state boundary recognition is therefore needed and is the subject of our current work.

Chapter 5. Context-Based Recognition of Process States

5.1 Introduction

In chapter 4, we have discussed the temporal pattern recognition using neural network in industrial process. However, it is found that patterns whose interpretation varies across contexts are common in many state identification problems. The resulting one-to-many mapping between patterns and their classes cannot be adequately handled by traditional pattern recognition approaches. In this chapter, this problem is addressed.

In many real-world domains, the classification of a pattern heavily depends on the *context*. In general, context can be considered as one or more features that "constrain problem solving without intervening in it explicitly" (Brezillon, 1999). It thus serves as a filter to define what knowledge or features should be considered, and how they should be used. As a classical example, consider the following rule in MYCIN (Clancey, 1983), an expert system for advising physicians on treating bacterial infections of the blood and meningitis.

IF

(1) The infection which requires therapy is meningitis

(2) Only circumstantial evidence is available for this case

(3) The type of meningitis is bacterial

(4) The age of the patient is greater than 17 years old, and

(5) The patient is an alcoholic

THEN

There is evidence that the organisms causing the infection, are diplococcus-pneumoniae or E.*coli*.

Here, clause (4) does not serve to identify the infection, but has the purpose of screening, i.e., to make the rule valid only for adults. It thus just constrains the applicability of the other features and does not explicitly determine the result.

Similar situations occur in chemical and biological processes. Consider the fermentation process for an antigen production reported by Muthuswamy and Srinivasan (2003). The process uses yeast *Pichia pastoris* for the production of the antigen. Being methylotrophic, *P. pastoris* is able to thrive on methanol as well as glucose/glycerol-based substrates. To take advantage of this, the process is operated in different phases. Initially the process is operated in a batch phase; airflow is introduced subsequently to overcome the stirrer's limited capacity to provide oxygen to the reactor. When all the nutrients are depleted, a glycerol feed is started. The process continues in this (fed-batch) phase until a critical cell mass is reached. The feed is then stopped and fermentation continues in the batch mode. When the glycerol nutrient in the fermentor is exhausted, a methanol feed is started in order to induce product formation.

The switch from one phase to another can be detected from patterns in the process variables. In this process, the depletion of nutrient is manifested as a spike in dissolved oxygen, which the process operator or a supervisory control system can use to infer the process state (Muthuswamy and Srinivasan, 2003). The feature is thus essential in the air-flow-assisted-phase as well as the subsequent batch-phase. However, as shown in Figure 5-1, the same pattern (spike in dissolved oxygen) also occurs numerous other times due to process disturbances. These do not have any physiological significance and no control actions need be taken at these times. Thus, the "spike" pattern is a relevant feature in some process states but irrelevant in others. The interpretation and use of this pattern therefore has to take the current process phase (i.e. the context) into account. Similar traits have been reported in other biological processes as well (Gollmer and Posten, 1995)

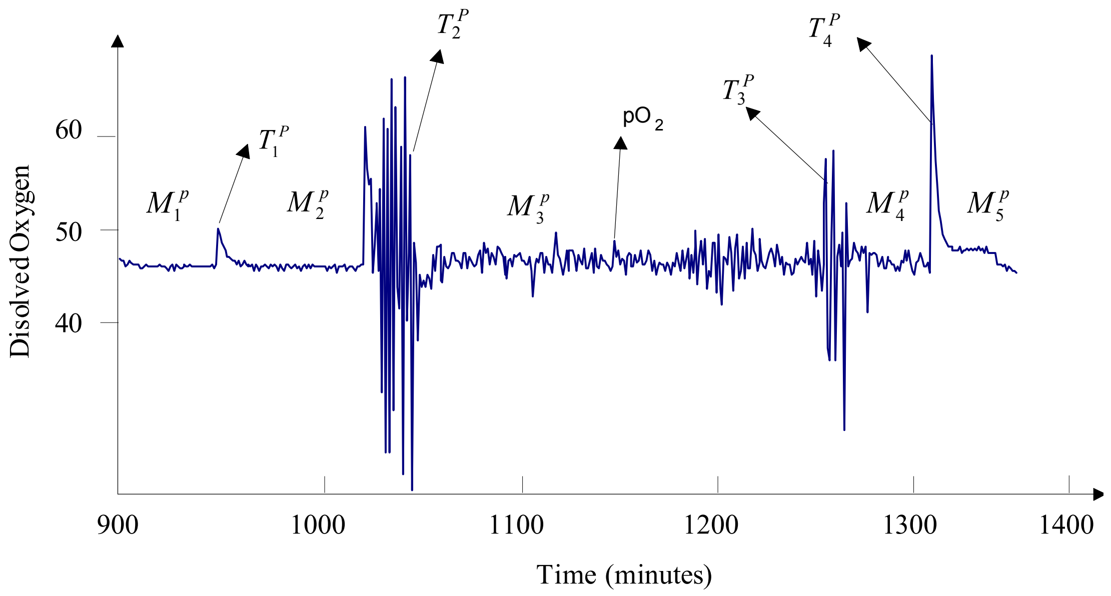

Figure 5-1: Operating states in run SMB78 of *P. pastoris*

In pattern recognition, a feature can be considered as contextual information if it does not directly determine the class of a pattern. However, the absence of this feature would lead to ambiguous or erroneous classification. The presence of contextual features usually becomes evident when a change in the context leads to a radical change in the interpretation of a pattern (Brezillon, 1999). As can be seen from the above example, context sensitivity necessitates a one-to-many mapping, i.e., a pattern may correspond to many classes. Traditional pattern recognition approaches are suitable for one-to-one or many-to-one mappings and cannot adequately characterize one-to-many situations. To convert the one-to-many mapping to a conventional one, contextual information has to be introduced.

Turney (1993a, 1993b) introduced the general definition of context-sensitive features in any pattern recognition problem. Let C_{nm} be a finite set of classes in an *nd*-dimensional input space and [X^{nd}, C] be a training pair, where $X^{nd} \in F_1 \times F_2 \times \ldots \times F_{nd}$ and $C \in C_{nm}$. We assume that instances are sampled identically and independently. In an instance of the form [X^{nd}, C], where $X^{nd} = [X_1, X_2, \cdots, X_{nd}]$, we use X_i to notate the *i*-th feature and x_i to represent the value of *i*-th feature. Similarly, *c* is the value of C. The context-based pattern recognition problem is defined by differentiating between three types of types of features: *primary, contextual, and irrelevant.*

Primary Feature: X_i is a primary feature for predicting class *C* when there is a value x_i of X_i and there is a value c of C such that

$$p(C = c \mid X_i = x_i) \neq p(C = c)$$

Primary features are defined as those features x_i that are useful for classification when considered in isolation, without regard to other features. In the fermentation example, the 'spike' in dissolved oxygen is a primary feature.

Contextual Feature: A feature X_i is contextual for predicting the class C when X_i is not a primary feature for predicting the class C and there is a value set of X^{nd} such that:

$$p(C = c \mid X_1 = x_1, \cdots, X_{nd} = x_{nd}) \neq p(C = c \mid X_1 = x_1, \cdots, X_{i-1} = x_{i-1}, X_{i+1} = x_{i+1}, \cdots, X_{nd} = x_{nd})$$

In other words, if X_i is a contextual feature, we can make a better prediction when we know the value of X_i rather than when the value is unknown. It is not useful in isolation, but can be useful when combined with other (primary) features. The phase of the fermentation process is an example of a contextual feature.

Irrelevant Feature: A feature X_i is irrelevant for predicting the class C when X_i is neither a primary feature nor a contextual feature. Irrelevant features are not useful for classification, either in isolation or when combined with other features.

$$p(Y = y \mid X_1 = x_1, \cdots, X_{nd} = x_{nd}) = p(Y = y \mid X_1 = x_1, \cdots, X_{i-1} = x_{i-1}, X_{i+1} = x_{i+1}, \cdots, X_{nd} = x_{nd})$$

Context-sensitive Feature: A primary feature X_i is said to be context-sensitive to a contextual feature X_j when there are values of these features such that:

$$p(Y = y \mid X_i = x_i, X_j = x_j) \neq p(Y = y \mid X_i = x_i)$$

In the previous example, dissolved oxygen is a context-sensitive feature.

Once features have been segregated into primary, contextual, and irrelevant ones, a mechanism to incorporate contextual information into pattern recognition has to be formulated. Several strategies have been proposed for this (see Chapter 2). This chapter proposes the use of context-based pattern recognition in chemical and biological processes. In Section 5.2, we show that online identification of process states can be formulated as a context-based pattern recognition problem. The primary and contextual features for this problem and the mechanism to use contextual information are also described. In Section 5.3, we describe a novel neural network-based architecture, called Operating State Identification Neural Network (OSINN). We explore three different ways to detect context changes in real-life processes. Subsequently, we illustrate the efficacy of these using two case studies. Normal and abnormal operating states during the startup of a fluidized catalytic cracking unit simulation are identified using OSINN in Section 5.4. In Section 5.5, the phases during the operation of a lab-scale fed-batch fermentation process are identified.

5.2 State Identification as a Context-based Pattern Recognition Problem

Chemical plants commonly operate normally at a finite set of operating states $\hat{S}$ which can be classified as modes and transitions (Srinivasan, 2002). An operating state where the process variables vary within a narrow band is termed as a *mode*. The state of the process between two modes and characterized by large changes in one or more variables is termed as a *transition*. A long transition can be sub-divided into several phases to obtain a higher resolution. The control of non-steady state processes often requires state-specific control configurations and controller settings (Muthuswamy and Srinivasan, 2003). Advanced control and optimization strategies make use of state-specific models. Alarm management systems can set alarm limits based on the current operating state to avoid nuisance alarms (Arnold *et al.*, 1989; Jensen, 1997; Wang *et al.*, 2000). To implement state-specific control strategies, the current active state of the process has to be determined (Rosen and Yuan, 2001).

Process operating state identification is the task of identifying the state of the plant at any point of time. The operating state is normally inferred from the time evolution of the real-time sensor values. For a process with *d* online measurements, the *nd* dimensional vector $X^{nd}(t):[X^d(t), X^d(t-1), \cdots, X^d(t-l)]$ is termed as the *process feature vector*, where $x^d(t-l)$ is the value of *d* process variables $X^d:[X_1, X_2, \cdots, X_d]$ at time $(t-l)$. Here, the data window ($l \geq 0$) is used in order to capture information from the process evolution and not just the current value of $X^d(t)$. l determines the size of the data window and $nd = d \times (l+1)$. Thus, the process feature vector $X^{nd} \in \mathbb{R}^{nd}$ reflects the dynamic status of the process at time *t* and can be used as the primary feature for the identification of operating state. An alternate primary feature can be developed as follows. Let *nm* be the number of distinguishable patterns in X^{nd}. A *process pattern* summarizes the temporal evolution of the process within duration *l* and arises from the mapping $PA_i \leftarrow X^{nd}(t)$. Each process pattern can be encoded by a label $PA_i \in \{PA_1, PA_2, \cdots, PA_{nm}\}$. Process feature vectors that are labeled with the same pattern PA_i have similar values and temporal evolutions in X^{nd}. A classifier for state identification can use the process pattern or the process feature vector as the primary feature. The selection of the former is appealing especially for cases where the temporal evolution of the feature vector is complex. In the subsequent discussion, the process pattern is used as the primary feature for ease of notation, but the process feature vector can also be used (see Section 5.3.1).

The process feature vector and the process pattern are context-sensitive and a contextual feature $\hat{S}_{con}$ is necessary for satisfactory state identification. In this thesis, we use the preceding operating state as the contextual feature. During process evolution, the foregoing state $\hat{s}_{\hat{j}}$ does not usually foretell which state will follow and is not a primary feature.

$$P(\hat{S}(t) = \hat{s}_{\hat{i}} \mid \hat{S}_{con} = \hat{s}_{\hat{j}}) << 1$$

But it contains essential historical process state information needed to interpret the process pattern which is used to infer the next state. It therefore serves a contextual role.

$$P(\hat{S}(t) = \hat{s}_{\hat{i}} \mid X^{nd} \in PA_i, \hat{S}_{con} = \hat{s}_{\hat{j}}) \geq P(\hat{S}(t) = \hat{s}_{\hat{i}} \mid X^{nd} \in PA_i) \qquad [5\text{-}1]$$

Thus, the operating state identification problem is formulated as a context-based pattern recognition problem.

Context-based operating state identification is illustrated using the fermentation process described in Section 5.1. As seen there, the operating state cannot be identified based only on the process pattern since the same pattern (for e.g., spike in pO_2) can occur numerous times and be either relevant or irrelevant in a given phase. Muthuswamy and Srinivasan (2003) detect the onset of a process state using rules, which consider the pattern only in the context of a process state. Their rule for identifying the beginning of T_2^P can be interpreted as follows:

IF

(1) There is a spike in pO_2, and

(2) The previous state is airflow-assisted-batch-phase

THEN

It can be concluded that the glycerol nutrient has been depleted (T_2^P)

If Clause (1) is used by itself, nutrient-depletion will be erroneously identified over 500 times during the process evolution, that is,

$$P(T_2^P \mid \text{Feature} = \text{Spike in pO}_2) \cong \frac{1}{500}$$

where P represents the probability that the transition T_2^P occurred at time t. The contextual information – previous process state (Clause (2)) – is used to constrain Clause (1) and improve the prediction. That is,

$$P(T_2^P \mid \text{Feature} = \text{Spike in pO}_2\ \&\ \text{context} = \text{airflow-assisted-batch-phase}) \cong 1$$

As can be seen from the above example, the contextual feature is a discrete variable that takes values from the set of discrete process states. The contextual feature is not a measured quantity, and does not change continuously at the process sampling rate. It changes only when a process state change occurs and therefore at irregular intervals during the evolution of the process. In the following section, we propose a neural network-based architecture, called Operating State Identification Neural Network (OSINN).

5.3 Neural Network Architecture for Operating State Identification

The identification of operating state requires the detection and management of context, as well as pattern recognition using primary and contextual information. Additionally, when the process pattern is used as the primary feature, the identification of process pattern from the process feature vector has to be considered. As shown in Figure 5-2, in OSINN, these are implemented by the Context Manager, State Identification Block, and Data-preprocessor.

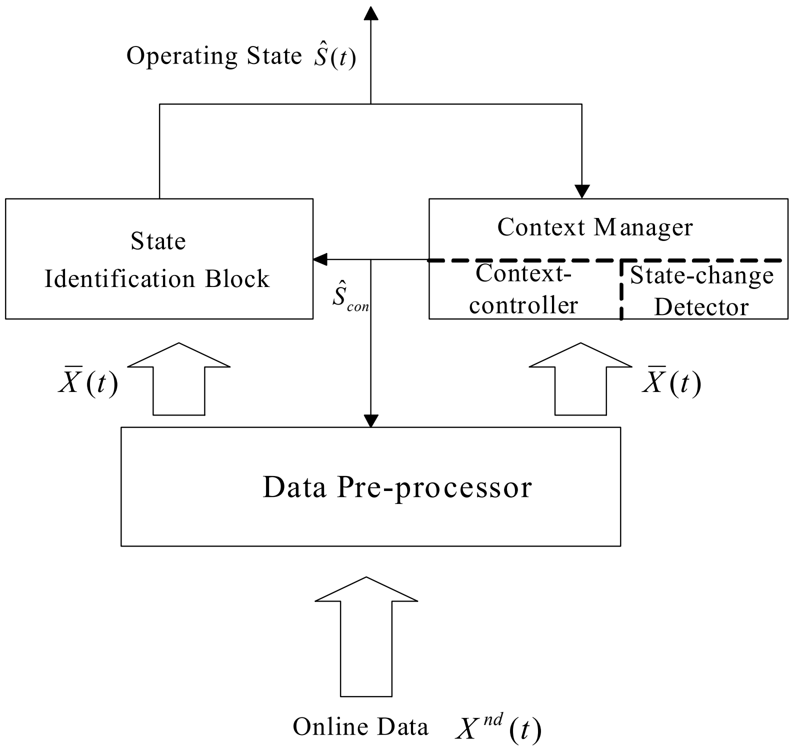

Figure 5-2: Structure of OSINN

A **data-preprocessor** is used to ameliorate the input data $X^{nd}(t)$ before it is used for state identification. Preprocessing can either be a normalization based on the contextual information $\hat{S}_{con}$ or a preliminary classification to identify the process pattern PA_i. Statistical pattern recognition, syntactic pattern recognition, or neural networks can be used for the latter. Neural networks have been adopted in this thesis, because they have ability to approximate complex nonlinear functions. Some commonly used neural networks are Time Delayed Neural Network (TDNN), Radial Basis Function (RBF) Network, and Elman Network. We use a RBF in case study 1 (Section 5.4) and TDNN for case study 2 (Section 5.5). The preprocessed data $\bar{X}(t)$ is fed to the context manager and/or state identification block. The main advantage of preprocessing the data is that the OSINN becomes more robust to noise which is filtered at this stage.

During process evolution, state change can occur at irregular intervals ranging from minutes to days. One main requirement in context-based pattern recognition is the ability to detect the occurrence of context change. The **context manager** detects changes in context and provides the correct contextual feature to the state identification block. It consists of two sub-blocks ─ Context-controller and State-change Detector. The output of the context-controller is the preceding operating state of the process which is a discrete contextual feature. When the process operates within the same context, there is no change in the context-controller's output. The state-change detector monitors the process and flags state change. A change of context is conditional on a state change. Similarly, a change of state is conditional on a change in the process pattern. Therefore, in OSINN, context change is detected by monitoring for drift in $\hat{S}(t)$ or $\bar{X}(t)$. Specifically, if the change of $\hat{S}(t)$ or $\bar{X}(t)$ exceeds a user-defined threshold called the sensitivity-threshold (θ), the context-change detector flags a change. When a context change has to be implemented, the context-controller replaces the value of the contextual feature with one corresponding to the current operating state.

The state identification block uses the contextual feature $\hat{S}_{con}$ along with the primary features to identify the current operating state of the process, i.e., it realizes the mapping $\hat{S}_{\hat{i}} \leftarrow \left\langle \bar{X}(t), \hat{S}_{con} \right\rangle$. The output of the state identification block is the tag (normally an integer value) of the current state. The general procedure for designing an OSINN is as follows:

Data Analysis: Analyze the process feature vector and specify the target operating states based on process knowledge. Clustering may be used for this purpose (Srinivasan *et al.*, 2004a). Include the value of the contextual features to the training data (See Section 5.4 for an example).

Structure Selection: Select a suitable algorithm for data preprocessing. Several algorithms and structures can be used as described in the following sections.

Network Training: The state identification block is first disconnected from the other blocks and trained. Suppose the process data contains *nk* operating states. The training pair for state identification block can be formed as: $\left\{[\left\langle \bar{X}_1, \hat{S}_{con_1} \right\rangle, \hat{s}_1], [\left\langle \bar{X}_2, \hat{S}_{con_2} \right\rangle, \hat{s}_2], \cdots, [\left\langle \bar{X}_{nk}, \hat{S}_{con_{nk}} \right\rangle, \hat{s}_{nk}]\right\}$, where $\bar{X}_1$ is the dataset containing all the pre-processed vectors generated from operating state $\hat{s}_1$, and $\hat{S}_{con_1}$ is the contextual feature for operating state $\hat{s}_1$. Thus, $\left\langle \bar{X}_1, \hat{S}_{con_1} \right\rangle$ is used as the input to the state identification block and $\hat{s}_1$ as the training target.

Once an OSINN has been trained, it can be used for online state identification by inputting real-time process data to the data-preprocessor and specifying the initial value of $\hat{S}_{con}$. The general architecture of OSINN described above can be implemented in different structures with particular algorithms for the three blocks. Three such structures are described next, each using a different method to detect context change. A comparison among them is presented subsequently using two case studies.

5.3.1 Contextual Normalization OSINN (OSINN-N)

This OSINN structure implements contextual normalization as shown in Figure 5-3. Here, the process feature vector $X^{nd}(t)$ is input to the data-preprocessor along with the contextual information $\hat{S}_{con}$. In the data-preprocessor, the process feature vector is normalized based on their measurement ranges as well as $\hat{S}_{con}$. $\hat{S}_{con}$ serves as a transform factor and shifts the input $X^{nd}(t)$ to a new region in the input space so that there is no overlap among the patterns.

$$\bar{X}(t) = G(X^{nd}(t), \hat{S}_{con}) \qquad [5\text{-}2]$$

where *G()* is a transform function. While a variety of functions can be used for *G()*, we use a simple linear transform in this thesis:

$$\bar{X}(t) = \tilde{X}^{nd}(t) + W\hat{S}_{con} \qquad [5\text{-}3]$$

where W is a real number selected so that there is no overlap among the range of the primary features in different context. The mapping for state identification $\hat{S}_{\hat{i}} \leftarrow \bar{X}(t)$ is thus transformed into a traditional one-to-one or many-to-one type.

The context manager monitors the state identification block output $\hat{S}(t)$ and provides the context information $\hat{S}_{con}$. The state change is flagged when $\Delta\hat{S}(t) = (\hat{S}(t) - \hat{S}(t-1)) > \theta_N$, the sensitivity-threshold.

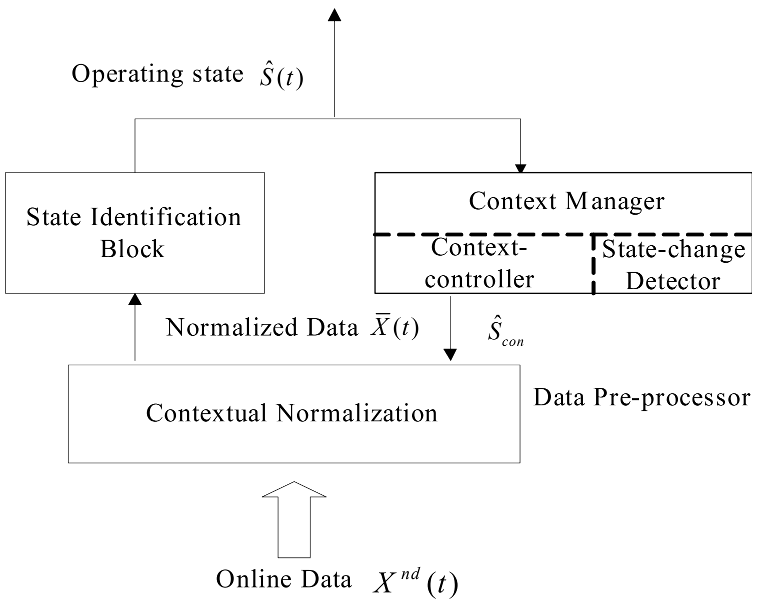

Figure 5-3: Structure of OSINN-N

5.3.2 Context Change Detection Using Drift in Process Pattern

This structure, pattern-based OSINN (OSINN-P), is an implementation of contextual expansion. The context information $\hat{S}_{con}$ is used explicitly together with the process pattern as an additional input as for pattern identification. The data-preprocessor shown in Figure 5-4 consists of two sub-parts: data normalization and process pattern identification. The normalization part scales the process feature vector $X^{nd}(t)$ based on Equation [5-4].

$$\tilde{X}_i = \frac{X_i - X_L}{X_H - X_L} \qquad [5\text{-}4]$$

where, $\tilde{X}_i$ is the normalized process variable X_i, and X_H and X_L are the high and low limits of the sensor range respectively. The normalized feature vector $\tilde{X}^{nd}(t)$ is presented to a neural network, which classifies the temporal inputs based on their values and evolution. The outputs $\bar{X}(t)$ of this network is the process pattern *PA(t)*. Since this pattern recognition step is context independent, any network can be chosen based solely on the complexity of the pattern.

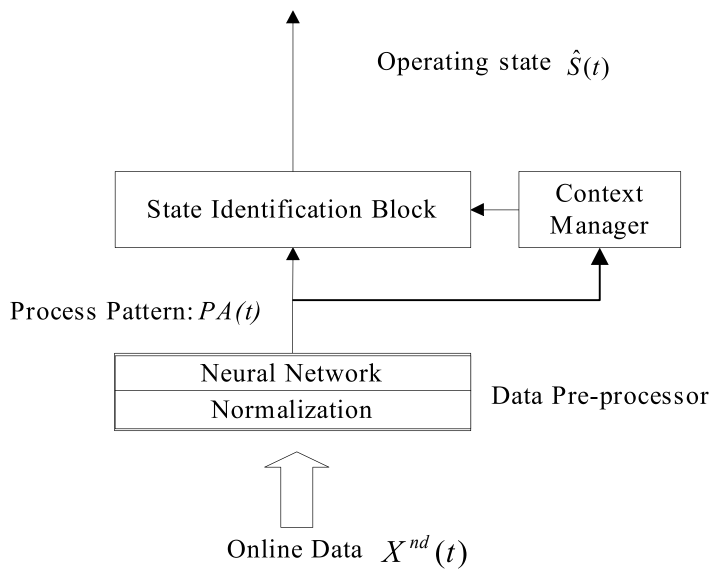

Figure 5-4: Structure of OSINN-P

The context manager and state identification block for OSINN-P are shown in Figure 5-5. The context manager consists of two sub-blocks: State-change Detector and Context-controller. The state-change detector monitors *PA(t)*. If the output *PA(t)* drifts from PA_i, that is, $\Delta PA(t) = (PA(t) - PA(t-1)) > \theta_P$, where θ_P is the sensitivity-threshold, the process is considered to undergo a state change. The state-change detector identifies the change and the context-controller updates the contextual information $\hat{S}_{con}$ if needed.

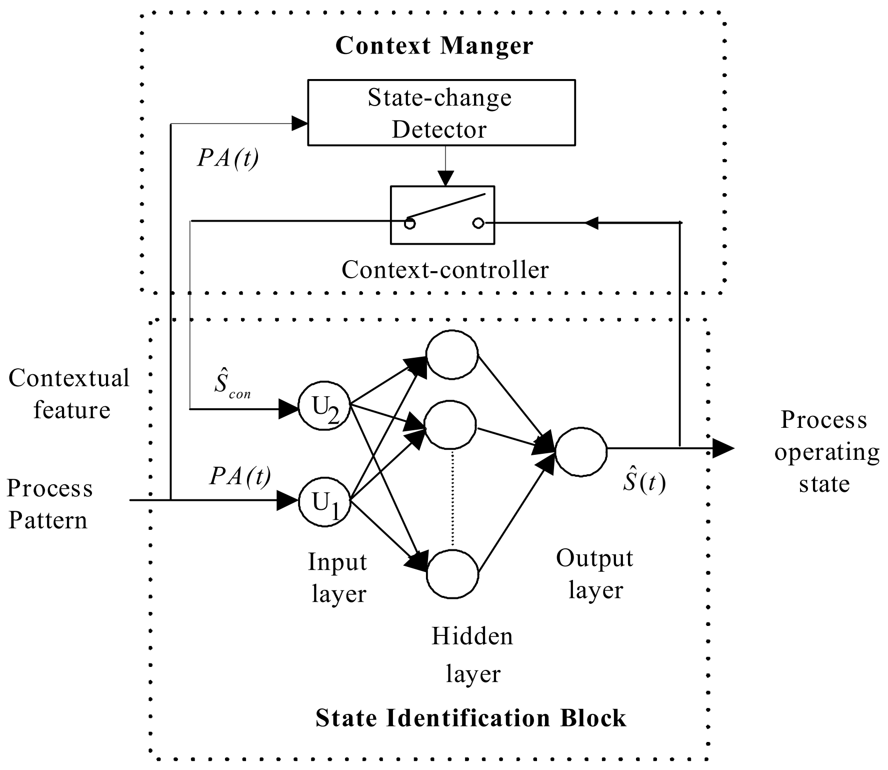

Figure 5-5: Structure of Context Manager and State Identification Block of OSINN-P

The state identification block is a neural network with two types of neurons in the input layer, U_1 and U_2. U_1 receives the process pattern *PA(t)* from the data-preprocessor and U_2 receives the contextual information $\hat{S}_{con}$ from the context manager. The state identification block identifies the operating state based on the combination of *PA(t)* and $\hat{S}_{con}$. This is again a traditional one-to-one or many-to-one mapping problem which can be directly solved by the network.

5.3.3 Context Change Detection Using Drift in Operating State

This structure, called state-based OSINN (OSINN-S), is another implementation of contextual expansion. Figure 5-6 shows the structure of the OSINN-S. The context manager and state identification blocks are shown in Figure 5-7. In contrast to OSINN-P, where drift in *PA* was used to detect context change, in OSINN-S a drift in state is used for this purpose. The state-change detector monitors and detects the drift in $\hat{S}(t)$.

Once $\overline{\hat{S}} = (\hat{S}(t) - \hat{S}(t-1)) > \theta_S$, where θ_s is the sensitivity-threshold, the context-controller will update the contextual feature accordingly. The working mechanism of OSINN-S is similar to OSINN-P.

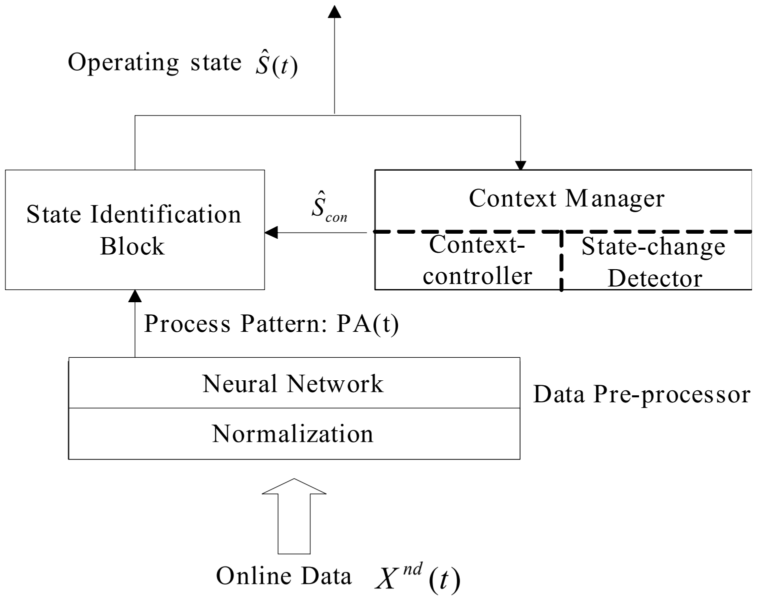

Figure 5-6: Structure of OSINN-S

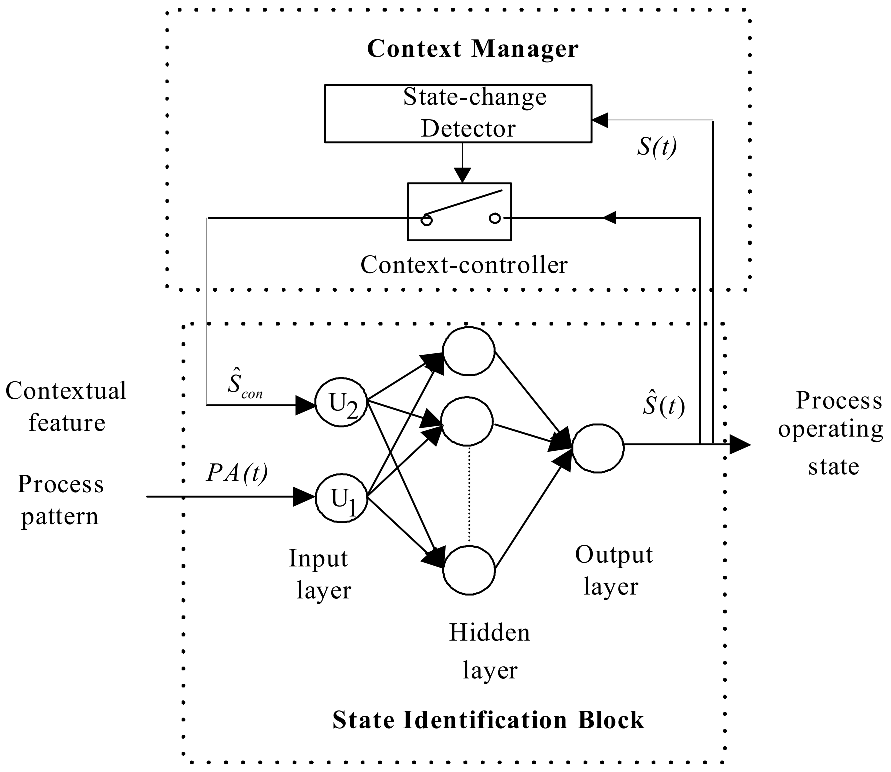

Figure 5-7: Structure of Context Manager and State Identification Block of OSINN-S

Disturbance Rejection in OSINN: Chemical processes are susceptible to large disturbances which can last for several samples. In order to make state identification robust, all three OSINN architectures incorporate three checks. The occurrence of any new context is flagged only after it has lasted for a minimum duration called **dwell-time** T_d (Bhagwat *et al.*, 2003). During this period, OSINN holds the value of the state, i.e., $\hat{S}(t) = \hat{S}(t-1)$. While this would delay state identification by a period equal to T_d samples, the result is less sensitive to disturbances. Reducing T_d will reduce the delay in identifying the new state but increase risk of premature context change detection. This is alleviated by a backtracking mechanism provided by the evaluation interval. If a context change occurs because of a state change, it is considered tentative until a period of time, called **evaluation-interval**, T_e has passed. If this (first) state change has occurred due to a disturbance, the new state will last only for a short duration. If another (second) state change is detected before the end of the evaluation-interval, it is construed as an indication of a process disturbance and the original context (the one before the first state change) is reverted to. The evaluation-interval does not delay state identification but provides a mechanism to recover the original context in the face of misidentification of state change. The duration of T_d and T_e are set based on the knowledge of process disturbances and expected minimum duration of the states. In the following sections, we illustrate three OSINN structures using two case studies.

5.4 Operating State Identification in a Fluidized Catalytic Cracking Unit

In this section, the performance of the OSINN for industrial-scale applications was tested using data from the ShadowPlant. The ShadowPlant has been introduced in Chapter 3. As mentioned earlier, several runs of the startup were performed following the standard operating procedure. Two of them G_1 and G_2 are used here. In each section, superfluous variables were first eliminated to decrease the computational requirements.

5.4.1 Air Bower Section

Data from air blower section is used here to evaluate the networks' ability to identify the operating state. The air blower supplies air which is used to preheat the reactor/regenerator section during startup. Figure 4-4 shows an overview of the section. Out of the sixteen variables in the section, the seven shown in TABLE 5-1 were selected.

TABLE 5-1: Variables of air blower section

Name of variable	Description
16SI100	Speed of air blower
16FI100	Flowrate of steam to air blower
16PDI101	Differential pressure ΔP in air blower
16FI101	Flowrate of air discharged from air blower
16FC102	Flow in surge control
16TI100	Air blower discharge temperature
16FI106	Flowrate through air vent

The process data was clustered and similar process patterns were first identified based on the values and trends (Srinivasan, *et al.*, 2004a). For example, the process data shown in Figure 5-8 was clustered into eight operating states. M_1^A is the initial "cold-start" state of the section, T_1^A corresponds to the air-blower start-up, when ΔP 16PDI101 increased. During M_2^A, since the regenerator overhead valve 16PV105 is opened, regenerator overhead pressure drops and leads to a drop in 16PDI101. T_2^A is an intermediate state. During M_3^A, the regenerator overhead pressure control valve 16PV105 is closed and the pressure in regenerator builds-up. Correspondingly, air blower ΔP is also increased. T_3^A and T_4^A correspond to the introduction of feed to the reactor. M_4^A is an intermediate state during which the operator stabilizes the regenerator, starts up other sections of the FCCU, and links them to the regenerator section.

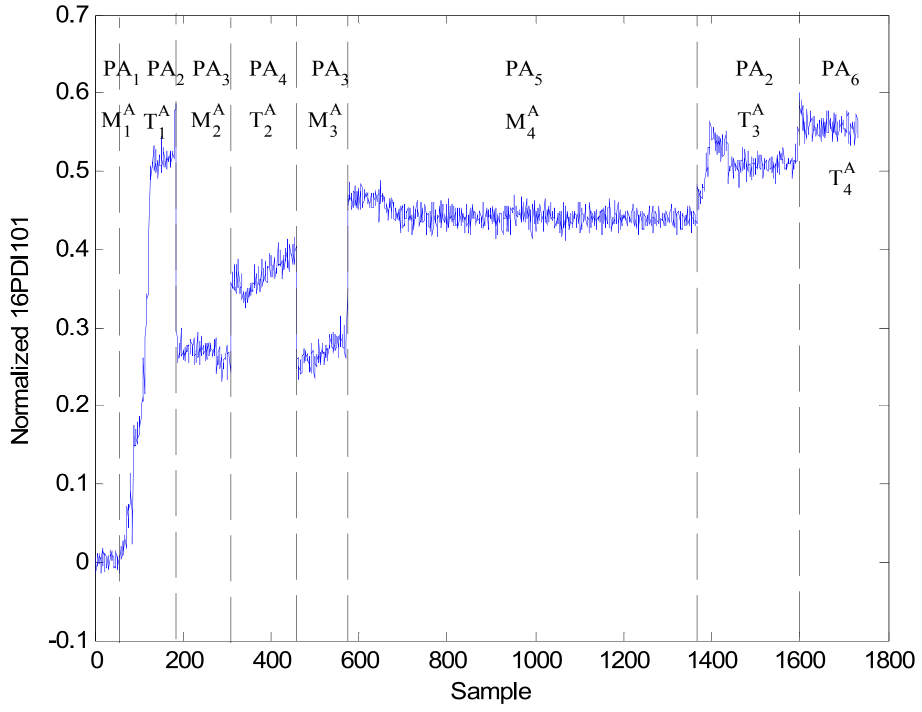

Figure 5-8: Process patterns and corresponding operating states in air blower section

An investigation of the data reveals that two modes – M_2^A and M_3^A – share the same process pattern PA_3. Two transitions T_1^A and T_3^A also have similar patterns, termed PA_2. From an operations point of view, these are different states since the valve 16PV105 is closed during T_1^A but is opened in T_2^A. However, a traditional pattern recognition method cannot distinguish between M_2^A and M_3^A or T_1^A and T_3^A and a context-based state identification is needed.

The performance of OSINN-P was evaluated first. A RBF (structure 70-40-6) which uses seven input variable values each with a data window of 10 samples is used as the data-preprocessor to identify the data pattern *PA(t)* based on temporal process features. Here, a new context is defined corresponding to each pattern (i.e. state). The state identification block is a RBF network. Samples from G_1 are used for the training of data-preprocessor to identify the six process patterns. The sensitivity-threshold for state-change detector in the Context Manager is set to $\theta_P = 0.5$.

The performance of each network was then evaluated using Equation [4-6] on the validation data set G2. The overall validation error on G2 for OSINN-P based on is 4.3%. In contrast, a RBF network with the same structure when trained directly to map the normalized process feature vector $\tilde{X}^{nd}(t)$ to operating states $\hat{S}_i$ without any contextual information has a validation error of 15%. The outputs of this RBF network without context information are shown in Figure 5-9. As can be seen, the network cannot distinguish between M_2^A and M_3^A or T_1^A and T_3^A. This illustrates that the traditional neural network structure is insufficient for operating state identification and context information is crucial.

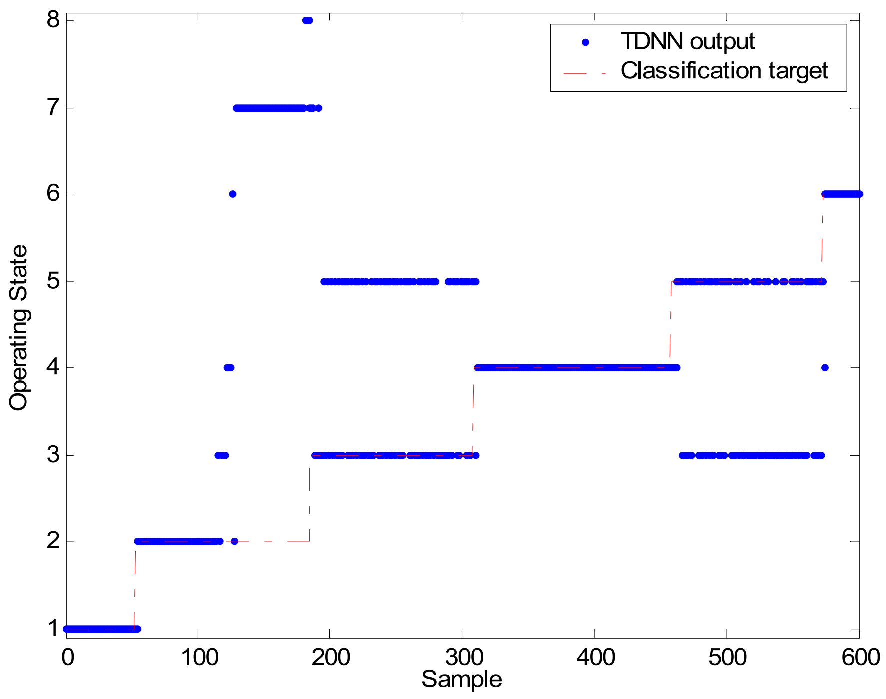

Figure 5-9: Operating state identification by RBF without context in air blower section

A detailed evaluation of the results was also performed. The classification error is categorized into Type-1 and Type 2 as presented in Chapter 4 (Section 4.3).

Figure 5-10 shows the output of OSINN-P for the validation dataset. The error distribution in the different states is shown in TABLE 5-2. It is clear that most of the errors are of Type-2. This is because (1) the process measurement is noisy especially during transitions, and (2) the dwell-time value of T_d=7 leads to a lag error during state changes. Similar results were obtained with OSINN-S ($\theta_S = 0.5$) as can be seen in Figure 5-11. The validation error was found to be 4.2%.

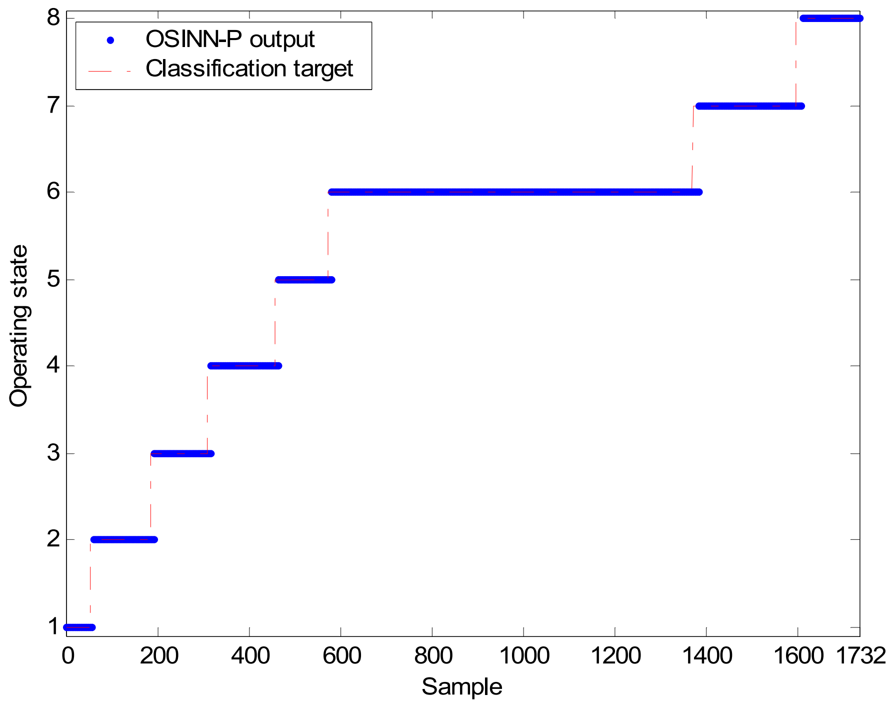

Figure 5-10: Operating state identification by OSINN-P in air blower section

TABLE 5-2: Validation errors by OSINN-P in air blower section

OSINN-P	Operating state	M_1^A	T_1^A	M_2^A	T_2^A	M_3^A	M_4^A	T_3^A	T_4^A
	Type-1 error	0	9	0	0	0	0	0	0
	Type-2 error	7	0	9	9	7	7	12	14

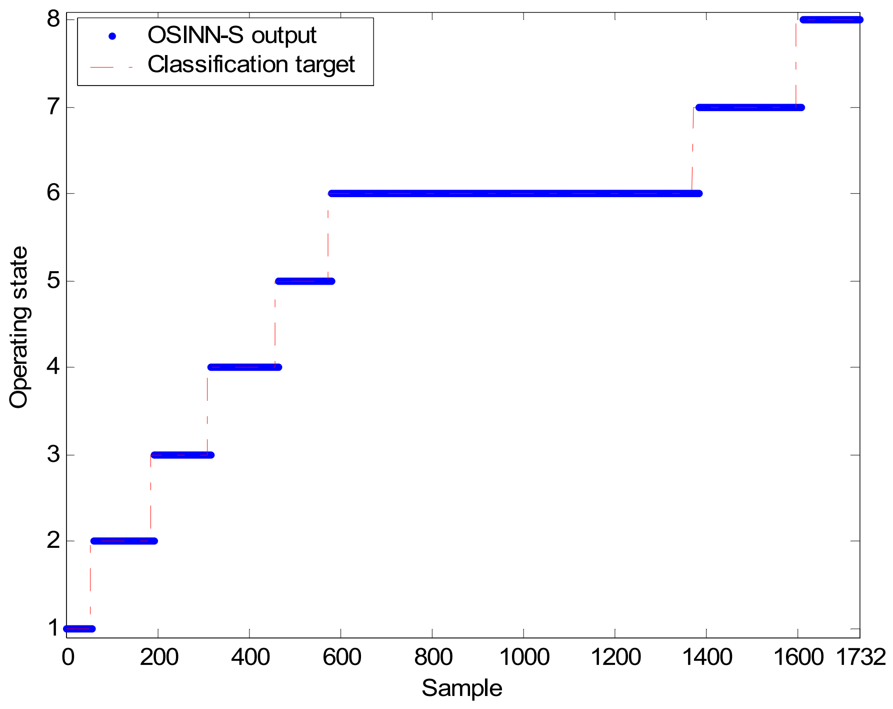

Figure 5-11: Operating state identification by OSINN-S in air blower section

OSINN-N was also constructed and tested for this section. To simplify the training, only two contexts are defined. $\hat{S}_{con} = 1$ for four operating states before T_2^A and $\hat{S}_{con} = 2$ for the subsequent states. This is adequate because as can be seen in Figure 5-8, the same process pattern is mapped to different operating states only after T_2^A. With a RBF (70-110-9) as the state identification block, the OSINN-N has an identification error of 3.5%. The complete identification results are shown in Figure 5-12 and TABLE 5-3. The Type-2 error induced by the dwell-time is smaller than that in OSINN-P and OSINN-S since the mis-identification during context switch is minimized.

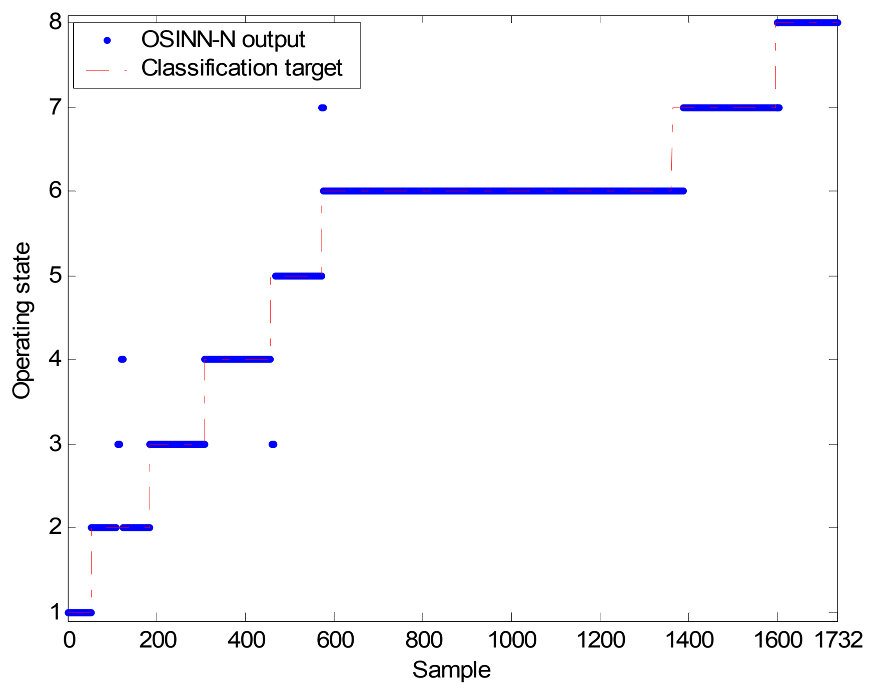

Figure 5-12: Operating state identification by OSINN-N in air blower section

TABLE 5-3: Validation errors by OSINN-N in air blower section

OSINN-N	Operating state	M_1^A	T_1^A	M_2^A	T_2^A	M_3^A	M_4^A	T_3^A	T_4^A
	Type-1 error	0	16	0	0	0	0	0	0
	Type-2 error	0	0	8	9	4	18	5	0

5.4.2 Selection of Parameter Settings

The proposed OSINN structures use three parameters whose main role is to filter noise. Their values have to be specified by the user based on knowledge of the process and its operation. In this section, we discuss the effect of these on state identification results. The sensitivity-threshold θ is used to detect context change. If θ is too small, the context-change detector will be triggered even by small changes in the process. This will make the OSINN susceptible to process disturbances. If θ is too large, the context-change detector will miss context changes and also leas to poor performance. A robust setting for θ is therefore important. In OSINN-N, the state-change detector monitors the change in process state, which is represented by an integer set (1, 2, 3, etc). Since, any context change results in a integer (large) change in $\hat{S}(t)$, we have found that it is easy to set a robust θ_N. θ_N values in [0.4 0.8] resulted in similar performance; hence, we have used 0.5 in all case studies. Similar results arise for θ_P and θ_S. In OSINN-P, the state-change detector monitors for a change in *PA(t)* which is also an integer set. The robustness for OSINN-S arises from the local activation property of the RBF network used in the state identification block.

The dwell-time and evaluation-interval work in concert to flag context switch correctly. T_d can be specified by analyzing the Type-2 error during the training of the data-preprocessor block (for OSINN-P and –S) or state-identification block (OSINN-N). If T_d, is set to a large value, the delay in context change detection will be unnecessary long. Reducing T_d will improve the speed of detection but will increase the risk of mis-identification by the context-controller. The evaluation-interval helps the context-controller to backtrack after a wrong state change. As discussed above, a valid state has to last at least for a period T_e. If the new state lasts for a shorter duration, it is considered to be invalid and the context-controller will revert to the previous contextual feature to correct the mistake. T_e is specified based on knowledge of the process operation and its time constants. A fraction (say 50%) of the shortest operating state can usually be used.

The robustness of the T_d setting is brought out using an example. In the above case study, if T_d is reduced from 7 to 4 and evaluation-interval turned off (set to zero), the OSINN error will increase substantially (about 30%). In the Figure 5-13(a), errors arise in OSINN-P's process pattern identification around sample 115 due to process disturbances. The resulting drift in $PA(t)$ is considered as a signal of state change and the context-controller prematurely changes the contextual feature incorrectly. The subsequent state identification results therefore are incorrect as shown in Figure 5-13(b). This is prevented by turning on the evaluation-interval. Figure 5-13(c) shows the state identification for the same value of T_d, but with $T_e=25$. As can be seen from the figure, although the initial state results are incorrect, at sample 122, the system automatically corrects the mis-action of the context-controller and the subsequent state identification result is correct. Because of this backtrack feature, acceptable results can be obtained even for small values of T_d.

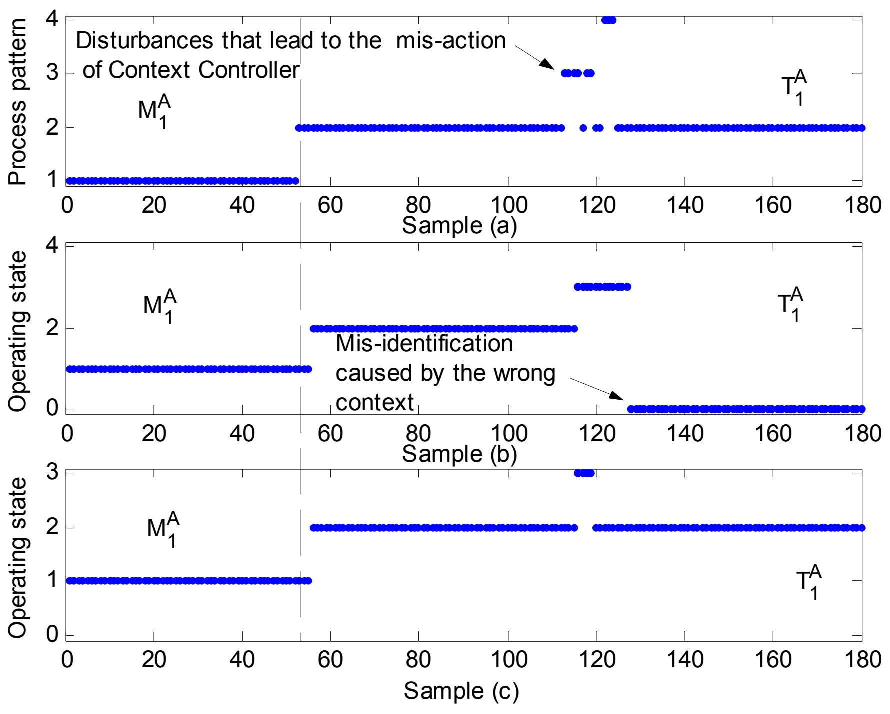

Figure 5-13: Example of the implementation of evaluation-interval in air blower section (a) Process pattern identification error (b) Mis-action of context controller leads to state identification error (c) State identification results with the implementation of evaluation-interval

5.4.3 Fractionator Section

In this section, data from Fractionator section is used to evaluate the networks' performance. Figure 4-18 shows the schematic of the Fractionator section. Twelve variables of the section are used for state identification as shown in TABLE 4-13.

Figure 4-19 shows the profile of two variables. As seen in the figure, there are five segments ─ two transitions (T_1^F and T_2^F) and three modes (M_1^F, M_2^F and M_3^F).

An investigation of the data reveals that two modes – M_2^F and M_3^F – share the same process pattern. Two transitions T_1^F and T_2^F also have similar patterns. Therefore, a traditional pattern recognition method cannot distinguish between M_2^F and M_3^F or T_1^F and T_2^F, and a context-based state identification is needed.

The performance of OSINN-P was evaluated first. The OCON presented in Chapter 4 (Section 4.3.3) is used as the data-preprocessor to identify the data pattern *PA(t)* based on temporal process features. Here, a new context is defined corresponding to each pattern (i.e. state). The state identification block is a RBF network. Samples from G_1 are used for the training of data-preprocessor to identify the six process patterns. The sensitivity-threshold for state-change detector in the Context Manager is set to $\theta_P = 0.5$.

The performance of each network was then evaluated using Equation [4-6] on the validation data set G_2. The overall validation error on G$_2$ for OSINN-P based on is 6.0%. In contrast, a TDNN network when trained directly to map the normalized process feature vector to operating states has a validation error of 38%. The outputs of this TDNN network without context information are shown in Figure 5-14. As can be seen, the network cannot distinguish between M_2^F and M_3^F or T_1^F and T_2^F. This illustrates that the traditional neural network structure is insufficient for operating state identification and context information is crucial.

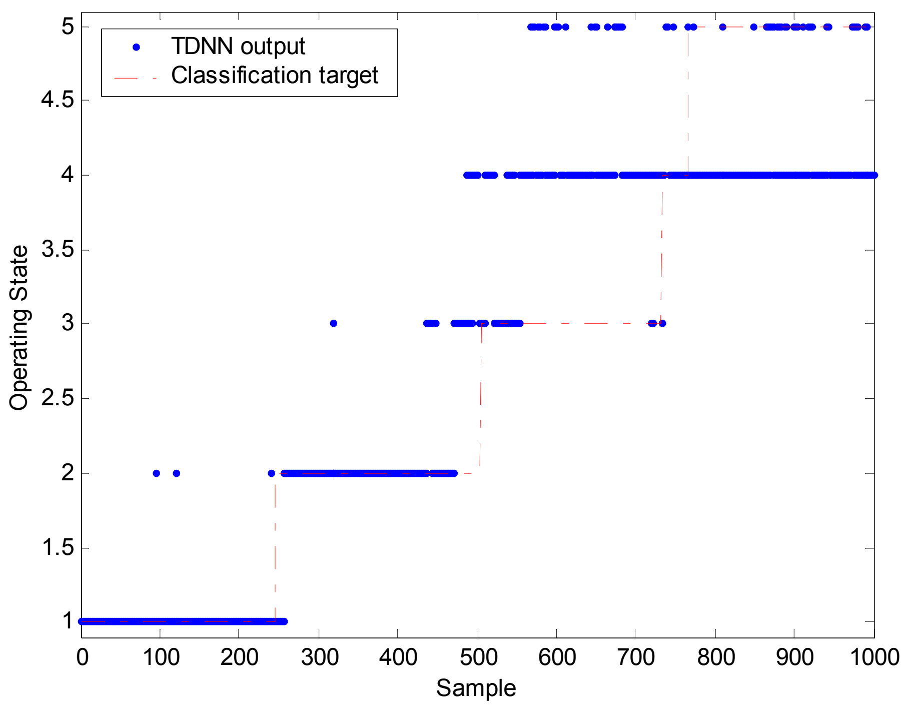

Figure 5-14: Operating state identification by TDNN without context in Fractionator section

Figure 5-15 shows the output of OSINN-P for the validation dataset. Similar results were obtained with OSINN-S ($\theta_S = 0.5$) as can be seen in Figure 5-16. The validation error was found to be 6.5%.

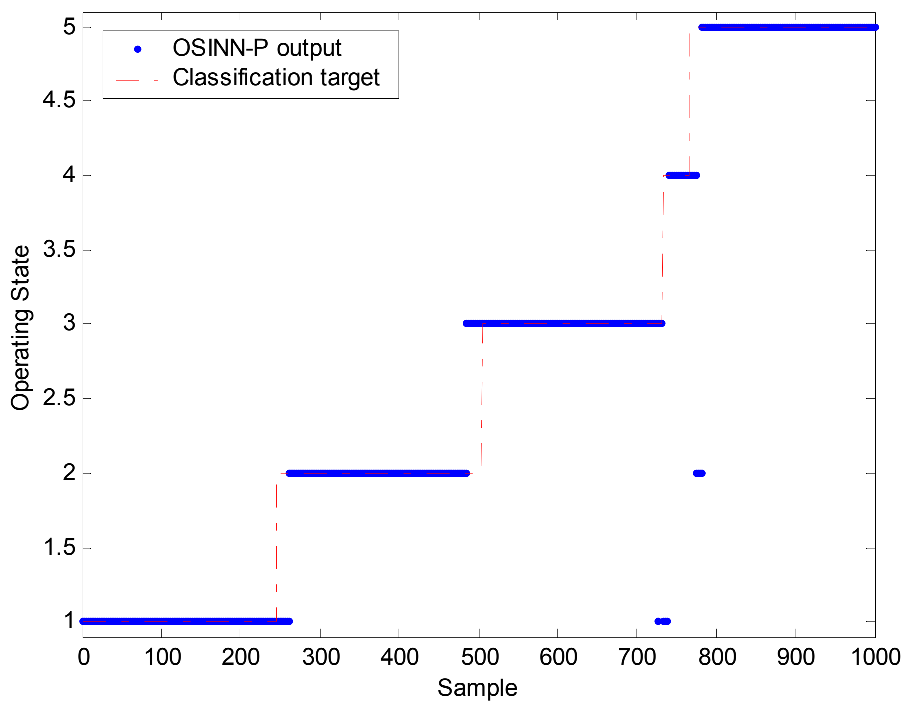

Figure 5-15: Operating state identification by OSINN-P in Fractionator section

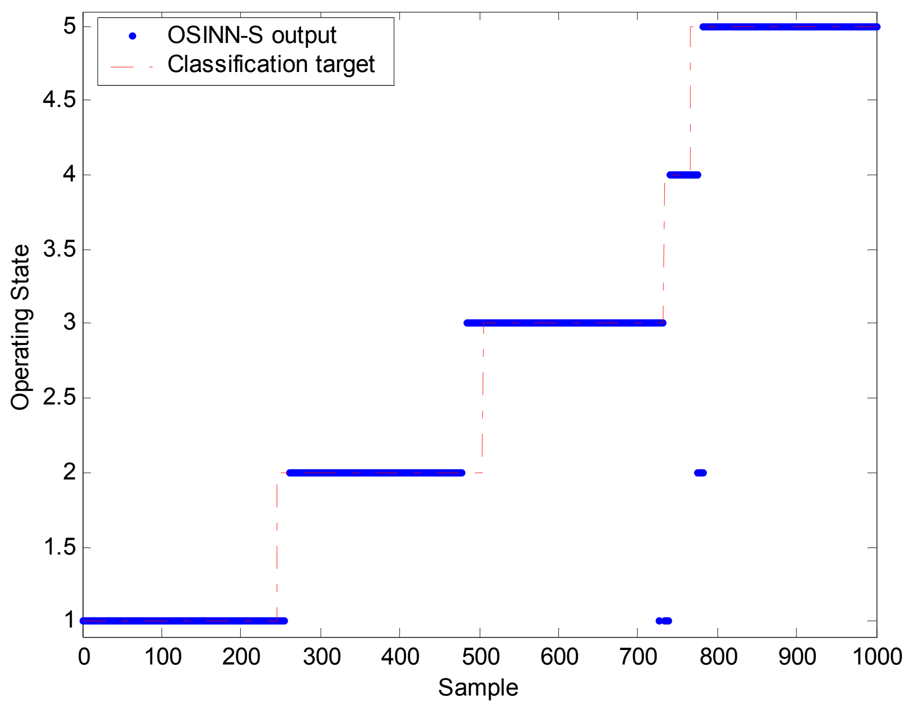

Figure 5-16: Operating state identification by OSINN-S in Fractionator section

OSINN-N was also constructed and tested for this section. To simplify the training, only two contexts are defined. $\hat{S}_{con} = 1$ for three operating states before T_2^F and $\hat{S}_{con} = 2$ for the subsequent states. This is adequate because as can be seen in Figure 4-19, the same process pattern is mapped to different operating states only after T_2^F. With a TDNN (60-50-1) as the state identification block, the OSINN-N has an identification error of 5.9%. The complete identification results are shown in Figure 5-17.

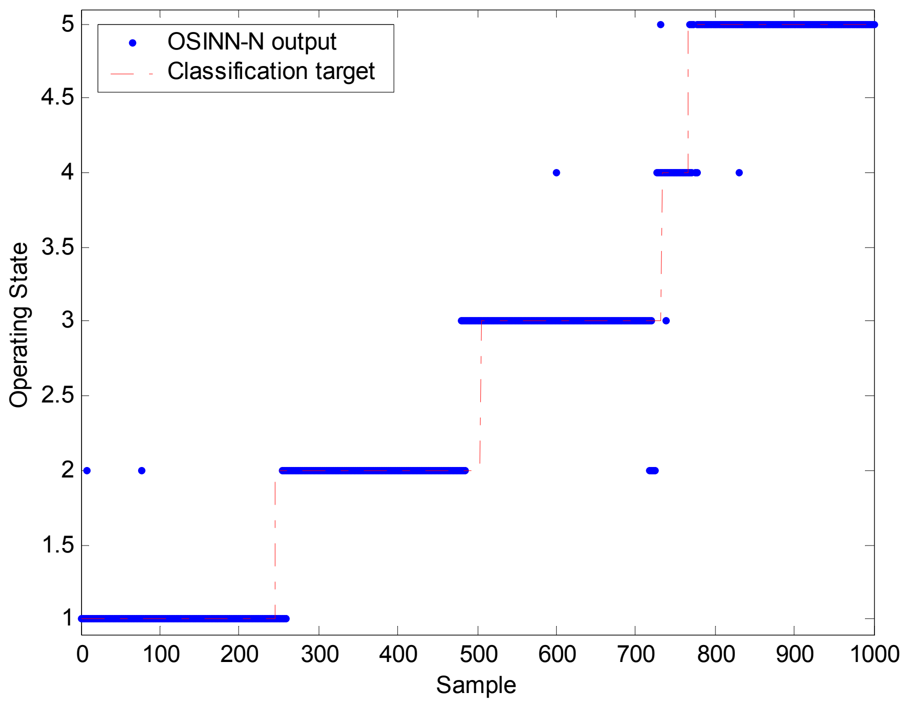

Figure 5-17: Operating state identification by OSINN-N in Fractionator section

5.4.4 Fault Detection during Air Blower Startup

One of the applications of state identification is to implement context-sensitive supervisory process monitoring. In this section, we illustrate context-based fault detection. During the startup of the air blower section, the same fault results in different signature (pattern) at different states of operation. Also, the same process pattern can reflect normal operation in one state, but be an indication of an abnormal situation at another. Context-based pattern recognition is therefore essential for accurate fault diagnosis.

During a normal startup, the air blower ΔP will drop after T_1^A when the regenerator overhead valve 16PV105 is opened. This can be seen in Figure 5-8 around sample 190. However, if the valve is stuck, 16PDI101 will remain at a high value. This situation is shown in Figure 5-18(a) which shows the profile of 16PDI101 during G_3 (which is the dataset containing fault). At the end of T_1^A (Sample 144 in G_3), 16PDI101 does not decrease and stays at a quasi-steady state around 0.6. This value for 16PDI101 by itself is not an indicator of a fault since during T_4^A this is considered normal (See Figure 5-8). In fact, all the seven variables in the air blower remains at a level which corresponds to PA_6 during normal operation and the fault cannot be detected if context information is not used.

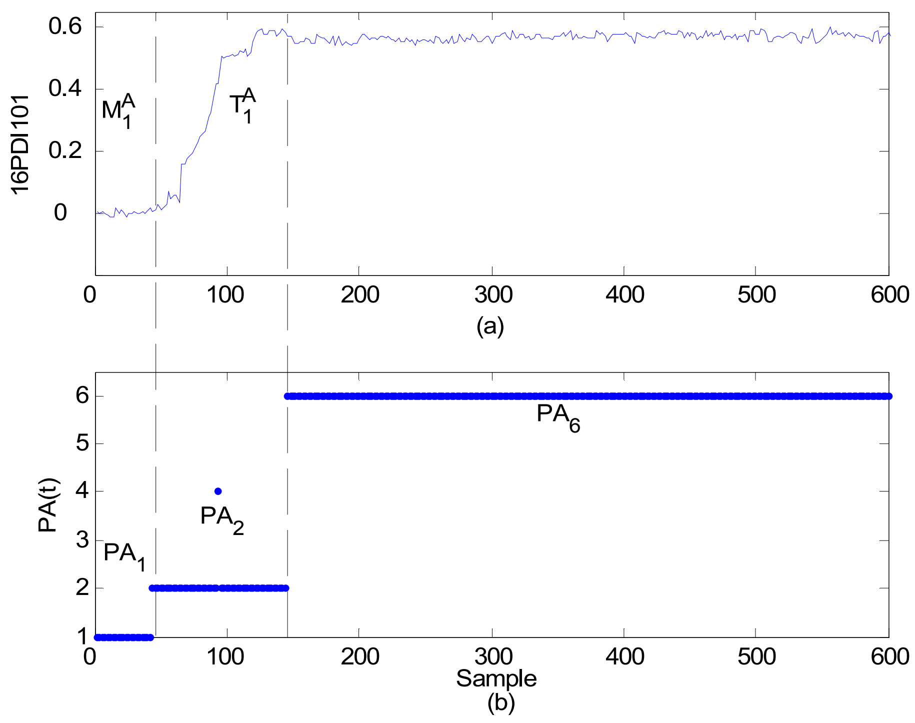

Figure 5-18: Example of valve 16PV105 fault (a) ΔP evolution in abnormal situation (b) process pattern identification by OSINN in abnormal situation

To use the OSINNs for fault detection, they have to be trained to signal a fault for patterns that do not arise during normal process evolution. To enable this, we use a RBF in the state identification block and exploit the RBF's local activation property which maps unknown patterns (that were not seen during training) to 0 (Srinivasan *et al.*, 2004b). The result of OSINN-P when used for abnormality detection is shown in Figure 5-19. In the figure, which shows only the first 600 samples for clarity, during the first 143 samples, when the process is evolving normally the network output tracks the current state of the process starting at State-1 and changing to State-2 at 56th sample. The abnormality is first detected by the state identification RBF at the 153rd sample and confirmed as $\hat{S}=0$ (abnormal operation) at the 160th sample after the dwell-time criterion has been satisfied. Subsequently, the output remains at zero indicating the abnormal evolution of the unit. The fault detection performance of

OSINN-S is similar to OSINN-P and is not shown here due to space constraints. The performance of OSINN-N is better as shown in Figure 5-20. The fault is detected and flagged at Sample 146. Fault detection is faster here since there is no context change in this architecture till state T_2^A and hence the dwell-time does not come into play.

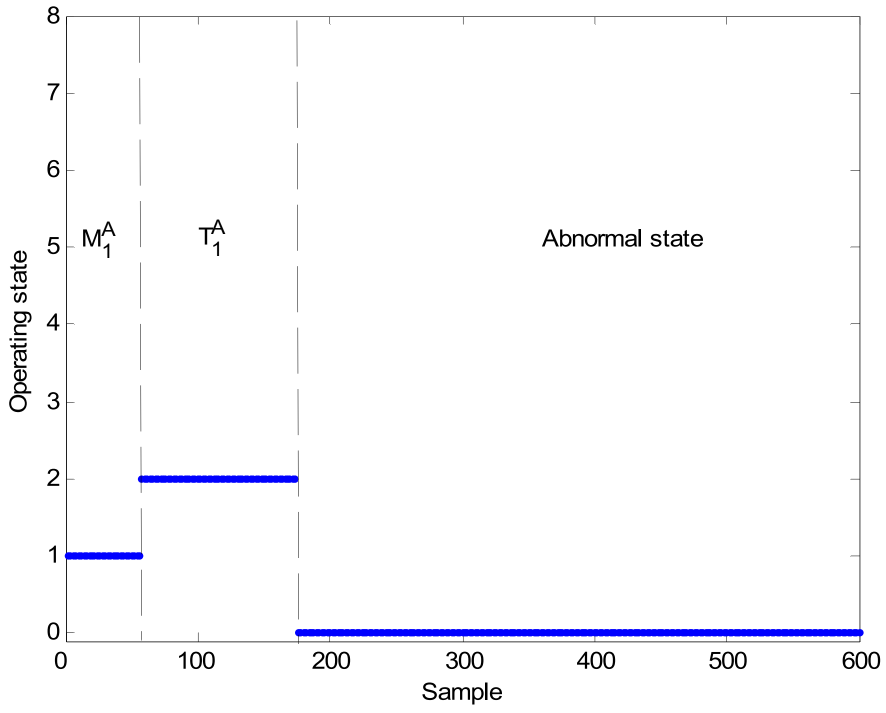

Figure 5-19: Fault detection by OSINN-P

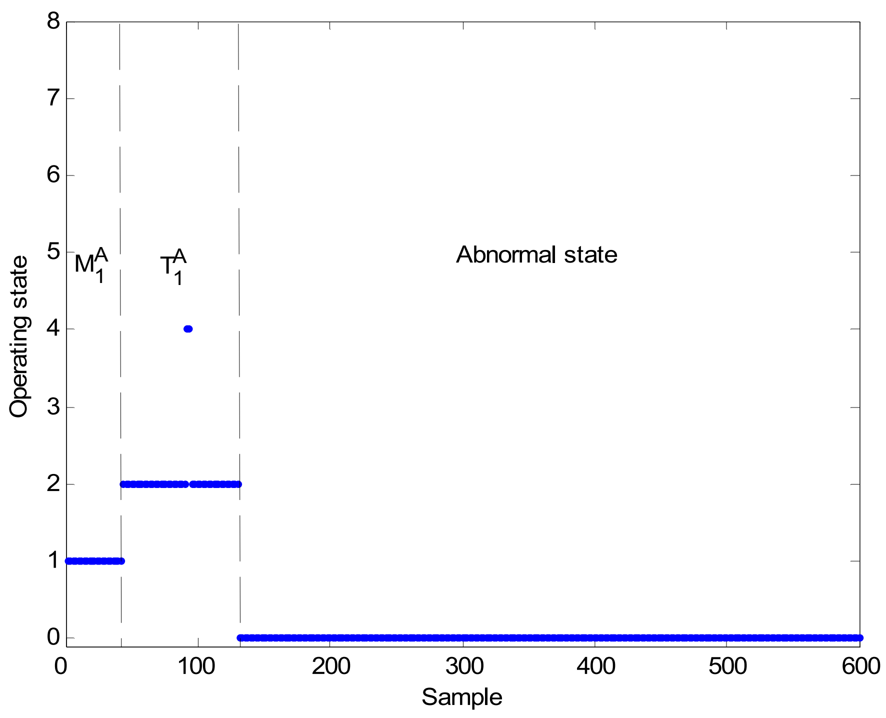

Figure 5-20: Fault detection by OSINN-N

5.5 Case Study 2: Operating State Identification in *P. Pastoris*

In this section, we describe the state identification for the fed-batch fermentation process described in Section 5.1. The reader is referred to (Muthuswamy and Srinivasan, 2003) for more details of the process. Eight variables are being measured online (in three minute intervals) in the process: Airflow, Stirrer speed, Dissolved oxygen (pO_2), Cumulative base addition, Cumulative acid addition, pH, Exit O_2 concentration, Exit CO_2 concentration. Further the carbon-dioxide evolution rate and O_2 uptake-rate are calculated by the control system based on the online measurements of O_2, CO_2. Data from 7 of these are used for phase identification; Cumulative acid and base addition and pH are not used here. Data from 6 runs of this lab-scale process were available (Muthuswamy, 2001). We have used data from Run SMB-74 to train the OSINNs and from Run SMB-78 to validate the results. It should be noted that the duration of the different states are not the same in the two runs; neither do the state changes occur at the same times. A list of the process states is shown TABLE 5-4. As seen in Figure 5-1, T_1^P, T_2^P, T_3^P and T_4^P have very similar process patterns (spike in pO_2) and are indistinguishable. M_2^P and M_3^P are also similar in terms of their trend and values. For accurate state identification, these five process patterns should be mapped to the nine operating states.

TABLE 5-4: Operating state of *P. pastoris* fermentation

Operating state	Process description
M_1^P	Batch-exponential-growth phase
T_1^P	Stirrer speed attains a predefined maximum
M_2^P	Airflow-assisted growth phase
T_2^P	Low substrate concentration causes a sharp increase in pO_2
M_3^P	Glycerol-fed-batch phase
T_3^P	Critical cell mass attained
M_4^P	Second-airflow-assisted growth phase
T_4^P	Low substrate concentration causes a sharp increase in pO_2
M_5^P	Methanol-fed-batch phase

A TDNN (structure 35-15-1) is used as the data-preprocessor for OSINN-P/S. As revealed by the structure, the input time delay is set to 5 min (i.e., 5 samples). The state identification block is a RBF. We set $\theta_P = 0.5$ and $\theta_S = 0.5$. The dwell-time T_d was set to 3 min since the states in the process last for short durations. The validation error for OSINN-P was found to be 6.34%. The error distribution during validation is shown TABLE 5-4. Figure 5-21 shows the classification output of OSINN-P. It is clear that the network can correctly identify the operating state even when the process patterns are identical in different operating states. For comparison, a TDNN with the structure of 35-15-1 and time delay of 5 min was trained. The classification error of this network is 17.18%. This shows the superiority of the OSINN when dealing with context-based patterns. In addition, this case study also clearly demonstrates that OSINN is robust to process measurement noise (see Figure 5-1 and Figure 5-21). Similar results were found for OSINN-S.

TABLE 5-5: Validation errors by OSINN-P for *P. pastoris* fermentation

	M_1^P	T_1^P	M_2^P	T_2^P	M_3^P	T_3^P	M_4^P	T_4^P	M_5^P
Lag error (N_{mis})	0	0	6	4	4	4	5	1	0
Classification error (N_{mis})	0	2	0	0	0	0	0	1	0

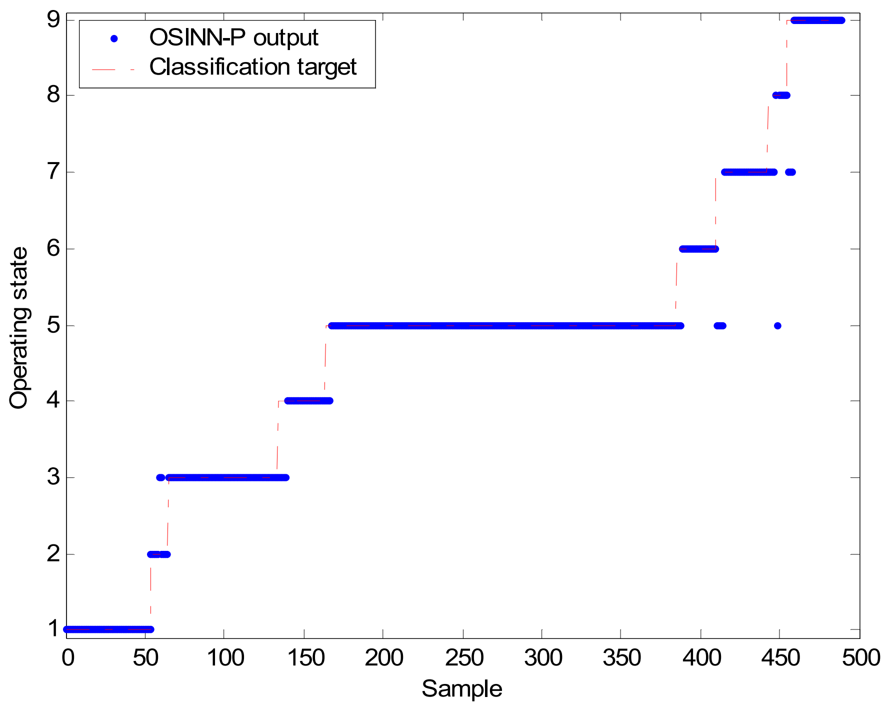

Figure 5-21: Operating state identification by OSINN-P in *P. pastoris*

OSINN-N was also tested for this case study. Three contexts were defined as shown in TABLE 5-6. With this, the four instances of PA_2 as well as two instances of PA_3 can be correctly mapped. With a TDNN (35-15-1) as the state identification block, the overall identification error is 8.38%. The operating state identification results are shown in Figure 5-22 and the error distribution in TABLE 5-7.

TABLE 5-6: Process patterns and corresponding operating states in *P. pastoris* fermentation

Process before T_2^P	*PA(t)*	PA_1	PA_2	PA_3	PA_2
	$\hat{S}(t)$	M_1^P	T_1^P	M_2^P	T_2^P
Process after T_2^P before M_4^P	*PA(t)*	PA_3	PA_2		
	$\hat{S}(t)$	M_3^P	T_3^P		
Process after T_4^P	*PA(t)*	PA_4	PA_2	PA_5	
	$\hat{S}(t)$	M_4^P	T_4^P	M_5^P	

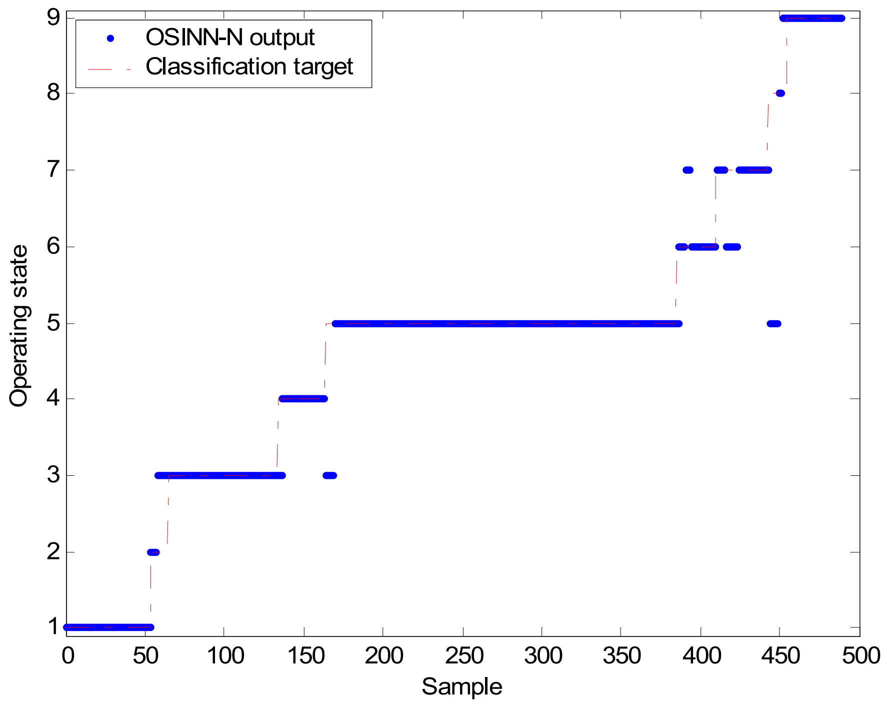

Figure 5-22: Operating state identification by OSINN-N in *P. pastoris*

TABLE 5-7: Validation errors by OSINN-N for *P. pastoris* fermentation

	M_1^P	T_1^P	M_2^P	T_2^P	M_3^P	T_3^P	M_4^P	T_4^P	M_5^P
Lag error (N_{mis})	0	0	3	0	2	1	1	6	0
Lead error (N_{mis})	0	0	0	0	6	0	0	0	3
Type-I error (N_{mis})	0	7	0	0	0	4	8	1	0

5.6 Conclusion

Situations where the same pattern maps to different classes suggest context-dependency. Traditional pattern recognition techniques perform poorly when the context changes and there is a dramatic change in the pattern mapping. Contextual features can be used in such cases to improve predictive performance of primary features. Context-based pattern recognition approaches exploit the ability of these contextual features to constrain reasoning, increase information content, and thus improve performance. In this chapter, we have shown that process state identification is a context-based pattern recognition problem. A neural network-based architecture — Operating State Identification Neural Network (OSINN) — is proposed for online state identification. In OSINN, the process state is recognized by a state identification block using multivariate temporal data from online sensors. The contextual feature is an additional input to the state identification which modifies the feature space so that the final mapping is reduced to the classical one-to-one or many-to-one situation. The contextual feature is regulated by a context manager which detects changes in context and controls the transfer of the feature to the state identification block.

The OSINN architecture can be implemented using different structures. Three suitable structures have been presented in this chapter. In OSINN-N, the contextual information is used to normalize the network's input so that process measurements from different contexts map to different regions of the input space. Through this contextual normalization strategy, process patterns from different contexts do not overlap and can be interpreted differently. OSINN-P and OSINN-S implement contextual expansion and realize context-based recognition in two steps. The process pattern is first identified by a data-preprocessor. The context manager detects the change in context by monitoring either *PA(t)* or $\hat{S}(t)$. The context is made available to the state identification block which uses it along with the process pattern to identify the process state. The additional network to identify process patterns makes OSINN-P and -S robust to measurement noise.

The context-based pattern recognition achieved by OSINN has several applications. We have shown that process supervision applications – specifically, process state identification and fault detection lead to context-dependent patterns. In the first case study, OSINN correctly identified the states arising during the startup of a simulated FCCU although the same process patterns occur in several states. We also show that abnormal situations during the startup that cannot be deciphered by normal pattern recognition approaches can be detected correctly using the proposed approach. In the second case study, the phases in a pilot-scale fed-batch fermentation process were identified accurately although the same pattern has different interpretations in different phases. The robustness of the OSINN to process noise as well as run-to-run variation is also highlighted in this case study.

The performance of OSINN is conditional on the correct detection of changes in the context. If the contextual feature used for state identification block is incorrect, the subsequent state identification will also be incorrect. Two noise-cancellation mechanisms – dwell-time and evaluation-interval – have been incorporated in OSINN to enhance context identification and management. The effect of the tuning parameters on state identification performance have been discussed in detail and guidelines developed to select suitable settings.

The improved performance from context-based pattern recognition comes at an additional cost – the selection of a suitable context is necessary. In the two case studies presented here, we have shown that this additional load is minimal since the previous process state from the same process unit can be used as the context. Thus, the only step that is additional to traditional pattern recognition approaches is the specification of context-change points. During the operation of large-scale processes, operators may incorporate information from not only the same section, but also other neighboring sections while interpreting process patterns. This is because exogenous variables from other process sections may have a substantial effect on the state of a process unit. In sections cases where external dependencies can be identified, the OSINN architecture can be extended to incorporate historical information from the neighboring sections as context. The implementation of this extension for large-scale multi-section processes is straightforward.

Chapter 6. Conclusions and Future Work

6.1 Conclusions

Supervisory control is an attractive strategy to optimize the control of nonlinear and batch processes. The identification of the process operating state can be used to localize the control configuration and parameters. The work in this thesis corroborates the growing importance of operating state identification in chemical and biology process and has focused on two main problems of process monitoring: clustering and classification.

Clustering is an offline technique that can identify the distinct operating states from historical process data. It can provide necessary information for online monitoring. A multivariate statistics-based methodology to cluster process states in historical operations data is proposed in this thesis. Process data are first segmented based on regions of steady state operations into modes and transitions. Similar modes are identified by comparing their means. A new dynamic PCA-based similarity factor, which accounts for the autoregressive nature, has been developed to cluster transitions. The technique was applied to data collected from two kinds of agile operation – startup of a simulated FCCU and multi-mode operation in the Tennessee Eastman plant simulation. Application to fault isolation was also demonstrated in the latter case study. In all cases, the method correctly identified and clustered the modes and transitions. These tests thus highlight the applicability of the state segregation and the superiority of the DPCA-based transition similarity factor.

The proposed clustering methodology offers several advantages. It accounts for the multivariate nature of chemical processes naturally and is hence superior to methods like DTW and qualitative trend comparison, which are designed for one-dimensional data. Also, the trend and the DTW-based transition comparison are computationally expensive and are therefore performed on selected key variables instead of all variables in a process unit. The selection of key variables for a process unit is a nontrivial problem which is eliminated in the proposed approach. The homologous problem of selecting *k*, the subset of the PCs, is a simpler one with clear guidelines. The use of PCs in lieu of the original variables also bestows the method with inherent noise filtering ability. In cases, where noise level is very high, the raw-data can be pretreated with wavelets or other filtering methods before clustering is performed. The normalization of process variables during transition comparison ensures that the focus is on the underlying dynamics. The magnitude of the change is correctly ignored thus making the comparison more robust to run-to-run differences. However, if a stricter comparison is necessary, magnitudes can also be included through the S_M factors as demonstrated in the TE case study. The DPCA factor results in a quantitative comparison of two transitions instead of a binary judgment. While this makes the similarity results dependent on the tuning parameters, the results are consistent for a range of parameter values. We have also developed clear guidelines for setting the parameter values. The computational requirement of this approach is quite modest and allows large amount of data to be analyzed in a short period.

After the process data are clustered, the data from different operating states can be used to build supervisory classifiers. Neural networks are used in this thesis to identify the operating state due to their nonlinear approximation ability and robustness to process measurement noise. While the training of the traditional neural networks to classify high dimensional temporal pattern is a challenging task, we present two neural network architectures which can achieve better classification performance and maintain lower computational requirements. In both structures, the original classification problem is decomposed into a number of simpler classification problems. The OVON uses a sub-state identification layer where a set of neural networks are used to identify simpler uni-variate, temporal patterns. A unification layer is subsequently used to infer the process state based on the sub-states, through multi-dimensional, static pattern recognition. The OCON incorporates a different philosophy: a state-identification layer is used to identify the presence or absence of a temporal pattern in multi-dimensions; the state of the process is inferred by analyzing the static, multi-dimensional outputs from the state-identification layer.

OVON and OCON are usually superior to traditional monolithic networks in terms of classification accuracy and training time. Each sub-network in OVON and OCON has a simpler structure than a monolithic neural network for the same problem. Generally, the OVON unification layer and the OCON regulator layer can filter out errors from the sub-layer and provides robustness to noise and disturbances. The OVON is superior in terms of training time, if retraining is necessary to accommodate sensor faults. The OCON is superior in terms of training time, if retraining is necessary to accommodate new operating states.

Situations where the same pattern maps to different classes suggest context-dependency. Traditional pattern recognition techniques perform poorly when the context changes and there is a dramatic change in the pattern mapping. Contextual features can be used in such cases to improve predictive performance of primary features. Context-based pattern recognition approaches exploit the ability of these contextual features to constrain reasoning, increase information content, and thus improve performance. In this thesis, we have shown that process state identification is a context-based pattern recognition problem. A neural network-based architecture —

Operating State Identification Neural Network (OSINN) — is proposed for online state identification. In OSINN, the process state is recognized by a state identification block using multivariate temporal data from online sensors. The contextual feature is an additional input to the state identification which modifies the feature space so that the final mapping is reduced to the classical one-to-one or many-to-one situation. The contextual feature is regulated by a context manager which detects changes in context and controls the transfer of the feature to the state identification block.

The OSINN architecture can be implemented using different structures. Three suitable structures have been presented in this thesis. In OSINN-N, the contextual information is used to normalize the network's input so that process measurements from different contexts map to different regions of the input space. Through this contextual normalization strategy, process patterns from different contexts do not overlap and can be interpreted differently. OSINN-P and OSINN-S implement contextual expansion and realize context-based recognition in two steps. The process pattern is first identified by a data-preprocessor. The context manager detects the change in context by monitoring either $PA(t)$ or $\hat{S}(t)$. The context is made available to the state identification block which uses it along with the process pattern to identify the process state. The additional network to identify process patterns makes OSINN-P and -S robust to measurement noise.

The context-based pattern recognition achieved by OSINN has several applications. We have shown that process supervision applications – specifically, process state identification and fault detection lead to context-dependent patterns. In the ShadowPlant case study (chapter 5), OSINN correctly identified the states arising during the startup of a simulated FCCU although the same process patterns occur in several states. We also show that abnormal situations during the startup that cannot be deciphered by normal pattern recognition approaches can be detected correctly using the proposed approach. In the fed-batch case study, the phases in a pilot-scale fed-batch fermentation process were identified accurately although the same pattern has different interpretations in different phases. The robustness of the OSINN to process noise as well as run-to-run variation is also highlighted in this case study.

The performance of OSINN is conditional on the correct detection of changes in the context. If the contextual feature used for state identification block is incorrect, the subsequent state identification will also be incorrect. Two noise-cancellation mechanisms – dwell-time and evaluation-interval – have been incorporated in OSINN to enhance context identification and management. The effect of the tuning parameters on state identification performance have been discussed in detail and guidelines developed to select suitable settings.

6.2 Suggestions for Future Work

While the developments in this thesis solve the overall problems in the state identification, these can be extended in the future.

The DPCA operation on process data provides a lot of information of the process profile. Besides reflecting the distribution of the process variable dynamics through the direction of main PCs, it also explicitly gives the resources from which the largest variations of the process come from by the coefficients of the loadings. How to utilize this information for transition identification is an interesting problem. And through these studies, the underlying mechanism of DPCA operation can be understood better.

DPCA similarity factor shows great potentials in pattern comparison. Its implementation in other possible field such as fault detection, model identification is a good study direction. For example, most modern industrial control algorithms need an explicit process mathematic model. Step test is the main method to get these models in practical applications. If these models can be derived from historical dataset, the model identification can be tremendously facilitated. However, the historical dataset is usually very large. How to find a proper period of data for model identification is then a challenge work. It may be possible to build a library including data patterns that have ideal features for model identification. Then DPCA similarity factor can be used to locate proper period of data from historical dataset.

6.2.1 OVON and OCON Structures

OVON and OCON have shown better classification performance than traditional neural network. However, the improved classification accuracy of OVON is derived at an additional cost. In order to train the sub-state identification layer, a prior knowledge of sub-states of each variable is necessary. The uni-variate nature makes this a straightforward step when the boundaries between clusters can be located accurately. However, for problems with a large number of variables, this analysis is cumbersome. In both OVON and OCON (and in traditional networks), misclassification occurs during state change where there is no clear separation between the states or the sub-states. A better method for accurate state boundary recognition is therefore needed and is the subject of our future work.

6.2.2 Context Recognition Problem

The improved performance from context-based pattern recognition comes at an additional cost – the selection of a suitable context is necessary. In the two case studies presented in chapter 5, it is shown that this additional load is minimal since the previous process state from the same process unit can be used as the context. Thus, the only step that is additional to traditional pattern recognition approaches is the specification of context-change points. During the operation of large-scale processes, operators may incorporate information from not only the same section, but also other neighboring sections while interpreting process patterns. This is because exogenous variables from other process sections may have a substantial effect on the state of a process unit. In sections cases where external dependencies can be identified, the OSINN architecture can be extended to incorporate historical information from the neighboring sections as context. The implementation of this extension for large-scale multi-section processes is straightforward.

Bibliography

Ahmed, M. N., Yamany, S. M., Nevin, M., Farag, A. A. and Moriarty, T. (2002), "A modified fuzzy C-means algorithm for bias field estimation and segmentation of MRI data", IEEE Transaction on Medical Imaging, Vol. 21, No. 3, pp 193-199.

Anderberg, M. R. (1973), "Cluster analysis for applications", Academic Press, Inc., New York, NY.

Anshuman, B., Srinivasan, R. and Krishnaswamy, P. R. (2003), "Fault detection during process transitions: a model-based approach", Chemical Engineering Science, Vol. 58, pp. 309-325.

Arnold, M. W. and Darius, I. H. (1989), "Alarm management in batch process control", ISA Transactions, Vol. 28, Issue 3, pp. 33-40.

Baltzakis, H. and Papamarkos, N. (2001), "A new signature verification technique based on a two-stage neural network classifier", Engineering Applications of Artificial Intelligence, Vol. 14, No. 1, pp. 95-103.

Bambang, P., Deneault, L. G. and Denault, A. Y. (2001), "Detection of hemodynamic changes in clinical monitoring by time-delay neural networks", International Journal of Medical Informatics, Vol. 63, No. 1-2, pp. 91-99.

Baughman, D. R. and Liu, Y. A. (1995), "Neural networks in bio-processing and chemical engineering", San Diego: Academic Press.

Bengio, Y. (1993), "A connectionist approach to speech recognition", International Journal of Pattern Recognition and Artificial Intelligence, Vol. 7, No. 4, pp. 647-667.

Bhagwat, A., R. Srinivasan and Krishnaswamy, P.R. (2003), “Fault detection during process transitions: a model-based approach”, Chemical Engineering Science, Vol. 58, Issue 2, pp. 309 – 325.

Brezillon, P. (1999), “Context in problem solving: a survey”, The Knowledge Engineering Review, Vol. 14, Issue 1, pp. 47-80.

Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., Cun, Y. L., Moor, C., Sackinger, E. and Shah, R. (1993), “Signature verification using a ‘Siamese’ time delay neural network”, International Journal of Pattern Recognition and Artificial Intelligence, Vol. 7, No. 4, pp. 669-688.

Brown, P. R. and Rhinehart, R. R. (2000), “Automated steady-state identification in multivariable systems”, Hydrocarbon Processing, Vol. 79, Issue 9.

Bulsari, A.B. (1995), “Neural networks for chemical engineers”, New York: Elsevier.

Burges, C. J. C., Ben, J. I., Denker, J. S., Cun, Y. L. and Nohl, C. R. (1993), “Off-line recognition of handwritten postal words using neural networks”, International Journal of Pattern Recognition and Artificial Intelligence, Vol. 7, No. 4, pp. 689-704.

Cao, S. l. and Rhinehart, R. R. (1995), “An efficient method for on-line identification of steady state”, Journal of Process Control, Vol. 5, No. 6, pp. 363-374.

Chen, B. H., Wang, X. Z., Yang, S. H. and McGreavy, C. (1999), “Application of wavelets and neural networks to diagnostic system development, 1, feature extraction”, Computers and Chemical Engineering, Vol. 23, pp. 899-906.

Chen, G. Q., Wei, Q. and Zhang, H. (2001), “Discovering similar time-series patterns with fuzzy clustering and DTW methods”, IEEE Transactions on Neural Networks, Vol. 9, No. 5, pp. 2160-2164.

Chen, J. H. and Liu, K. C. (2002), "On-line batch process monitoring using dynamic PCA and dynamic PLS models", Chemical Engineering Science, Vol. 57, pp. 63-75.

Chen, S., Cowan, C. F. N., and Grant, P. M. (1991), "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks", IEEE Transaction on Neural Networks, Vol. 2, No. 2, pp.302-309.

Clancey, W. J. (1983), "The epistemology of a rule-based expert system: A framework for explanation", Artificial Intelligence Journal, Vol. 20, Issue 3, pp. 197-204.

Douglas, C. Montgomery and George, C. Runger (1994), "Applied statistics and probability for engineers", New York: John Wiley and Sons.

Downs, J. J. and Vogel, E. F. (1993), "A plant-wide industrial process control problem", Computers and Chemical Engineering, Vol. 17, No. 3, pp.245-255.

Drucker, H., Schapire, R. and Simard, P. (1993), "Boosting performance in neural networks", International Journal of Pattern Recognition and Artificial Intelligence, Vol. 7, No. 4, pp. 705-719.

Elman, J. L. (1990), "Finding structure in time", Cognitive Science, Vol. 14, pp. 179-211.

Eltoft, T. and De Figueiredo, R. J. P. (2001), "A new neural network for cluster detection and labeling", IEEE Transactions on Neural Networks, Vol. 9, No. 5, pp. 1021-1035.

Gollmer, K. and Posten, C. (1995), "Pattern recognition for phase detection in bioprocesses", Computer Applications in Biotechnology: a Postprint Volume from the 6th International Conference, Garmisch-Partenkirchen, Germany, pp. 41-47.

Gupta, L., McAvoy, M. and Phegley J. (2000), "Classification of temporal sequences via prediction using the simple recurrent neural network", Pattern Recognition, Vol. 33, pp. 1759-1770.

Hecht-Nielsen, R. (1988), "Theory of back-propagation neural network", Proceedings of International Neural Network Society Annual Meeting, September.

Hecht-Nielsen, R. (1989), "Theory of back-propagation neural network", Proceedings of the International Joint Conference on Neural Networks, pp. 593-608.

Hornik, K., Stinchcombe, M. and White, H. (1989), "Multilayer feedforward neural networks are universal approximators", Neural Networks, Vol. 2, No. 5, pp. 359-366.

Honeywell Inc, (2000), “Guide to the Fluidized Catalytic Cracking Unit Standard Model”.

Hunter, A. (2001), “A default logic based framework for context-dependent reasoning with lexical knowledge”, Journal of Intelligent Information Systems, Vol. 16, pp. 65-87.

Jain, A. K., Murty, M. N. and Flynn, P. J. (1999), “Data clustering: A review”, ACM Computing Surveys, Vol. 31, No. 3.

Jan, S., Frank, J. and Gustavo, D. (2001), "Temporal clustering with spiking neurons and dynamic synapses: Towards technological applications", Neural Networks, Vol. 14, pp. 275-285.

Jensen, L. D. (1997), “Need for dynamic configuration and other augmentation of distributed control systems for improved alarm management”, IEE Colloquium (Digest).

Jiang, T. W., Chen, B. Z., He, X. R. and Stuart, P., (2003), “Application of steady-state detection method based on wavelet transform”, Computers and Chemical Engineering, Vol. 27, pp569-578.

John, Peter W. M. (1990), “Statistical methods in engineering and quality assurance”, John Wiley and Sons, Inc.

Jolliffe, I. T. (1986), “Principal component analysis”, Springer-verlag New York Inc.

Josef, K. and Fabio, R. (2000), “Multiple classifier systems“, Proceedings of First International Workshop, MCS 2000, Cagliari, Italy, June 21-23, Springer.

Jou, IC., Hu, MS. and Juang, YT. (1992), “Mandarin syllables recognition based on One Class One Net neural network with modified selective update algorithm”, IEEE International Workshop on Intelligent Signal Processing and Communication Systems, pp. 577-591.

Jou, JL., Jyh-Bin Ren, Chao-Sen Wang and Bin-Chang Chieu (1995), “One class one network with fuzzy number input and fuzzy number output for face recognition”, Proceedings of Neural, Parallel and Scientific Computations,Vol.1, Proceedings of the First International Conference, pp. 227-232.

Kassidas, A., MacGregor, J. F. and Taylor, P. A. (1998), “Synchronization of batch trajectories using dynamic time warping”, AIChE Journal, Vol. 44, Issue 4, pp 864-875.

Katz, A. J., Gately, M. T. and Collins, D. R. (1990), “Robust classifiers without robust features”, Neural Computation, Vol. 2, pp. 472-479.

Kavchak, M. and Budman, H. (1999), “Adaptive neural network structures for non-linear process estimation and control”, Computers and Chemical Engineering, Vol. 23, pp. 1209–1228.

Kettebekov, S. and Sharma, R. (1999), "Toward multimodal interpretation in a natural speech/gesture interface", Proceedings of International Conference on Information Intelligence and Systems, pp. 328-335.

Khalilian, M. and Dhib, R. (2001), “Online identification and control of a fluidized catalytic cracking unit”, Proceedings of the IASTED International Conference Modeling, Identification, and Control, pp. 448-452.

Kim, E.-K., Wu, J.-T., Tamura, S., Sato, Y., Close, R., Taketani, H., Kawai, H., Inoue, M. and Ono, K. (1993), “Comparison of neural networks and K-NN classification methods in vowel and patellar subluxation image recognitions”, International Journal of Pattern Recognition and Artificial Intelligence, Vol. 7, No. 4, pp. 775-782.

Klaus, P., Jens, K. and Klaus-robert, M. (1996), “Annealed competition of experts for a segmentation and classification of switching dynamics”, Neural Computation, Vol. 8, pp. 340-356.

Kreinovich, V. K. (1991), "Arbitrary nonlinearity is sufficient to represent all functions by neural networks: A theorem", Neural Networks, Vol. 4, pp. 381-383.

Krzanowski, W. J. (1982), “Between-group comparison of principal components-some sampling results”, Journal of Statistical Computation and Simulation, Vol. 15, Issue 2-3, pp. 141-154.

Ku, W., Storer, R. H. and Georgakis, C. (1995), “Disturbance detection and isolation by dynamic principal component analysis”, Chemometrics and Intelligent Laboratory Systems, Vol. 30, pp. 179-196.

Lawrence, M. Jr. (1970), “Applied engineering statistics for practicing engineers”, Barnes and Noble, Inc. New York.

Levin, E., Pieraccini, R. and Bocchieri, E. (1993), “Time-warping network: A neural approach to hidden Markov model-based speech recognition”, International Journal of Pattern Recognition and Artificial Intelligence, Vol. 7, No. 4, pp. 783-799.

Li, H. and Wang, J. (1993), “Computing optical flow with a recurrent neural network”, International Journal of Pattern Recognition and Artificial Intelligence, Vol. 7, No. 4, pp. 801-814.

Li, W. and Nasrabadi, N. M. (1993), "Invariant object recognition based on a neural network of cascaded RCE Nets", International Journal of Pattern Recognition and Artificial Intelligence, Vol. 7, No. 4, pp. 815-829.

McAvoy T. J. and N. Ye, (1994), "Base control for the Tennessee Eastman problem", Computers and Chemical Engineering, Vol.18, Issue 5, pp.383-413.

Martin, G. R. (2001), "Application of a time-delay neural network to promoter annotation in the drosophila melanogaster genome", Computers and Chemistry, Vol. 26, No. 1, pp. 51-56.

Maskery, H and Meads, J. (1992), "Context: In the eyes of users and in computer systems", SIGCHI Bulletin, Vol. 24, Issue 2, pp. 12-21.

Mathworks (2002), Matlab Neural Network Toolbox.

Moore, R. L. (1997). "Distributed embedded intelligence blazing processing trails", InTech, Vol. 44, Issue 3, pp 50-53.

Mozer, M. C. (1992), "Neural net architectures for temporal sequence processing", Time Series Prediction: Forecasting the Future and Understanding the Past, pp. 243-264.

Muthuswamy, K. and Srinivasan, R. (2003), "Phase-based supervisory control for fermentation process development", Journal of Process Control, Vol. 13, No. 5, pp 367 – 382.

Muthuswamy, K. (2001), "A feature recognition approach to control and supervision of fermentation processes", MEng Thesis, National University of Singapore.

Pramit, S. and Raghunathan, R. (2000), "Multivariable gain-scheduled fuzzy logic control of a fluidized catalytic cracker unit", Computers and Chemical Engineering, Vol. 24, Issues 2-7, pp1083-1089.

Pravdova, V., Walczak, B. and Massart, D. L. (2002), “A comparison of two algorithms for warping of analytical signals”, Analytica Chimica Acta, Vol. 456, Issue 1, pp. 77-92.

Rosen, C. and Yuan, Z. (2001), “Supervisory control of wastewater treatment plants by combining principal component analysis and fuzzy c-means clustering”, Water Science and Technology, Vol. 43, pp. 147-156.

Roverso, D. (2000), “Soft computing tools for transient classification“, Information Sciences, Vol. 127, No. 3, pp. 137-156.

Sekhar, C. C. and Yegnanarayana, B. (1996), “Neural network models for spotting stop consonant-vowel (SCV) segments in continuous speech”, IEEE International Conference on Neural Networks, Vol. 4, pp. 2003-2008.

Schalkoff, R. J. (1992), “Pattern recognition: statistical, structural, and neural approaches”, John Wiley and Sons, Inc.

Sebzalli, Y. M. and Wang, X. Z. (2001), “Knowledge discovery from process operational data using PCA and fuzzy clustering”, Engineering Application Artificial Intelligence, Vol. 14, No. 5, pp. 607-616.

Singhal, A. and Seborg, D. E. (2001), “Matching patterns from historical data using PCA and distance similarity factors”, Proceedings of the American Control Conference, pp. 1759-1764.

Srinivasan, R., Vedam, H. and Nochur, A. (2001), “Characterizing operating states in chemical plants: application to a fluidized catalytic cracking unit”, Presented in the AIChE annual meeting, Reno, Nov 4–9.

Srinivasan, R. (2002), “Efficiently managing transitions in chemical plants”, Technical Report, Laboratory for Intelligent Applications in Chemical Engineering, Department of Chemical and Environmental Engineering, National University of Singapore.

Srinivasan, R., C. Wang, W. K. Ho and K. W. Lim (2004a), “Dynamic Principal Component Analysis Based Methodology for Clustering Process States in Agile Chemical Plants”, Industrial and Engineering Chemistry Research, Vol.43, Issue 9, pp. 2123 - 2139.

Srinivasan, R., C. Wang, W. K. Ho and K. W. Lim (2004b), “Neural Network Systems for Multi-dimensional Temporal Pattern Classification”, Computers and Chemical Engineering, Accepted.

Sundarraman, A. and R. Srinivasan (2003), “Monitoring Transitions in Chemical Plants using Enhanced Trend Analysis”, Computers & Chemical Engineering, Vol. 27, Issue 10, pp. 1455–1472.

Stiles, Bryan W. and Ghosh, Joydeep (1997), “Habituation based neural networks for spatio-temporal classification”, Neurocomputing, Vol. 15, Issue 3-4, pp. 273-307.

Tsay, SC., Hong, PR. and Chieu, BC. (1992), “Handwritten digits recognition system via OCON neural network by pruning selective update”, 11th IAPR International Conference on Pattern Recognition Conference B: Pattern Recognition Methodology and Systems, Vol. 2, pp. 656-659.

Tsai, C.-S., Chang, C.-T. and Chen, C.-S. (1996), "Fault detection and diagnosis in batch and semi-batch processes using artificial neural networks", Chem. Eng. Comm., Vol. 143, pp. 39-71.

Turney, P. D. (1993a), “Exploiting context when learning to classify”, Proceedings of the European Conference on Machine Learning, ECML-93, pp. 402-407.

Turney, P. D. (1993b), “Robust classification with context-sensitive features”, In Industrial and Engineering Application of Artificial Intelligence and Expert systems, IEA/AIE-93, pp. 268-276. Edinburgh, Scotland: Gordon and Breach.

Turney, P. and Halasz, M. (1993), “Contextual normalization applied to aircraft gas turbine engine diagnosis”, Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies, Vol. 3, Issue 2, pp. 109-129.

Valle, S., Li, W.H. and Qin, S.J., (1999), “Selection of the number of principal components: the variance of the reconstruction error criterion with a comparison to other methods”, Industrial and Engineering Chemistry Research, Vol. 38, Issue 11, pp. 4389 – 4401.

Wang, C., Srinivasan, R., Ho, W. K., and Lim, K. W. (2000), “Dynamic alarm configuration in chemical processes through process mode identification”, Proceedings of ASCC'2000, Shanghai, pp. 1.

Wang, D. and Arbib, M. A. (1990), “Complex temporal sequence learning based on short-term memory”, Proceedings of IEEE, Vol. 78, pp. 1536-1544.

Wang, X. Z., Chen, B. H., Yang, S. H. and McGreavy, C. (1999), “Application of wavelets and neural networks to diagnostic system development, II, an integrated framework and its application”, Computers and Chemical Engineering, Vol. 23, pp. 945-954.

Wohler, C. and Anlauf, J. K. (2001), “Real-time object recognition on image sequences with the adaptable time delay neural network algorithm-applications for autonomous vehicles”, Image and Vision Computing, Vol. 19, No. 9-10, pp. 593-618.

Wolfgang, M. (1997), "Networks of spiking neurons: The third generation of neural network models", Neural Networks, Vol. 10, No. 9, pp. 1659-1671.

Zullo, L. (1996), "Validation and verification of continuous plants operating modes using multivariate statistical methods", Computers and Chemical Engineering, Vol. 20, Part A, pp. 683-688.

Author’s Publications

Srinivasan, R., C. Wang, W. K. Ho and K. W. Lim (2004), “Dynamic Principal Component Analysis Based Methodology for Clustering Process States in Agile Chemical Plants”, Industrial and Engineering Chemistry Research, Vol.43, Issue 9, pp. 2123 - 2139.

Srinivasan, R., C. Wang, W. K. Ho and K. W. Lim (2004), “Neural Network Systems for Multi-dimensional Temporal Pattern Classification”, Computers and Chemical Engineering, Accepted.

Srinivasan, R., C. Wang, W. K. Ho and K. W. Lim (2004), “Context-based recognition of process states using neural networks”, Chemical Engineering Science, Accepted.

Wang C., R. Srinivasan, W. K. Ho and K. W. Lim (2000), “Dynamic Alarm Configuration in Chemical Processes through Process Mode Identification”, The Proceedings of ASCC'2000, Shanghai, Page. 1.

Wang C., R. Srinivasan, W. K. Ho and K. W. Lim (2000), “A Hierarchical Neural Network Structure For Process Mode Classification”, The Proceedings of CPEC & RSCE’2000, Singapore.

Wang C., R. Srinivasan, W. K. Ho and K. W. Lim (2001), “Neural Network-Based Sequential Pattern Recognition for Process Mode Identification”, The Proceedings of AIChE Annual Meeting.

Wang C., R. Srinivasan, W. K. Ho and K. W. Lim (2002), “PCA Clustering of Process States for Control of Agile Chemical Plants”, The Proceedings of AIChE Annual Meeting.

Wang C., R. Srinivasan, W. K. Ho and K. W. Lim (2002), “A One-Class-One-Net Neural Network-Based Structure for Operating State Identification”, The Proceedings of ASCC'2002, Singapore.