PROCESS MODELING

AND OPTIMIZATION USING

ARTIFICIAL INTELLIGENCE AND

MACHINE LEARNING FORMALISMS

THESIS SUBMITTED TO THE

UNIVERSITY OF PUNE

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

CHEMICAL ENGINEERING

BY

Yogesh Pandurang Badhe

UNDER THE GUIDANCE OF

Dr. B. D. Kulkarni

Chemical Engineering and Process Development Division

National Chemical Laboratory

Dr. Homi Bhabha Road

Pune 411008

India

January 2008

राष्ट्रीय रासायनिक प्रयोगशाला
(वैज्ञानिक तथा औद्योगिक अनुसंधान परिषद)
डॉ. होमी भाभा मार्ग पुणे – 411 008. भारत
NATIONAL CHEMICAL LABORATORY
(Council of Scientific & Industrial Research)
Dr. Homi Bhabha Road, Pune - 411 008. India.

CERTIFICATE

This is to certify that the work incorporated in the thesis, "PROCESS MODELING AND OPTIMIZATION USING ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING FORMALISMS" submitted by Mr. Badhe Yogesh Pandurang, for the Degree of Doctor of Philosophy, was carried out by the candidate under my supervision in the Chemical Engineering and Process Development Division, National Chemical Laboratory, Pune – 411 008, India. Such material as has been obtained from other sources has been duly acknowledged in the thesis.

Dr. B. D. Kulkarni

(Research Advisor)

DECLARATION

I hereby declare that the thesis "PROCESS MODELING AND OPTIMIZATION USING ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING FORMALISMS" submitted for the degree of Doctor of Philosophy to the University of Pune has not been submitted by me for a degree to any other University.

Badhe Yogesh Pandurang

Dedicated to

AISG, NCL, Pune.

ACKNOWLEDGEMENT

I take this opportunity with deep sense of gratitude to record my sincere thanks to my research supervisor Dr. B. D. Kulkarni for introducing me to a fascinating and challenging frontier of artificial intelligence and machine learning that has brought a positive turning point in my career. I remain deeply indebted to him for his precious advice, his caring attention, for helping me out in most stressful situations, for giving me the edge to meet future challenges...

My heartfelt thanks are due to Dr. S. S. Tambe for his keen interest, valuable suggestions, personal care and for helping me in all possible ways to comprehend my work in its present form. I will be grateful to him also for giving me exposure to the exciting field of time series analysis.

Colleagues and Staff at National Chemical Laboratory (NCL), Pune were quite supportive throughout my stay. Particularly I am indebted to Kiran, Vinay, Uttam, Jeevan, Ravikumar, Jayaram, Savita, Somanath, Avinash, Prakash, Abhijeet, Nilesh, Sudheerkumar, Kalpendra, for making my stay at NCL pleasant and memorable. Technical discussions with Kiran, Vinay, Uttam were really useful.

I am thankful to the teachers at M. J. College and SMIT Polytechnic, Jalgaon for orienting my mind towards research during the school and undergraduate days itself. I am also grateful to Mr. S. L. Pandharipande, Dr. R. L. Sonolikar at LIT, Nagpur for their valuable guidance and encouragement during my career development.

I would also like to express my sincere thanks to the numerous anonymous referees who have reviewed parts of this work prior to publication in journals and whose valuable comments have contributed to the clarification of many of the ideas presented in this thesis.

Whatever I am and whatever I will be in future is because of the goodwill and unstinted support that I have received from my parents. Their constant encouragement, sacrifice and support made me achieve the goal. I owe them a lot

for which mere words are insufficient. In addition, I will be thankful to my loving wife for putting up with having a 'student' husband.

I would also like to thank Council for Scientific and Industrial Research (CSIR), New Delhi for granting me Senior Research Fellowship.

Badhe Yogesh Pandurang

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

During the past decade, factors such as global competition, stringent environment protection laws, emphasis on high and consistent product quality and greater product variation have created a challenging environment for process manufacturing. Some of these challenges include fast technical innovation, demand for reduction of design and production of lead times, etc. To this end, in recent years, modeling, simulation and optimization of chemical processes have been employed rigorously both in academia and industry. Conventionally, modeling is conducted using phenomenological approach. This method works very well for processes wherein underlying phenomena are fully understood and measurements thereof are available. Such an approach however can fail miserably for those processes in which the complex nonlinear interactions of different competing mechanisms make it impossible to build even a rudimentary model. Thus, for instance, in microbial fermentations, it is impossible to measure the intracellular variables accurately and hence phenomenological descriptions may fail to capture the underlying features over the ranges of governing parameters. The above discussion highlights the need for a paradigm shift in the approach to modeling of chemical processes.

To address the issue alluded to above, in the present thesis, artificial intelligence and machine learning formalisms are adopted and suitably tailored to build data-driven models that are employed successfully for tasks such as steady state and dynamic modeling, process identification, fault detection diagnosis and dimensionality reduction. This thesis is structured as follows.

Chapter 1 provides a brief overview of various artificial intelligence (AI) and machine learning (ML) formalisms, viz. neural networks, expert systems, fuzzy logic, tabu search and evolutionary algorithms such as genetic algorithms, memetic algorithms and genetic programming. The application domain of these formalisms has also been clearly spelt out.

The second chapter described in detail the AI and ML based algorithms used in the thesis along with the pertinent literature survey. Depending upon their applications, these are classified into four categories namely *modeling*,

classification, *optimization* and *data reduction/projection* formalisms. A number of formalisms covered under modeling category are error back propagation neural networks, resilient-back propagation neural network, general regression neural network, radial basis function neural network, support vector regression and genetic programming. The formalisms covered under classification are K-means clustering and self-organizing maps. The formalisms detailed under optimization are simultaneous perturbation stochastic approximation, tabu search, genetic algorithms and memetic algorithms. Finally, the formalisms considered for dimensionality reduction and data projection are principle component analysis, curvilinear component analysis, auto associative neural networks, locally linear embedding, Sammons mapping and artificial neural network-based Sammon's mapping and fuzzy curves and surfaces.

The third chapter provides details of a number of AI and ML based modeling studies. In the first study, ANN based softsensors are developed for monitoring the polyethylene manufacturing process. A recently proposed algorithm of noise superimposition based data enlargement has been shown to be useful in accurately predicting the magnitudes of process variables. In the next study, the same algorithm has been used for a chemical process to demonstrate the algorithm's effectiveness in a number of systems. Next, an ANN based model is constructed for the improved evaluation of Indian coals. Also, SVR models are proposed as softsensors for the prediction of biochemical batch process variables. In the subsequent section, SVR is once again employed for the identification of the biochemical treatment of polluted waste water process. In the last section of chapter 2, a novel AI-based formalism, namely genetic programming, has been used for the prediction of selectivity and yield of a cumene production process. Specifically, a modified GP algorithm augmented with local search is used in this study.

Chapter 4 focuses on the classification/clustering studies. It deals with the cluster analysis of a number of process faults that can occur in batch fermentation processes namely, protein synthesis and citric acid production. Also demonstrated is an application of an artificial intelligence based clustering method, namely, self-organizing map (SOM) for the classification of biochemical batch process data. The said study aims at exploring the efficacy of SOM for classification

applications involving nonlinear projection of a high dimensional input space onto a low, i.e. two dimensional (2-D) projected space to diagnose faulty batches. Here, a case study involving the biosynthesis of protein has been conducted to illustrate SOM's efficacy in process monitoring and fault detection and diagnosis applications.

The fifth chapter deals with process optimization studies. An artificial neural network based process model is developed first from the process data of protein synthesis and citric acid production. The input space of this ANN model representing process operating variables is then optimized using the memetic algorithm formalism with a view of simultaneously optimizing multiple process output variables and thereby improving the process performance. Also, the results of the memetic algorithm-based optimization have been compared with those obtained in an earlier study using the genetic algorithm formalism.

Chapter 6 comprises case studies demonstrating the effectiveness of AI-based novel algorithms for low-dimensional projection of process data, feature extraction and dimensionality reduction. In the first section of the chapter, a recently proposed neural network based Sammon's mapping is utilized for the dimensionality reduction of glass data and fault detection and diagnosis of a continuous stirred tank reactor (CSTR). In the second section, locally linear embedding formalism is demonstrated for the fault detection and diagnosis of the invertase production process. Further, curvilinear component analysis method is utilized for the fault detection and diagnosis of a CSTR and invertase production process. In the last study of this chapter, fuzzy curves and surfaces approach is used for the selection of important model inputs of heat exchanger and pH control processes.

Finally, the seventh chapter of this thesis provides a summary of important results and conclusions drawn from the case studies performed in chapters 3 to 6. Also, this chapter highlights the guidelines on the deployment of AI-based formalisms.

CHAPTER 1

INTRODUCTION TO ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING FORMALISMS

1.1 WHAT IS ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING?

Artificial intelligence (AI) is a branch of computational science, which develops mathematical algorithms mimicking various kinds of intelligent behaviour exhibited by biologically evolving species, with the aim of providing novel and efficient solutions to complex modeling, classification and optimization problems. However, the AI does not have to confine itself to methods that are observed only in the nature. Accordingly, machine learning algorithms are also considered to be part of the AI. The intelligent behaviour exhibited by biologically evolving species involves perception, reasoning, learning, communication, decision making and acting in complex environments.

AI has been one of the most controversial domains of inquiry in computer science since it was first proposed in the 1950s. Defined as the part of computer science concerned with designing systems that exhibit the characteristics associated with human intelligence, the field has attracted researchers owing to its ambitious goals and enormous underlying intellectual challenges [National Research Council (NRC), 1999]. The ultimate aim of AI is to make computer programs that are capable of solving real-world problems and achieving goals as done by humans – the pursuit of so called 'strong AI'. This goal has caught the attention of the media, but by no means do all AI researchers view strong AI as worth investigating; an excessive optimism in the 1950s and 1960s concerning strong AI has given way to an appreciation of the extreme difficulty of the problem [Copeland, 2000]. To date, progress in this direction has been meagre. Because 50 years of failure eventually starts affecting funding, the AI field has diversified and experts have established themselves in other areas where they can be said to have had some success. These new areas are less concerned with the business of making computers "think", focusing instead on what can be referred to as 'weak AI' – the development of practical technology for modeling aspects of human behaviour [Goodwins, 2001]. In this way, AI research has produced an extensive body of principles, representations, and algorithms. Today, successful

AI applications range from custom-built expert systems to mass-produced software and consumer electronics.

The number of applications for weak AI is growing. AI-related patents in the US increased from 100 to 1700 between 1989 and 1999, with a total of 3900 patents mentioning AI related terms. AI systems are generally embedded within larger systems – applications can be found in video games, speech recognition, and in the commercial 'data mining' sector. Full speech recognition, leading to voice-led internet access or recognition in security applications, is anticipated relatively soon. However, the ability to extract meaning from the natural language recognition still remains way off. The data mining market uses software to extract general regularities from online data, dealing in particular with large volumes or patterns humans may not look for or incapable of their capture. Such systems could be used to predict consumer preferences or extract trends from market data such as patents and news articles. Sales already have reached US$3.5 billion and were anticipated to be US$8.8 billion in 2004. Weak AI is already behind systems that detect 'deviant' behaviour in credit card use, which has lead to improved credit card fraud detection. Potential applications of these techniques to state-security situations are likely to be controversial [Arnall, 2003].

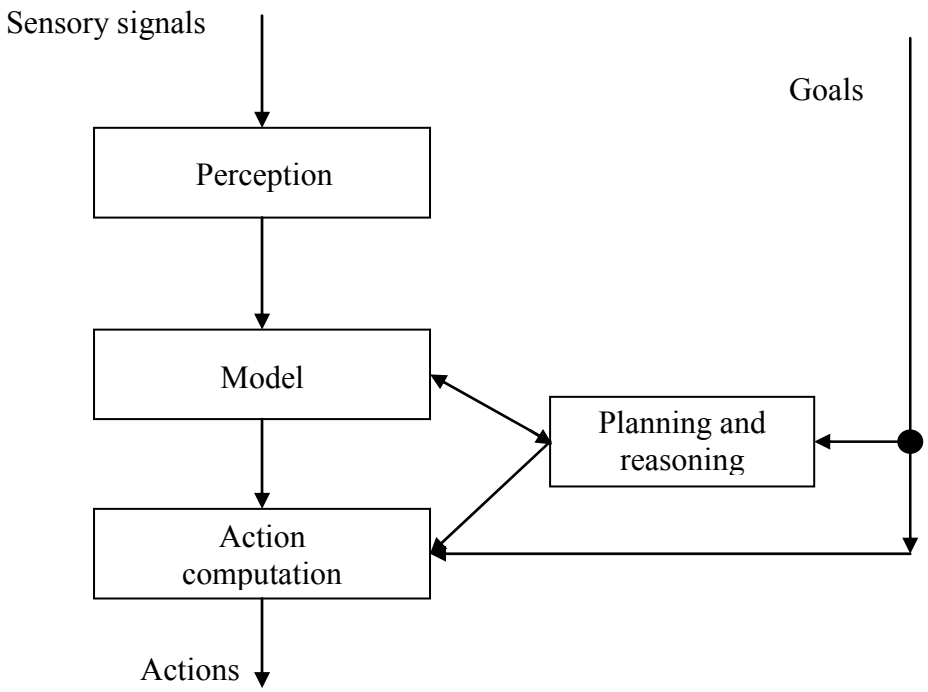

Figure 1.1: An AI system

The architecture of a typical AI agent is shown in Figure 1.1. This agent perceives and models its environment and computes appropriate actions perhaps by anticipating their effects. Changes made to any of the components shown in Figure 1.1 might count as "learning". Different learning mechanisms may be employed depending on which subsystem is being changed.

As a broad subfield of AI, *machine learning* (ML) is concerned with the development of algorithms and techniques, which allow computers to "learn". Learning, like intelligence, covers such a broad range of processes that it is difficult to define precisely. A dictionary definition includes phrases such as "to gain knowledge or understanding of or skill in, by study, instruction or experience and modification of a behavioural tendency by experience." Zoologists and psychologists study learning in animals and humans. In this thesis, we focus on learning in machines, specifically computing machines that learn relationships in a given data set using AI-based algorithm; the knowledge gained via learning is subsequently utilized to solve real-world problems such as process modelling, classification and optimization. There are several parallels between animal and machine learning. Certainly, many techniques in machine learning derive from the efforts of psychologists to make more precise their theories of animal and human learning through computational models. It seems likely also that the concepts and techniques being explored by researchers in machine learning may illuminate certain aspects of biological learning.

In ML, a machine learns whenever it changes its structure, program or data based on its inputs or in response to external information in such a manner that its expected future performance improves. Some of these changes, such as the addition of a record to a data base, fall comfortably within the province of other disciplines and are not necessarily better understood for being called learning. But, for instance, when the performance of a speech recognition machine improves after hearing several samples of a person's speech, we feel quite justified saying that the machine has "learned". Machine learning usually refers to the changes in systems that perform tasks associated with AI. Such tasks involve recognition, diagnosis, planning, process control, prediction, etc. The changes might be either enhancements to already performing systems or synthesis of new systems.

Table 1.1: Well-known artificial intelligence based algorithms and there applications [Tambe et al, 1996]

AI Areas	AI Formalism	Applications	References
Artificial neural networks	Perceptron	Speech and image processing, pattern recognition	Rosenblatt [1958, 1962]
	Multilayer perceptron	Function approximation, signal processing, signal filtering, data compression and reduction, time series modeling, noise reduction, pattern recognition and classification, image and speech processing, process control	Rumelhart et al. [1986a, b], Werbos [1974], Parkar [1985], Le Cun [1985]
	Radial basis function neural networks	Function approximation, process control, process modeling, pattern recognition and classification.	Moody and Darken [1989]
	Kohonen self-organizing map	Data compression, clustering, classification and mapping.	Kohonen [1988, 1989]
	Counterpropogation network	Function approximation, lookup table, statistical analysis, pattern recognition and classification.	Hecht-Nielsen [1987, 1988]
	SAMANN	Dimensionality reduction, data projection, classification	Mao and Jain [1995]
	Auto-associative neural network	Dimensionality reduction, data projection	Kramer, 1991, 1992; Leonard and Kramer, 1993; Kuespert and McAvoy, 1994

AI Areas	AI Formalism	Applications	References
Evolutionary methods	Genetic algorithms	Process optimization and modeling	Davis [1991], Goldburg[1989]
	Genetic programming	Function approximation, process modeling, pattern recognition	Koza[1992]
	Tabu search	Process optimization	Glover [1989, 1990]
Fuzzy logic	Nero-fuzzy networks	Modeling and pattern recognition, classification, rule extraction	Zhang [1997], Conde [2000]
	Fuzzy curves and surfaces	Input selection, data mining, dimensionality reduction	Lin et al., [1996, 1998]
Support vector machines	Support vector classification and regression	Classification, pattern recognition, function approximation, process modeling and control.	

1.2 VARIOUS AI FORMALISMS

In the quest to create intelligent machines, the field of Artificial Intelligence has split into several different approaches based on the most promising theories, methods and applications. These rivalling theories have lead researchers to one of two basic approaches; *bottom-up* and *top-down*. Bottom-up theorists believe the best way to achieve artificial intelligence is to build electronic replicas of the human brain's complex network of neurons, while the top-down approach attempts to mimic the brain's behaviour with computer programs. This thesis focuses on the top-down approach. Ever since the path-breaking publication of the error-back-propagation algorithm to train multi-layer perceptron neural networks [Rumelhart et al., 1989] the field of AI has witnessed an explosive growth in related theories and applications. There is hardly any scientific and technology field wherein AI has not found applications. Accordingly, this thesis is concerned with the applications of AI to chemical engineering and technology processes. The major AI paradigms and their applications are listed in Table 1.1. In what follows an overview of major AI formalisms is provided.

1.2.1 Artificial Neural Networks

Artificial neural networks (ANNs) is an information-processing paradigm founded on the mechanisms followed by the highly interconnected cellular structure of the human brain. The human brain is made up of a network of billions of cells called *neurons*, and understanding its complexities is seen as one of the last frontiers in scientific research. It is the aim of AI researchers who prefer the bottom-up approach to construct electronic circuits that act similar to neurons in the human brain. Although much of the working of the brain remains unknown, the complex network of neurons is what gives humans intelligent characteristics. By itself, a neuron is not intelligent, but when grouped together, neurons are able to pass electrical signals through networks. Research has shown that a signal received by a neuron travels through the dendrite region, and down the axon (see Figure 1.2). Separating nerve cells is a gap called the *synapse*. In order for the signal to be transferred to the next neuron, the signal must be converted from

electrical to chemical energy. The signal can then be received by the next neuron and processed.

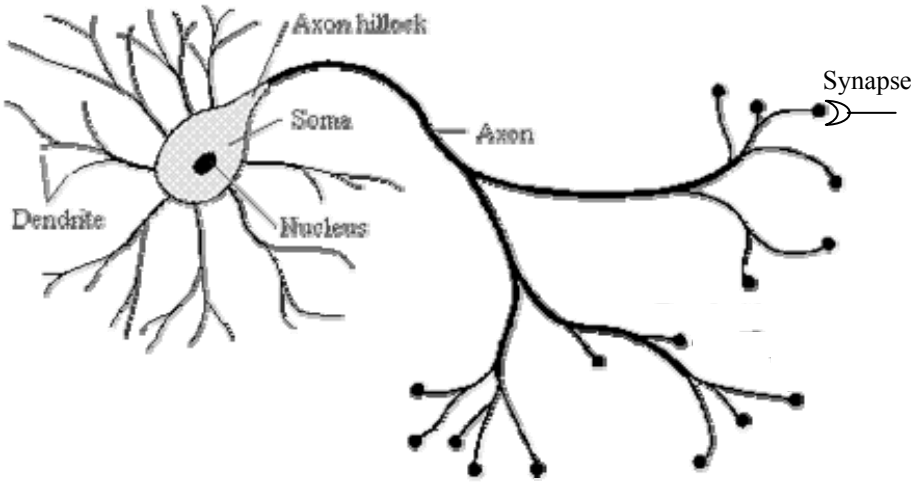

Figure 1.2: A biological neuron

In modern software implementations of ANNs the approach inspired by biology has more or less been abandoned for a more practical approach based on statistics and signal processing. In some of these systems neural networks or parts of neural networks (such as "artificial" neurons) are used as components in larger systems that combine both adaptive and non-adaptive elements. While the more general approach of such adaptive systems is more suitable for real-world problem solving, it has far less to do with the traditional artificial intelligence connectionist models. What they do however have in common is the principle of non-linear, distributed, parallel and local processing and adaptation. The various types of ANNs and their applications are tabulated in Table 1.1.

1.2.2 Evolutionary Algorithms

In artificial intelligence, an evolutionary algorithm (EA) is a subset of evolutionary computation, a generic population-based metaheuristic optimization algorithm. An EA uses some mechanisms inspired by the biological evolution namely *reproduction*, *mutation*, *recombination*, *natural selection* and *survival of the fittest*. Candidate solutions to the optimization problem play the role of individuals in a population, and the cost function determines the environment

within which the solutions "live". Evolution of the population then takes place after the repeated application of the above operators (See Figure 1.3). Evolutionary algorithms became widely recognized as search and optimization methods as a result of the work of Ingo Rechenberg in the 1960s and early 1970s - his group was able to solve complex engineering problems through evolution strategies (1971 PhD thesis and the resulting 1973 book) [Rechenberg, 1971]. Also highly influential was the work of John Holland in the early 1970s, and particularly his 1975 book [Holland, 1975].

EAs are often viewed as a global optimization method although convergence to a global optimum is only guaranteed in a weak probabilistic sense. However, one of the strengths of EAs is that they perform well on "noisy" functions where there may be multiple local optima. EAs tend not to get "stuck" in local minima and can often find globally optimal solutions. EAs are well suited for a wide range of combinatorial and continuous problems, though their variations are tailored towards specific domains. In the following, a brief overview of a few important evolutionary algorithms is provided.

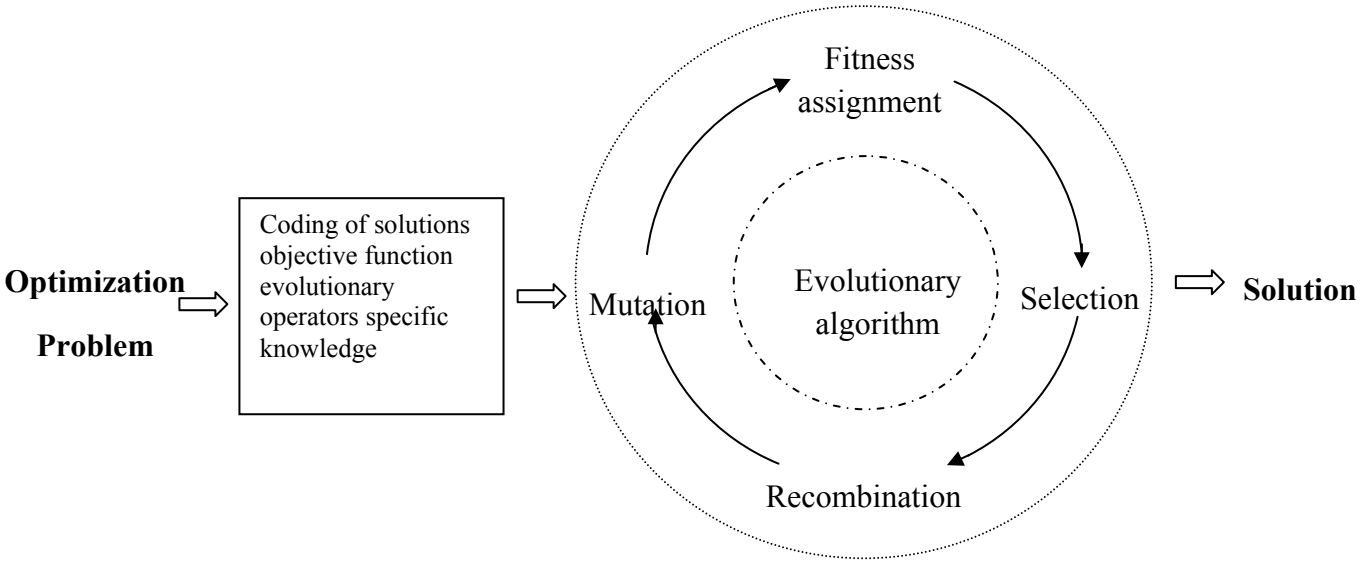

Figure 1.3: Schematic of Evolutionary Algorithms

A. Genetic algorithms

A genetic algorithm (GA) [Holland, 1975; Goldberg, 1989] is a stochastic search and optimization technique used in computing the true or approximate solutions to function maximization/minimization problems. The term "stochastic" indicates a random element in their implementation procedure. Genetic algorithms are categorized as global search heuristics. They are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called "*recombination*").

Genetic algorithms are implemented as a computer simulation in which a population of abstract representations (called *chromosomes* or the *genotype* or the *genome*) of candidate solutions (called *individuals*, *creatures*, or *phenotypes*) to an optimization problem are created. Thereafter, GA, while performing simplified genetic operations, evolves toward better solutions. Traditionally, solutions are represented in binary as strings of "0s" (zeros) and "1s" (ones), but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and proceeds over a number of generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are selected stochastically from the current population (based on their fitness), and modified (recombined and possibly mutated) to form a new population of candidate solutions. The new population is then acted upon similarly in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been evolved, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

B. Genetic programming

Genetic programming (GP) is an evolutionary algorithm based methodology inspired by the biological evolution to search and develop computer programs performing a user-defined task. Thus it is a technique used to optimize a population of computer programs according to a fitness landscape determined by the program's ability to perform a given computational task. The GP has roots in

the evolutionary algorithms first utilized by Nils Barricelli in 1954. Stephen Smith [1980] and Nichael Cramer [1985] reported the first results on the GP methodology. In 1981 Forsyth reported the evolution of small programs in forensic science for the UK police. John Koza is the principal proponent of the modern day GP and has pioneered the application of genetic programming in various complex optimization and search problems.

GP is a computationally intensive procedure and therefore in the 1990s it was mainly used to solve relatively simple problems. However, more recently, thanks to improvements in the GP technology and to the exponential growth in the CPU power, GP produced many novel and outstanding results in areas such as quantum computing, electronic design, game playing, sorting, searching and many more. These results include the replication or development of several post-year-2000 inventions. GP has also been applied to evolvable hardware as well as computer programs. There are several GP patents listed in the web site [http://www.genetic-programming.com/patents.html]. In recent years, the GP has been utilized in performing "symbolic regression". Given input-output data for a model, the GP can search and optimize the functional form and its parameters automatically to arrive at a best fitting linear/nonlinear data fitting function. In the present thesis, the said symbolic regression characteristic of the GP has been explored for process modeling.

C. Ant colony optimization

The ant colony optimization algorithm (ACO), introduced by Marco Dorigo [Dorigo et al., 1996, 1999], is a probabilistic technique for solving computational problems that can be reduced to searching good paths through graphs. The algorithm is inspired by the behaviour of ants in finding paths from the colony to food.

In the real world, ants (initially) wander randomly, and upon finding a food source return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep travelling at random, but instead to follow the trail, returning and reinforcing it if they eventually find food. Over time, however, the pheromone trail starts to evaporate, thus reducing its attractive

strength. The more time it takes for an ant to travel down the path and back again, the pheromones have more time to evaporate. A short path, by comparison, gets marched over faster, and thus the pheromone concentration remains high as it is laid on the path as fast as it can evaporate. Pheromone evaporation has also the advantage of avoiding the convergence to a locally optimal solution. If there was no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the exploration of the solution space would be constrained. Thus, when one ant finds a good (short, in other words) path from the colony to a food source, other ants are more likely to follow that path, and positive feedback eventually leaves all the ants following a single “short” path. The idea of the ant colony algorithm is to mimic this behaviour with "simulated ants" walking around the graph representing the optimization problem needing a solution.

Ant colony optimization algorithms have been used to produce near-optimal solutions to the *Travelling Salesman Problem* [Dorigo and Gambardella, 1997]. They have an advantage over simulated annealing and genetic algorithm approaches when the graph may change dynamically; the ant colony algorithm can be run continuously and adapt to changes in real time.

1.2.3 Tabu Search

Tabu search [Glover, 1989 and 1990] is a mathematical optimization method, belonging to the class of local search techniques. Tabu search enhances the performance of a local search method by using memory structures. Tabu search method generally attributed to Fred Glover [Glover, 1989, 1990] uses a local or neighbourhood search procedure to iteratively move from a solution x to a solution x' in the neighbourhood of x, until some stopping criterion has been satisfied. To explore regions of the search space that would be left unexplored by the local search procedure and—by doing this—escape local optimality, tabu search modifies the neighbourhood structure of each solution as the search progresses. The solutions admitted to $N^*(x)$, the new neighbourhood, are determined through the use of special memory structures. The search now progresses by iteratively moving from a solution x to a solution x' in $N^*(x)$.

Perhaps the most important type of short-term memory to determine the solutions in $N^*(x)$–also the one that gives its name to tabu search–is the use of a "tabu list". In its simplest form, a tabu list contains the solutions that have been visited in the recent past (less than n moves ago, where n is the tabu tenure). Solutions in the tabu list are excluded from $N^*(x)$. Selected attributes in solutions recently visited are labelled tabu-active. Solutions that contain tabu-active elements are taboo. This type of short-term memory is also called "recency-based memory".

Tabu lists containing attributes are much more effective, although they raise a new problem. When a single attribute is forbidden as tabu, typically more than one solution ends up being taboo. Some of these solutions that must now be avoided might be of excellent quality and might not have been visited. To overcome this problem, aspiration criteria are introduced which allow overriding the tabu state of a solution to include it in the allowed set. A commonly used aspiration criterion is to allow solutions that are better than the currently best known solution.

1.2.4 Expert Systems

An expert system, also known as a "knowledge based system", is a computer program that contains a part of the subject-specific knowledge, and knowledge and analytical skills of one or more domain experts in that subject. This class of program was first developed by researchers in artificial intelligence during the 1960s and 1970s and applied commercially throughout the 1980s. The most common form of expert systems is a program made up of a set of rules that analyze information (usually supplied by the user of the system) about a specific class of problems, as well as providing mathematical analysis of the problem(s), and, depending upon their design, recommend a course of user action in order to implement appropriate corrections. It is a system that utilizes what appear to be reasoning capabilities to reach conclusions.

Expert systems are most valuable to organizations that have a high-level of know-how experience and expertise that cannot be easily transferred to other members. They are designed to carry the intelligence and information found in the

intellect of domain experts and provide this knowledge to other members of the organization for problem-solving purposes.

Table 1.2: Examples of expert systems in chemical engineering and technology

No.	Expert systems	Application details
1	FALCON	Fault Analyzer Consultant for Chemical plant, Du Pont
2	CATDEX	Catalytic Cracking Unit Diagnostic Expert, Columbia University
3	BIOEXPERT	Fault diagnosis extert system for waste-water treatment, Lepoint et al., 1989
4	EXACT	Expert for Adaptive PID Controller Tuning, Foxboro
5	ASPEN PLUS, DESIGN II, PRO II	Computer aided process design
6	EXSEP	Multi-component separation design
7	HENS, HEATEX	Heat Exchanger Network Synthesis
8	CONPHYDE	Consultant for physical property decision, Carnegie-Mellon
9	BATCHKIT	Batch process planning
10	MIN-CYANIDE	System for minimizing cyanide wastes in electroplating plants
11	CAPS	System for plastics selection for the final product
12	PASS	Pump Application Selection System
13	DECADE	Design Expert for Catalyst Development, Carnegie Mellon Univ.

Typically, the problems to be solved by an expert system are of the type that would normally be tackled by a professional. Real experts in the problem domain (which will typically be very narrow, for instance, "selection of a pump") are asked to provide "rules of thumb" on how they evaluate the problems, either explicitly with the aid of experienced systems developers, or sometimes

implicitly, by getting such experts to evaluate test cases and using computer programs to examine the test data. Generally, expert systems are used for problems for which there is no single "correct" solution, which can be encoded, in a conventional algorithm — one would not write an expert system to find shortest paths through graphs, or sort data, as there are simply easier ways to do these tasks. A list of notable expert systems in the area of process systems engineering and control is given in Table 1.2.

1.2.5 Fuzzy Logic

Fuzzy logic [Zadeh, 1965] is derived from the fuzzy set theory dealing with the reasoning that is approximate rather than precisely deduced from the classical predicate logic. It can be thought of as the application side of the fuzzy set theory with well-thought out real world expert values for a complex problem [Klir, 1997].

Degrees of truth are often confused with probabilities. However, they are conceptually distinct; fuzzy truth represents a membership in vaguely defined sets, not likelihood of some event or condition. To illustrate the difference, consider this scenario: Bob is in a house with two adjacent rooms, the kitchen and the dining room. In many cases, Bob's status within the set of things "in the kitchen" is completely plain; he is either "in the kitchen" or "not in the kitchen". What about when Bob stands in the doorway? He may be considered "partially in the kitchen". Quantifying this partial state yields “fuzzy set membership”. With only his big toe in the dining room, we might say Bob is 99% "in the kitchen" and 1% "in the dining room", for instance. No event (like a coin toss) will resolve Bob to being completely "in the kitchen" or "not in the kitchen", as long as he is standing in that doorway. Fuzzy sets are based on vague definitions of sets, not randomness.

Fuzzy logic allows for set membership values to range (inclusively) between 0 and 1, and in its linguistic form represents, imprecise concepts like "slightly", "quite" and "very". Specifically, it allows partial membership in a set. A typical fuzzy logic system is shown in Figure 1.4.

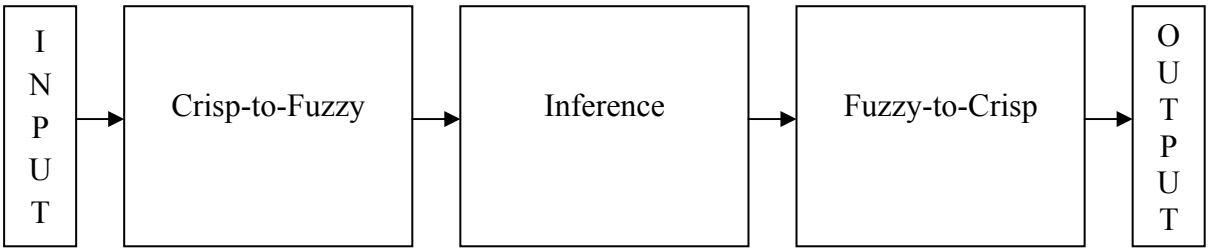

Figure 1.4: Schematic of fuzzy logic systems

1.3 THESIS OUTLINE

The aim of this thesis is design and develop applications of AI and ML based formalisms for chemical and biochemical engineering/technology systems. The second chapter of this thesis contains a detailed overview along with the corresponding literature survey of the major AI and ML based formalisms used in the thesis. Depending upon their applications, the formalisms are classified into four categories namely *modeling*, *classification*, *optimization* and *data reduction/projection*. The third chapter details various AI and ML based process modeling studies. Chapter 4 describes applications of the clustering/classification formalisms and chapter 5 deals with various process optimization studies. Chapter 6 reports case studies demonstrating the effectiveness of the AI-based novel algorithms for low-dimensional projection, feature extraction, dimensionality reduction and input selection of process data. Finally, chapter 7 provides the concluding remarks on the work presented in the thesis.

1.4 REFERENCES

1. Arnall, H. (2003), Future Technologies, Today's Choices, *A report for the Greenpeace Environmental Trust*, Canonbury Villas, London N1 2PN, ISBN 1-903907-05-5. www.greenpeace.org.uk.
2. Barricelli, Nils Aall (1954), Esempi numerici di processi di evoluzione, *Methodos*, 45–68.
3. Conde, G.A.; Ramos, P.G.; Vasconcelos, G.C. (2000), Neuro-fuzzy networks for pattern classification and rule extraction Neural Networks. Proceedings. *Sixth Brazilian Symposium*, Page(s):289.
4. Copeland, B.J. (2000). What is Artificial Intelligence? AlanTuring.net; Reference Articles: May 2000 [online].
5. Cun, Le, Y. (1985), Une Procedure d'Atprentissage pour reseau a ceuil assymetrique. Incognitiva 85: A la frontiere de l'Intlligence artificielle des sciences de la connaissance des neuron sciences, CESTA, Parris, 599–604.
6. Davis L. (1991), *Handbook of GAs*, New York: Van Nostrand Reinhold.
7. Dorigo, M. and L. M. Gambardella (1997), Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1 (1), 53–66.
8. Dorigo, M., G. Di Caro and L. M. Gambardella (1999), Ant Algorithms for Discrete Optimization. *Artificial Life*, 5 (2), 137–172.
9. Dorigo, M., V. Maniezzo and A. Colorni (1996), Ant System: Optimization by a Colony of Cooperating Agents, *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, 26 (1), 29–41.
10. Glover, F. (1989), Tabu Search-Part I. *ORSA J. Comput.* 1, 190–206.
11. Glover, F. (1990), Tabu Search-Part II. *ORSA J. Comput.* **2**, 4–32.
12. Goldberg, D. E. (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA.

13. Goodwins, R. (2001). *The Machine that Wanted to be a Mind*. ZDNet UK; News: 23 Jan 2001 [online]. http://news.zdnet.co.uk/story/0,,s2083911,00.html
14. Hecht-Nielsen, R. (1988), Applications of counterpropagation networks. *Neural Networks*, 1, 131–139.
15. Hecht-Nielsen, R. (1987). Counterpropagation networks. *Appl. Optics.*, 26, 4979–4984.
16. Holland, John H (1975), *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor. http://www.alanturing.net/turing_archive/pages/Reference%20Articles/what_is_AI/What%20is%20AI02.html.
17. Klir G., UTE H. St. Clair and Bo Yuan (1997), *Fuzzy Set Theory Foundations and Applications*.
18. Kohonen, T. (1982), Self-Organized formation of topologically correct feature maps. *Biol. Cybern.* 43, 59–69.
19. Kosko, B. (1992), Fuzzy cognitive maps. *Int. J. Man-Machine Stud.*, 24, 1986, 65–75.
20. Koza, J., *Genetic programming*, The MIT Press Cambridge, MA.
21. Moody, T. and C. Darken (1989). Fast learning in networks of locally tuned processing units. *Neural computation*, 1, 281–294.
22. National Research Council (1999). Funding a Revolution: Government Support for Computing Research. Washington, DC, USA: National Academy Press. http://www.nap.edu/readingroom/books/far/notice.html.
23. Parkar, D. (1985), Learning logic, *Technical report* TR-47, center for computational research in economics and management science, MIT, Cambridge, MA.
24. Rechenberg, I. (1971): *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution* (PhD thesis). Reprinted by Fromman-Holzboog (1973).
25. Rosenbaltt, F. (1962), *Principles of neorodynamics* . Spartan, New York.
26. Rosenbaltt, F. (1958), The perceptron: A probabilistic model for information storage and organization in the brain. *Psycho. Rev.*, 65, 386–408.

27. Rumehart, D., G. Hinton, and R. Williams (1986b), Learning internal representations by error back propagation. In *Parallel distributed processing: Explorations in microstructure of cognition*, vol.1: Foundations, MIT Press, Cambridge, MA.
28. Rumehart, D., G. Hinton, and R. Williams (1986a), Learning representations by back propagating errors. *Nature*, 323, 533–536.
29. Tambe, S. S., B. D. Kulkarni and P. B. Deshpande (1996), *Elements of Artificial Neural Networks with selected applications in Chemical Engineering, and Chemical and Biological Sciences*, Simulations & Advanced Controls, Louisville, KY.
30. Werbos, P. (1974), *Beyond regression: New tool prediction and analysis in the behavioral sciences*. Ph. D. Thesis, Harvard University.
31. Zadeh, L. A. (1965), Fuzzy Sets, *Information and Control*, Vol. 8, 338–353.
32. Zhang; Jie, Morris, J. (1997), Neuro-fuzzy networks for process modelling and model-based control Neural and Fuzzy Systems: Design, Hardware and Applications (Digest No: 1997/133), *IEE Colloquium*, 9, 6/1–6/4.

CHAPTER 2

OVERVIEW OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING FORMALISMS

2.1 BACKGROUND

The past decade has created a challenging environment for process manufacturing. Some of these challenges include rapid technological innovations, stiff competition, complex process designs, huge capital investment, stringent product quality specifications, large product volumes, tight delivery schedules and stiff environmental regulations. To overcome these challenges, in recent years, process modeling, monitoring, control, fault detection and diagnosis, simulation and optimization are being increasingly and extensively employed in chemical engineering and technology. Accordingly, researchers in these fields are constantly striving to develop new and efficient methodologies to perform the stated tasks. The most striking outcome of these efforts is utilization of artificial intelligence and machine learning formalisms as an alternative to the conventional phenomenological and empirical modeling and optimization strategies.

Artificial intelligence (AI) is a branch of computational science, which develops algorithms mimicking various kinds of intelligent behavior exhibited by biologically evolving species, to provide novel and efficient solutions to complex modeling, classification and optimization problems. As a broad subfield of AI, *machine learning* (ML) is concerned with the development of algorithms and techniques, which allow computers to "learn" relationships in a given data set. Unlike AI, the ML algorithms are not based on the intelligent behaviour observed in nature but they are based on rigorous mathematical and statistical foundations. Both AI and ML-based modeling and classification formalisms are exclusively data-driven and aim at learning (capturing) linear/nonlinear correlations and trends/patterns in available data sets.

The AI and ML formalisms possess a number of attractive properties *vis a vis* conventional modeling and optimization strategies and therefore this thesis aims at developing AI and ML-based applications for a variety of process engineering tasks such as steady-state and dynamic modeling, soft-sensor development, fault detection and diagnosis, data reduction/projection, process monitoring, clustering/classification and optimization. The specific AI and ML formalisms developed, utilized and further improvised in the thesis are artificial

neural networks, genetic programming, fuzzy logic, genetic algorithms, tabu search, memetic algorithms, Sammon's mapping based neural network, auto-associative neural networks, self organizing maps, etc.

2.2 MODELING FORMALISMS

In a number of real world chemical processes, phenomenological models are either unavailable or difficult to construct. This happens since the physico-chemical phenomena underlying a chemical process is usually not fully understood. Also, obtaining the kinetic, thermodynamic and heat and mass transfer information that is needed to construct a phenomenological process model is a cumbersome, time consuming and expensive task. Notwithstanding these difficulties, models are necessary for predicting the process performance under varying operating conditions and also in improving the process efficiency. In the absence of phenomenological models, real-life process input-output data can be used to construct an empirical or a "black box" process model. These models, similar to the phenomenological ones may then be used for predicting the process performance and a variety of other tasks such as control, fault detection and diagnosis and process optimization. A significant advantage of the empirical or black box models vis-à-vis the phenomenological ones is that these can be developed relatively easily. Conventionally, linear or nonlinear regression methods are used in developing empirical models. A major drawback of these methods is that the mathematical structure (form) of the data-fitting model must be specified *a priori* before an estimation of model parameters could be attempted. This is a significant difficulty since most of the chemical processes exhibit nonlinear behaviour and therefore it is not known *a priori* which nonlinear model is appropriate for fitting the process data. Furthermore, for processes with multiple operating variables (inputs) and output variables (such as conversion, yield, etc.), it is necessary to select multiple nonlinear functions for fitting the process data which becomes even more tedious and in most cases an impossible task. How AI and ML based formalisms overcome these difficulties is described in the following sections.

2.2.1 Artificial Neural Networks

Artificial neural networks (ANNs) are an artificial intelligence based information-processing paradigm founded on the mechanisms followed by the highly interconnected neuronal structure of the human brain. The ANNs mimic some of the observed properties, such as the pattern recognition (classification) possessed by the human brain. These mathematical models also exhibit a similarity with the "learning-by-experience" principle followed by the biological species. The ANNs are based on the concept that a highly interconnected system of simple processing elements (also called "nodes") can learn (model) complex nonlinear interrelationships existing between independent and dependant variables of a data set [Freeman et al., 1991; Bishop, 1994; Bulsari, 1995; Tambe et al., 1996; Nandi et al., 2002]. Neural networks have attracted much interest lately for their use as predictive models as well as for pattern recognition. They have been used successfully to model dynamic and nonlinear systems such as deterministic chaos [Lapedes and Farber, 1987; Ydstie, 1990; Levin, 1990], chaotic chemical systems [Admoaitis et al., 1989] and other chemical reactions [Bhat et al., 1990]. Neural networks have been used for system identification and control by a number of researchers [Donat et al., 1990; Hemandez and Arkun, 1990; Narendra and Parthasarathy, 1990; Psichogios and Ungar, 1991; Haesloop and Holt, 1990; Willis et al., 1991] as also for process fault diagnosis by Hoskins and Himmelblau [1988], Watanabe et al. [1989], Venkatasubramanian et al. [1990] among others. ANNs owing to there architecture are well-suited for parallel competition and thus they allow a speedier solution to a large-dimensional modeling and prediction problem. They also have a powerful representational capability. Cybenko [1989] has shown that given enough nodes in the hidden layer, the multi-layer perceptron network with a single hidden layer is sufficient to approximate any function. The radial basis function network (RBFN) has also been shown to have a similar capability to represent arbitrary functions [Park and Sandberg, 1991].

The function approximated by an ANN is defined by many factors, for example by the number and arrangement of neurons, their interconnections, etc. For developing a nonlinear model, a feed-forward ANN architecture namely "multilayer perceptron (MLP)" is most commonly used; the MLP is also known as "Back-propagation" neural network. The MLP network approximates nonlinear

input-output relationships as defined by, $\mathbf{y}_n = f(\mathbf{x}_n, \mathbf{w}), n = 1, 2, ..., N$, where **x** is an *I*-dimensional vector of inputs, **y** refers to an *L*-dimensional output vector and **w** is the vector defining network weights. The MLP network usually consists of three layers. The layers described as *input*, *hidden,* and *output* layers comprise *I*, *J*, and *L* number of processing nodes, respectively. There may be more than one hidden layers in the ANN architecture, such as, Figure 2.1 depicting the MLP architecture with two hidden layers. Each node in the input (hidden) layer is linked to all the nodes in the hidden (output) layer using weighted connections. The MLP architecture also houses a bias node (with a fixed input, e.g., ±1) in its input and hidden layers; the bias nodes are also connected to all the nodes in the next layer. Usage of bias nodes helps the MLP-approximated function to be positioned anywhere in the *M*-dimensional input space; in their absence, the function is forced to pass through the origin of the *I*-dimensional space. The *I* number of nodes in the input layer is equal to the number of independent variables, whereas the number of output nodes (*L*) equals the number of process outputs. However, the number of hidden nodes is an adjustable parameter whose magnitude is determined by issues such as the desired approximation and generalization performance of the network model. In order that the MLP network accurately approximates the nonlinear relationship existing between its inputs and outputs, it needs to be trained in a manner such that a pre-specified error function is minimized. In essence, the MLP training procedure aims at obtaining an optimal weight set **w** that minimizes a pre-specified error function. The commonly employed error function is the *root-mean-squared error* (RMSE) defined as:

$$RMSE = \sqrt{\frac{\sum_{n=1}^{N}\sum_{l=1}^{L}\left(y_{n,l} - \hat{y}_{n,l}\right)}{N \times L}} \tag{2.1}$$

where *N* refers to the number of input-output data pairs available for training, $y_{n,l}$ and $\hat{y}_{n,l}$ are the desired (target) and MLP predicted values of the *l*th output node, respectively. The widely used formalism for the RMSE minimization is the *error-back-propagation* (EBP) algorithm [Rumelhart et al., 1986] utilizing a gradient-descent technique known as the *generalized delta rule* (GDR) for iterative updation of weights as given by.

$$\mathbf{w}(t+1) = w(t) - \eta \frac{\partial E}{\partial \mathbf{w}} \tag{2.2}$$

where E refers to an error measure. This gradient-decent technique poses two major limitations on the use of activation function for the hidden and output layer neurons:

1. First derivative of the activation function must be feasible.
2. Activation function must be a bounded one.

The functions satisfying these limitations are:

1. Linear: $y = mx + c$ (2.3)
2. Step: $y = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$ (2.4)
3. Logistic sigmoid: $y = \frac{1}{1 + \mathrm{e}^{-x}}$ (2.5)
4. Hyperbolic tangent: $y = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$ (2.6)
5. Radial basis: $y = \frac{e^{-x}}{2\sigma}$ (2.7)

The details of the heuristic procedure involved in obtaining an optimal network model possessing good prediction and generalization capabilities can be found in e.g., Freeman and Skapura [1991], Bishop [1994], Tambe et al. [1996] and Nandi et al. [2002]. The EBP training algorithm makes use of two adjustable (free) parameters namely, the learning rate, η ($0 < \eta \leq 1$) and the momentum coefficient, α ($0 < \alpha \leq 1$). The magnitudes of both these parameters need to be optimized heuristically.

The advantages of an MLP-based model are:

(i) They are constructed exclusively from the data (example set) comprising independent (causal) and dependant (response) variables of a system.

(ii) The detailed knowledge of the fundamental mechanisms underlying the system behaviour is unnecessary for the model development.

(iii) An MLP model with properly fitted weight coefficients possesses the much desired "generalization" ability owing to which it can accurately predict outputs for a new set of inputs.

(iv) Even multiple input-multiple output (MIMO) nonlinear relationships can be fitted simultaneously.

(v) Since an MLP network uses a generic nonlinear function for fitting data, it is not necessary to pre-specify the form of the data-fitting function explicitly; this feature is greatly advantageous in modeling nonlinear systems where guessing an appropriate form of the nonlinear data fitting function is a cumbersome, difficult and time-consuming task.

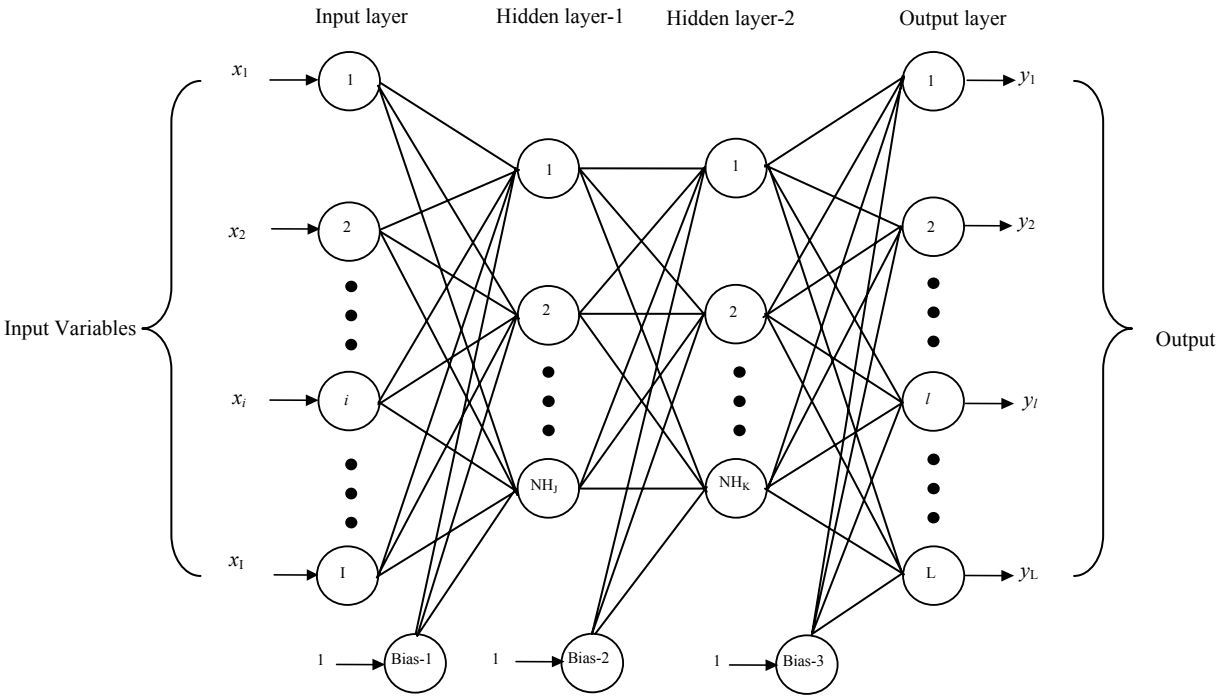

Figure 2.1: Architecture of Multilayer Perceptron Network Model

The MLP-based models also possess a few drawbacks:

(i) Fitting of weight coefficients of the network model is usually an ill-posed problem.

(ii) Adjusting the weight coefficients of the model iteratively is a time consuming process.

(iii) MLP networks are known as 'black-box' models since the model coefficients can not be explained (interpreted) in terms of the data used for fitting the model.

(iv) To build an MLP model with good prediction accuracy and generalization capability, it is necessary that the data should be statistically well-distributed and preferably large-sized as also free of noise and errors.

To be useful, an MLP model must not only possess a good output prediction accuracy but also good generalization ability so that the model captures the underlying trends existing in the example input-output data. The phenomenon, which adversely affects the MLP model's generalization performance is known as "overfitting". It occurs when the network model, in an attempt to increase its prediction accuracy, also learns the noise in the example data (known as "over-training") and/or when the model architecture houses more hidden nodes than necessary (known as "over-parameterization"). An over-trained and/or over-parameterized MLP model makes poor predictions for a new set of inputs. To prevent an occurrence of over-fitting, it is necessary to monitor the generalization performance of the MLP model continuously while it undergoes training (e.g., at the end of each training iteration). The step-wise procedure for avoiding an occurrence of over-fitting and thereby obtaining an optimal MLP model architecture and weight coefficients is given below [Nandi et al., 2004]. This procedure though meant for training a single hidden-layer MLP network, can be easily extended to two-hidden layer MLP networks.

Step 1. Partition the available set of example data comprising model inputs and outputs into two sets, namely *training* and *validation* sets; the ratio for this partitioning could be 4:1 or 3:2. Assume a small number of nodes (e.g., one or two) in the network has hidden layer and initialize the network weights randomly. Select the values of the EBP algorithm parameters, namely, learning rate η ($0 < \eta < 1.0$) and momentum coefficient, μ ($0 < \mu < 1.0$); fix the maximum number of iterations (t_{max}) over which the model is trained.

Step 2. Adjust the network weights iteratively using the EBP algorithm and training set data over t_{max} number of iterations. The weights resulting in

the least RMSE for the validation set (E_{val}) are considered to be optimal for the chosen number of hidden nodes and the particular set of randomized weights used for the network initialization. Such weights could be obtained in any one of the t_{max} number of training iterations.

Step 3. Repeat step 2, a number of times using a different random number sequence each time for initializing the network weights. This is performed for exploring the weight-space rigorously and locating the deepest minimum on the network's error surface. Record the weights leading to the smallest E_{val}.

Step 4. Repeat steps (2) and (3) by varying the number of hidden nodes systematically till E_{val} attains its smallest possible magnitude; the EBP parameters η and μ can also be optimized in a manner similar to the number of hidden nodes.

There are a number of algorithms for the training of MLP neural networks, for example,

- Error-back-propagation (EBP) [Rumelhart et al., 1986]
- Conjugate Gradient [Reifman et al., 1994]
- Genetic Algorithms [Holland 1975; Goldberg, 1989]
- Quickprop [Fahlman, 1988]
- Resilient Back–propagation [Riedmiller et al., 1993]
- Levenberg – Marquardt's algorithm [Levenberg, 1944; Marquardt, 1963]
- Bayesien Learning [Neal, 1996]

A few of these algorithms, which are relevant to the work presented in this thesis, are explained below.

A. Error-back propagation

The *Error Back Propagation* (EBP) [Rumelhart et al., 1986] is the most widely used algorithm for supervised training of a multilayer perceptron network. Owing to its extensive use in the training of MLP networks, the network itself is often referred to as an EBP or BP network. The EBP algorithm employees a special kind of error-correction strategy, which can be viewed as a generalization

of the "least-mean-squared (LMS)" error minimization technique. The LMS learning rule as proposed by Widrow and Hoff [1960] is targeted at single linearly processing unit, whereas the EBP algorithm trains the weights associated with the feed forward network comprising elements that perform nonlinear processing.

An MLP network can be viewed as a set of algebraic equations arranged in an hierarchical order to form an input-output mapping. Changing the structure of the MLP is akin to changing the hierarchical order of algebraic equations and network training is another way of estimating the parameters of the complex input-output transformation carried out by network's activation and transfer functions. MLP network's training begins by applying the *I*-dimensional input vector $\mathbf{x}_n$, to the input layer having *I* number of nodes.

Since the input layer nodes just serve as distribution points and perform no information processing, their input becomes input to the hidden layer nodes. When an input vector $\mathbf{x}_n$ is applied to the input layer, each hidden layer neuron computes the activation according to the weighted sum of its input as given by

$$\alpha_j = \sum_{i=1}^{I} \mathbf{w}_{ij}.\mathbf{x}_i + \theta_j \tag{2.8}$$

Where, α_j represents the activation of first hidden layer neurons. The vector W_{ij} denotes the weights of the connection between input layer nodes and j^{th} hidden node and θj refers to the strength of the connection that the bias neuron makes with j^{th} hidden node.

The output of j^{th} hidden unit is,

$$\mathbf{x}_j = f_1(\alpha_j) \tag{2.9}$$

The output of k^{th} hidden unit is,

$$\mathbf{x}_k = f_2(\alpha_k) \tag{2.10}$$

The output of l^{th} output layer unit is,

$$\mathbf{y}_l = f_3(\alpha_l) \tag{2.11}$$

where, f_m is known as the activation function.

An approach used for the adjustment of weights in the EBP algorithm is known as "generalized delta rule (GDR)". The GDR approach for weight adaptation is based on the principle of minimizing an error function. Starting from an arbitrary point in the weight space the GDR adapts the network weights in the stepwise manner so that the error function, which measures the learning performance of an MLP network, is minimized. A commonly assumed error function is the "sum–of–squares" (see Eq. 2.12) of the individual errors over all the output layer units, and over the entire pattern in the training set. It is referred to as the cumulative sum-squared- error (SSE) by symbol *E*.

$$E = \frac{1}{2}\sum_{n=1}^{n}\sum_{l=1}^{L}\left(y_{nl} - \hat{y}_{nl}\right)^2 \tag{2.12}$$

The weight adaption rules for various layers are given below.

1. Weight updation for output layer nodes

$$w_{kl}(t+1) = w_{kl}(t) + \eta.\delta_l.y_l \tag{2.13}$$

2. Weight updation in the first and second hidden layer nodes

$$w_{jk}(t+1) = w_{jk}(t) + \eta\delta_k x_k \tag{2.14}$$

3. Weight updation for nodes in between 1st hidden and input layer

$$w_{ij}(t+1) = w_{ij}(t) + \eta\delta_j x_i \tag{2.15}$$

4. For Output Layer bias node

$$\theta_l(t+1) = \theta_l(t) + \eta\delta_l \tag{2.16}$$

5. For 2nd hidden layer bias node

$$\theta_{\mathrm{k}}(t+1) = \theta_{\mathrm{k}}(t) + \eta\delta_k \tag{2.17}$$

6. For 1st hidden layer bias node

$$\theta_{\mathrm{j}}(t+1) = \theta_{\mathrm{j}}(t) + \eta\delta_j \tag{2.18}$$

The weight adaptation can be carried out either after each pattern-application (*pattern* mode), or after all training patterns have been applied once to the network (*batch* mode). In the pattern mode, the error with respect to an individual input pattern, E_k, is minimized, while in the batch mode, the cumulative error, E, representing the sum of the pattern-wise errors, is minimized. In both the weight adaptation modes, the network training continues until the network outputs satisfy a certain pre-selected convergence criterion.

B. Resilient-back propagation

Gradient descent techniques are the most widely used class of algorithms for supervised learning in neural networks. Adaptive gradient based algorithms with variable step-sizes try to overcome the inherent difficulty of the choice of the right learning rates. This is done by controlling the weight update for every single connection during the learning process in order to minimize oscillations and to maximize the update step-size. The best of these techniques known in terms of convergence speed, accuracy and robustness with respect to its parameters is the "resilient backpropagation (Rprop)" algorithm [Riedmiller et al., 1993] also refer to Schimann et al., [1993]; Riedmiller,[1994]; Joost et al., [1998] for comparisons of Resilient-Back Propagation with other supervised learning techniques.

Resilient-Back Propagation (Rprop) performs a direct adaptation of the weight step based on local gradient information. In crucial difference to the previously developed adaptation techniques, the efforts of adaptation are not blurred by the gradient behaviour. Individual update-value, Δ_{ij}, which solely determines the size of the weight-update, is introduced. This adaptive update-value evolves during the learning process based on its local sight on the error

function E (see Eq. 2.12) according to the following learning rule [Riedmiller et al., 1993].

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^{+} * \Delta_{ij}^{(t-1)}, & \text{if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} * \frac{\partial E^{(t)}}{\partial w_{ij}} > 0 \\ \eta^{-} * \Delta_{ij}^{(t-1)}, & \text{if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} * \frac{\partial E^{(t)}}{\partial w_{ij}} < 0 \\ \Delta_{ij}^{(t-1)}, & \text{else} \end{cases} \tag{2.19}$$

where $0 < \eta^{-} < 1 < \eta^{+}$

Verbalized, the adaption-rule works as follows: Every time the partial derivative of the corresponding weight w_{ij} changes its sign, which indicates that the last update was too big and the algorithm has jumped over a local minimum, the update-value, Δ_{ij}, is decreased by the factor η^{-}. If the derivative retains its sign, the update-value is slightly increased in order to accelerate convergence in a shallow region.

Once the update-value for each weight is adapted, the weight-update itself follows a very simple rule: if the derivative is positive (increasing error), the weight is decreased by its update-value, if the derivative is negative, the update-value is added:

$$\Delta w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)}, & \text{if } \frac{\partial E^{(t)}}{\partial w_{ij}} > 0 \\ +\Delta_{ij}^{(t)}, & \text{if } \frac{\partial E^{(t)}}{\partial w_{ij}} < 0 \\ 0, & \text{else} \end{cases} \tag{2.20}$$

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)} \tag{2.21}$$

However, there is one exception: if the partial derivative changes sign i.e. the previous step was too large and the minimum was missed, the previous weight update is reverted:

$$\Delta w_{ij}^{(t)} = -\Delta w_{ij}^{(t-1)}, \text{ if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} * \frac{\partial E^{(t)}}{\partial w_{ij}} < 0 \tag{2.22}$$

Due to this 'backtracking' weight-step, the derivative is supposed to change its sign once again in the following step. In order to avoid a double punishment of the update-value, there should be no adoption of the update-value in the succeeding step. In practice, this can be done by setting $\frac{\partial E^{(t-1)}}{\partial w_{ij}} = 0$ in the Δ_{ij} adaption-rule given above. The update-value and the weights are changed every time the whole pattern set has been presented once to the network (learning by epoch).

At the beginning, all update-values, Δ_{ij}, are set to initial value, Δ_0. Since Δ_0 directly determines the size of the first weight-step. It is preferably chosen in a reasonable proportion to the size of the initial weights. A good choice may be $\Delta_0 = 0.1$. However, the choice of this parameter is not at all critical. Even for much larger or much smaller values of Δ_0 fast convergence is reached.

The choice of decrease factor η^- and increase factor η^+ can be laid by following considerations: if a jump over a minimum occurred, the previous update-value was too large, for it is not known from the gradient information how far the minimum was missed; in average it will be a good guess to halve the update-value, i.e., $\eta^- = 0.5$. The increase factor η^+ has to be large enough to allow fast growth of the update-value in shallow regions of the error function. On the other hand the learning process can be considerably disturbed, if a too large increase factor leads to persistent changes of the direction of the weight step. The choice of $\eta^+ = 1.2$ is commonly used for good results independent of examined problem [Riedmiller, 1993]. A slight variation of this value does neither improve nor deteriorate convergence time.

One of the main advantages of Rprop lies in the fact that for many algorithms, problems of choice of parameter is not needed at all to obtain an optimal or at least nearly optimal convergence times. The main reason for the success of the Rprop algorithm roots in the concept of 'direct adoption' of the size of the weight-update. In contrast to all other algorithms, only the sign of the partial derivatives is used to perform both learning and adaptation. This leads to a transparent yet powerful adaptation process, that can be straight forward and very efficiently computed with respect to both time and storage consumption. Another aspect of common gradient descend is that the size of the derivative decreases

exponentially with the distance between the weight and the output-layer due to the limiting influence of the slope of the sigmoid activation function. Consequently, weights far away from the output-layer are less modified and do learn much slower. Using Rprop, the size of the weight-step is only dependent on the sequence of the signs, not on the magnitude of the derivative. For that reason, learning is spread equally all over the entire network; weights near the input layer have the equal chance to grow and learn as weights near the output layer.

C. Generalized regression neural network

The generalized regression neural network (GRNN) was introduced by Nadaraya [1964] and Watson [1964] and rediscovered by Specht [1991] to perform general (linear or nonlinear) regressions. The GRNN has been applied to solve a variety of problems such as prediction, control, plant modeling or general mapping problems [Rutkowski et al., 2003]. GRNNs are memory based feedforward networks that were introduced as a generalization of both the radial basis function networks (RBFNs) and probabilistic neural networks (PNNs) [Specht, 1991]. With increasing number of training samples, the GRNN asymptotically converges to the optimal regression surface. In addition to having a sound statistical basis, the GRNNs possess a special property in that the networks do not require iterative training. In Figure 2.2, GRNNs multiinput–multioutput (MIMO) architecture comprising four layers, namely, the *input*, *hidden*, *summation* and *output* layers is depicted. Unlike the most popular error-back-propagation algorithm [Rumelhart et al., 1986] that trains multilayer feedforward networks iteratively, the GRNN training is a single pass procedure. Also, GRNNs formulation comprises only one free parameter that can be optimized easily.

The principal advantages of the GRNN-based models [Kulkarni, et al., 2004] are that these models can efficiently and simultaneously approximate nonlinear multiinput–multioutput (MIMO) relationships and models can be developed in a significantly shorter time in comparison with the MLP or RBFN-based process models since the training of the model which is a one step procedure involves fixing a value of only a single free parameter. In what follows, the mathematical formulation of a GRNN described.

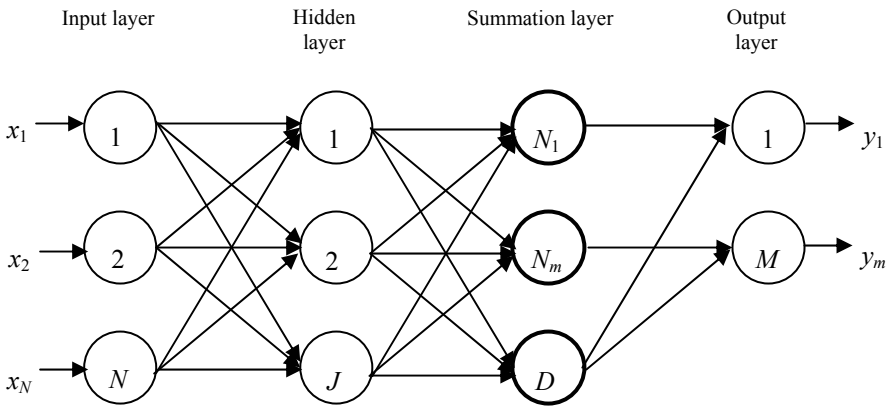

Figure 2.2: The schematic of multi-input multi-output GRNN

Consider a N-dimensional vector, $\boldsymbol{x} = [x_1, x_2,..., x_N]^T$, describing process input variables and the corresponding scalar output, y, representing the dependent (output) variable. GRNN performs regression by computing the conditional expectation of y given $\boldsymbol{x}$. Specifically, the GRNN estimates the joint probability density function (PDF) of $\boldsymbol{x}$ and y, i.e. $f(\boldsymbol{x}, y)$, to create a probabilistic model for predicting y. The PDF estimator model is constructed from the training input–output data set $\{x_i, y_i\}$; $i = 1, 2,..., I$, via nonparametric density estimation (NDE). Given $\boldsymbol{x}$ and assuming that the function being approximated is continuous and smooth, the expected value of y, ($E[y|\boldsymbol{x}]$) can be estimated as

$$E[y \mid \boldsymbol{x}] = \frac{\int_{-\infty}^{\infty} y\, f(\boldsymbol{x}, y)\mathrm{d}y}{\int_{-\infty}^{\infty} f(\boldsymbol{x}, y)\mathrm{d}y} \tag{2.23}$$

Using the training set and assuming Gaussian PDF, the function $f(\boldsymbol{x}, y)$ can be defined as

$$f(\boldsymbol{x}, y) = \frac{1}{(2\pi)^{(j+1)/2}\sigma^{(j+1)}}$$

$$\times \frac{1}{I}\sum_{i=1}^{I}\left[\exp\left(\frac{-(\boldsymbol{x}-\boldsymbol{x}_i)^T(\boldsymbol{x}-\boldsymbol{x}_i)}{2\sigma^2}\right)\right]$$

$$\times\left[\left(-\frac{(y-y_i)^2}{2\sigma^2}\right)\right] \tag{2.24}$$

where $\boldsymbol{x}_i$ and y respectively denote the ith training input vector and the corresponding output, and σ denotes the width (*smoothing parameter* or *spreading factor*) of the Gaussian PDF. Given $\boldsymbol{x}$, the corresponding regression estimate, $\hat{y}(\boldsymbol{x})$ can be determined as a conditional mean by substituting Eq. (2.23) in Eq. (2.22).

$$\hat{y}(\boldsymbol{x}) = E(y \mid \boldsymbol{x}) = \frac{\sum_{i=1}^{I} y_i h_i}{\sum_{i=1}^{I} h_i}; \qquad h_i = \exp\left[-\frac{d_i^2}{2\sigma^2}\right] \tag{2.25}$$

where h_i denotes the Gaussian radial basis function and d_i^2 represents the squared Euclidean distance between vectors $\boldsymbol{x}$ and x_i defined as

$$d_i^2 = (\boldsymbol{x}-\boldsymbol{x}_i)^T(\boldsymbol{x}-\boldsymbol{x}_i) \tag{2.26}$$

Given a test input vector, $\boldsymbol{x}$, the GRNN procedure for predicting the value of an mth output variable, y_m (m = 1, 2, . . ., M), can be viewed as computing the weighted average of all the target values of that variable, wherein weights are taken proportional to the Euclidean distances between the training input vectors and the test input vector. GRNN's input layer houses N nodes to serve as 'fan-out' units. The hidden layer contains J number of nodes, such that each hidden node represents a different training input vector. When an input vector is applied to the input nodes, its distance (d_i) from each of the I training vectors stored in the hidden nodes is computed, which is then transformed using the Gaussian RBF to compute the hidden unit output, h_j (j = 1, 2, . . . , J). GRNN's third layer consists of two types (I and II) of summation units. An m^{th} type-I unit (indexed as N_m and shown in dark border), computes the summation ($\sum_{j=1}^{J} y_j h_j$) defined in the numerator of Eq. (2.24), by utilizing the hidden unit outputs, h_j, and the mth elements of all the M-dimensional target output vectors, y_m. The single type-2 unit in the third layer performs summation of all hidden node outputs (see denominator of Eq. 2.24). Finally, the mth output layer node performs the normalization step

defined in Eq. (2.24) to compute the GRNN-predicted value of the *m*th output variable, $\hat{y}_m(\boldsymbol{x})$. GRNN's training algorithm uses only one adjustable (free) parameter namely the width (σ) of the Gaussian RBF. Significance of the width parameter is that as the value of this parameter becomes smaller (larger), the regression performed by the GRNN becomes more local (global). Hence, the magnitude of σ needs to be chosen judiciously as it significantly affects the accuracy of GRNN's predictions. The commonly employed technique for automatically selecting the optimal σ value is the 'leave-one-out' cross-validation method. In this technique, a single input–output vector is removed from the training set of I vectors and a GRNN model is built using the remaining I-1 vectors for predicting the outputs corresponding to the removed pattern. This procedure is repeated I times, by keeping aside each time a different training pattern, and the mean-squared-error (MSE) is evaluated by comparing the GRNN-predicted and the corresponding target output values. This procedure is repeated by varying the σ value systematically and the value that minimizes the MSE is chosen to be optimal.

D. Radial basis function neural network

The architecture (as shown in the Figure 2.3) of a radial basis function neural network (RBFN) [Tambe et al., 1996; Haykins, 1999] comprises three layers of nodes namely *input*, *hidden* and *output* layers. The input layer nodes, similar to an EBP network, serve only as "fan-out" units to distribute the inputs to the J number of hidden layer nodes. Each hidden node represents a kernel function that implements a non-linear transformation of an N-dimensional input vector. The commonly used kernel is the Gaussian RBF whose response is typically limited only to a small region of the input space where the function is centred. The Gaussian RBF is characterized by two parameters, namely *center* (C_j) and the *peak width* (σ_j). While C_j represents an N-dimensional vector, σ_j is a scalar determining the portion of the input space where the jth (j = 1, 2,…, J) RBF has a significant non-zero response. The centers are adjustable parameters and the nonlinear approximation and generalization characteristics of an RBFN depend critically on their magnitudes. Thus, centers must be selected judiciously. On the

other hand, the peak width parameter does not affect an RBFN's approximation and generalization performance significantly and thus these can be fixed heuristically.

For a given input vector, $\mathbf{x}_i$, the output of the jth Gaussian hidden node can be calculated as:

$$O_j = \Phi_j \cdot \left\| \mathbf{x}_i - C_j \right\| = \exp\left(- \left\| \mathbf{x}_i - C_j \right\|^2 \Big/ 2\sigma_j^2 \right) \qquad (2.27)$$

where, $\left\| \mathbf{x}_i - C_j \right\|$ denotes the Euclidian distance between $\mathbf{x}_i$ and C_j.

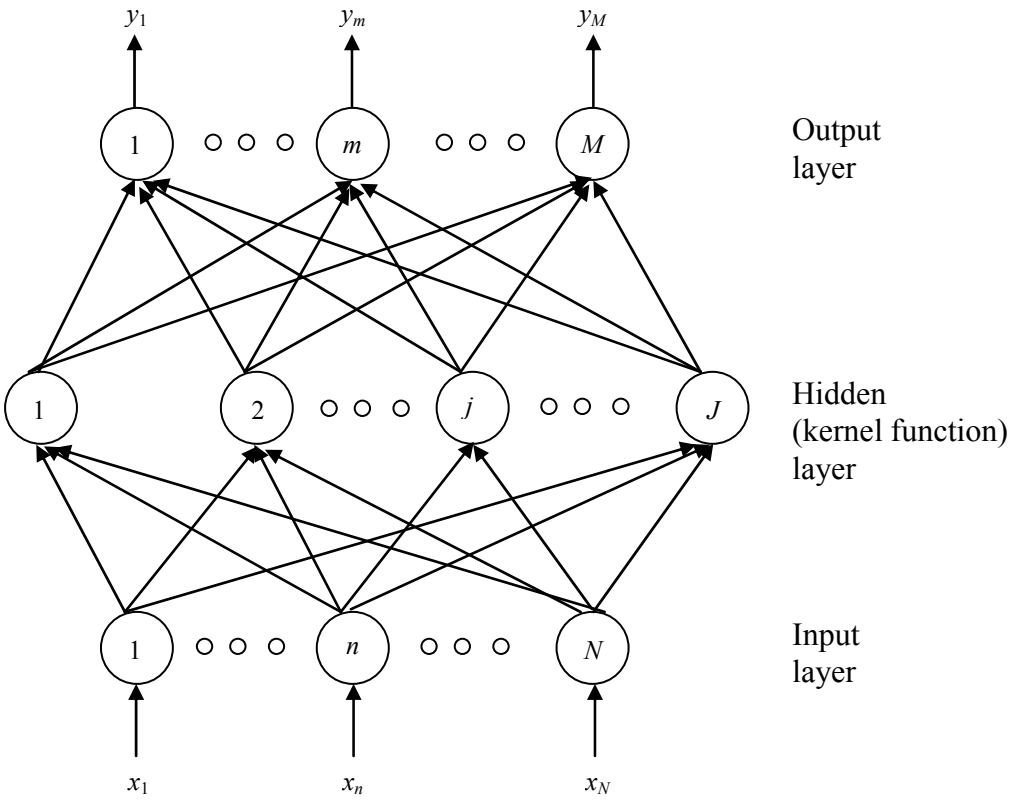

Figure 2.3: The schematic of Radial Basis Function Neural Network

The outputs of the Gaussian hidden nodes serve as inputs to the output nodes and the output of each output node is computed using a linear function of its inputs as given below:

$$\hat{y}_m = \sum_{j=1}^{J} w_{jm} O_j; \qquad m = 1, 2, ..., M \qquad (2.28)$$

where, $\hat{y}_m$ refers to the output of the mth output layer node; M denotes the number of output nodes and w_{jm} refers to the weight of the connection between jth hidden node and mth output layer node.

Development of an RBFN based model involves selecting the centers, peak widths, the number of hidden layer nodes (J) and the weights, w_{jm}. The centers can be selected using a number of methods [see e.g. Bishop, 1994] such as the random subset selection, K–means clustering [Moody et al., 1989]], orthogonal least-square learning algorithm [Chen et al., 1991] and rival penalizing competitive learning [Xu et al., 1993]. The width parameter can either be chosen same for all the hidden units or can be different for each unit. The width parameter can be set equal for all the hidden nodes. Once the centers and the widths of the RBFs are chosen, the weights, w_{jm}, on the connections between the hidden and output nodes are adjusted using a standard least-squares procedure with the objective of minimizing a pre-specified error function such as the sum-squared-error. Once trained, the magnitude of the response of each of the RBFs is a function of the distance between the network input ($\mathbf{x}_i$) and the RBF center, C_j. Finally, the output layer node combines these signals to produce the network output, $\hat{y}_m$.

2.2.2 Support Vector Regression

The support vector regression (SVR) is an adaptation of a recently introduced statistical/machine learning theory based classification paradigm namely, *support vector machines* [Vapnik, 1995; Vapnik et al., 1996; Burges, 1998; Smola, et al., 1998; Schölkopf, 2001]. In SVR, the inputs are first nonlinearly mapped into a high dimensional "feature" space (**Φ)** wherein they are correlated linearly with the outputs. This SVR characteristic distinguishes it from the common ANNs such as MLP that approximate the nonlinear input-output relationships directly. Other distinguishing features of the SVR vis-à-vis MLP are: (i) while the parameters of an SVR model are obtained by solving a quadratic optimization problem, the parameters (weights) of an ANN model are commonly estimated using a least-squares error minimization method such as the generalized delta rule based *error back propagation* (EBP) algorithm [Rumelhart et al., 1986],

(ii) in SVR, the objective function is of quadratic form, and thus it possesses a single minimum, which unlike an MLP network avoids the heuristic procedure involved in locating the global or the deepest minimum on the objective function surface, (iii) commonly, the SVR builds a multiple input-single output (MISO) model while MLP network are capable of simultaneously approximating multiple input – multiple output (MIMO) relationships, and (iv) in contrast to an MLP network, the SVR model and its parameters are amenable to interpretation in terms of training data.

The SVR formulation follows the *structural risk minimization* (SRM) principle, as opposed to the *empirical risk minimization* (ERM) approach commonly employed by the conventional statistical/machine learning methods, and also in developing the MLP models. In the ERM, a suitable measure of the prediction error such as the root-mean-square-error (RMSE), pertaining to the training data is minimized. Since the ERM formulation is based exclusively on the training set error, it does not guarantee a good generalization performance by the resultant model. On the other hand, the SRM feature equips the SVR model with a greater potential to generalize the input-output relationship learnt during its training phase. The improved generalization performance by the SRM strategy stems from creating an optimized model such that the prediction error and model complexity are concurrently minimized.

A. Regression formulation

To understand the working principles of the SVR formalism, we first formulate a general problem of regression estimate in the framework of the statistical learning theory. Consider a set of measurements (training data), $\hat{\boldsymbol{D}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{p}$, where $\mathbf{x}_i \in \mathfrak{R}^N$ is a vector of model inputs and $y_i \in \mathfrak{R}$, represents the corresponding scalar output. The objective of the regression analysis is to determine a function, $f(\mathbf{x})$, so as to predict accurately the desired (target) outputs, $\{y\}$, corresponding to a new set of input-output examples, $\{(\mathbf{x}, y)\}$, that are drawn from the same underlying joint probability distribution, $P(\mathbf{x}, y)$, as the training set, $\hat{\boldsymbol{D}}$. In essence, the task is to find a function, f, that minimizes the expected risk, $R[f]$, defined as:

$$R[f] = \int L\left(f(\mathbf{x}) - y\right) dP(\mathbf{x}, y) \tag{2.29}$$

where L denotes a loss function. For a given function, $f(\mathbf{x})$, the expected risk (test error) is the possible average error committed by the function on an unknown example drawn randomly from the sample probability distribution, $P(\mathbf{x}, y)$; the loss function, L, indicates how this error is penalized. In practice, the true distribution, $P(\mathbf{x}, y)$, is unknown and, therefore, Eq. 2.29 can not be evaluated. Thus, an inductive principle is used to minimize the expected risk. Here, a stochastic approximation to the $R[f]$, called *empirical risk* (R_{emp}) (see Eq. 2.29) is computed by sampling the data following which R_{emp} minimization is performed.

$$R_{emp} = \frac{1}{p} \sum_{i=1}^{p} L\left(f(\mathbf{x}_i) - y_i\right) \tag{2.30}$$

The empirical risk is a measure of the prediction error with respect to the training set, i.e., the difference between the outputs predicted by the function, $f(\mathbf{x}_i)$ and the corresponding target outputs, y_i (i = 1,2,…, p). It approaches the expected risk as the number of training samples goes to infinity, i.e.,

$$R_{emp}[f]_{p \to \infty} = R[f] \tag{2.31}$$

This however implies that for a small-sized training set, minimization of R_{emp} does not ensure minimization of $R[f]$. As a result, a selection of $f(\mathbf{x})$ based solely on the empirical risk minimization does not guarantee a good generalization performance (ability to predict accurately outputs of the test set) by the regression function. The inability to generalize originates from a phenomenon known as ‘over-fitting’. It occurs when the regression function—by way of higher model complexity—fits not only the mechanism underlying the training data but also the noise contained therein.

For overcoming the problem of over-fitting and thereby enhancing the generalization ability of the fitting function, $f(\mathbf{x})$, it is necessary to implement what is known as “capacity control”. The capacity of a regression model is a measure of its complexity. For instance, a very high-degree polynomial assuming a wiggly shape, which fits the training set exactly but does not generalize well outside the training data, has a high capacity [Vapnik, 1998]. In the SVR formalism described

below, a capacity control term is included to overcome the problem of function overfitting. The underlying idea is if we could choose a function (hypothesis)—from a low capacity function space—yielding a small empirical risk, then the true risk, $R[f]$, is also likely to be small.

To solve a nonlinear regression problem, the SVR formalism considers the following linear estimation function:

$$f(\mathbf{x}) = (\boldsymbol{w} \cdot \Phi(\mathbf{x})) + b \tag{2.32}$$

where, $\boldsymbol{w}$ denotes the weight (parameter) vector; b is a constant; $\boldsymbol{\Phi}(\mathbf{x})$ denotes a function termed *feature*, and $(\boldsymbol{w} \cdot \Phi(\mathbf{x}))$ describes the dot product in the feature space, $\boldsymbol{\Phi}$, such that Φ: $\mathbf{x} \rightarrow \boldsymbol{\Phi}$, $\boldsymbol{w} \in \boldsymbol{\Phi}$. In SVR, the input data vector, $\mathbf{x}$, is first nonlinearly mapped into a high-dimensional feature space, $\boldsymbol{\Phi}$, and a linear regression is performed in this space for predicting the output, y. Thus, the problem of nonlinear regression in the lower dimensional input space is transformed into a linear regression problem into a high dimensional feature space. In essence, the original optimization problem involving a nonlinear regression is recasted as searching the flattest function in the feature space, $\boldsymbol{\Phi}$, and not in the input space, $\mathbf{x}$.

To avoid over-fitting of the regression model and thereby improving its generalization capability, the SVR formalism minimizes the following regularized risk functional comprising the empirical risk and a complexity term, $\|\boldsymbol{w}\|^2$:

$$R_{reg}[f] = R_{emp}[f] + \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{2.33}$$

where R_{reg} denotes the regression risk and $\|.\|$ is the Euclidean norm. The minimization of the regression risk R_{reg} leads to penalization of the model complexity while simultaneously keeping the empirical risk small. The regularization term, $\frac{1}{2}\|\boldsymbol{w}\|^2$, in Eq. 2.33 controls the trade-off between the complexity and approximation accuracy of the regression model to ensure that it possesses an improved generalization performance. Specifically, the complexity of the linear function is controlled by keeping $\boldsymbol{w}$ as small as possible.

Equation 2.33 is similar to the cost function augmented with a standard weight-decay term used in developing the ANN models possessing good generalization ability. This approach decreases the complexity of an ANN model by limiting the growth of the network weights via a kind of weight-decay. Specifically, the weight-decay method prevents the weights from growing too large unless it is really necessary [Krogh, 1995]. This is achieved by adding a term to the cost function that penalizes the large weights. The resultant form of the cost function is [Krogh, 1995; Hertz, 1991],

$$E = E_0 + \frac{1}{2}\gamma \sum_{i,j} w_{ij}^2 \tag{2.34}$$

where, E and E_0 denote the modified and original cost functions, respectively, γ is a parameter governing how strongly the large weights are penalized and w_{ij} are the weights on the connections between ith and jth network nodes. The commonly used procedure for (such as the error-back-propagation (EBP) algorithm) minimizes only the E_0, which in most cases represents the sum-squared-error (SSE) with respect to the training set. The EBP updates the weights using the following equation:

$$w_{ij}^{new} = w_{ij}^{old} - \eta \frac{\partial E_0}{\partial w_{ij}} \tag{2.35}$$

where η denotes the learning rate. A comparison of the respective terms of Eqs 2.33 and 2.34 indicates that minimization of the regression risk attempted by the SVR is similar to the minimization conducted by the ANNs of a cost function comprising a weight decay (penalty) term. However, the SVR and ANNs use conceptually different approaches for minimizing the respective cost functions.

A number of cost functions such as the Laplacian, Huber's, Gaussian and ε-insensitive can be used in the SVR formulation. Among these, the robust ε-insensitive loss function (L_ε) [Vapnik, 1998] (see Figure 2.4), given below is commonly used.

$$L_{\varepsilon}\left(f(\mathbf{x})-y\right)=\begin{cases}\left|f(\mathbf{x})-y\right|-\varepsilon & \text{for } \left|f(\mathbf{x})-y\right|\geq\varepsilon \\ 0 & \text{otherwise}\end{cases} \tag{2.36}$$

where ε is a precision parameter representing the radius of the tube located around the regression function, $f(\mathbf{x})$ (see Figure 2.4). The region enclosed by the tube is known as 'ε-insensitive zone, since the loss function assumes a zero value in this region and as a result it does not penalize the prediction error with magnitudes smaller than ε.

The minimization of the empirical risk using the symmetric loss function (defined in Eq. 2.36) is equivalent to adding the slack variables, ξ_i and ξ_i^*, i = 1, 2, …p, into the functional, $R[f]$, with a set of linear constraints. The slack variables ξ_i and ξ_i^* measure the deviation $\left(y_i - f(\mathbf{x}_i)\right)$ from the boundaries of the ε-insensitive zone. Thus, using the ε-insensitive loss function and introducing the regularization constant, C, the optimization problem in Eq. 2.33 can be written as,

Minimize: $$\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{p}\left(\xi_i + \xi_i^*\right) \tag{2.37}$$

subject to, $$\left.\begin{cases}\left(\boldsymbol{w}\cdot\Phi(\mathbf{x}_i)\right)+b-y_i\leq\varepsilon+\xi_i^* \\ y_i-\left(w\cdot\Phi\left(\mathbf{x}_i\right)\right)-b\leq\varepsilon+\xi_i \\ \xi_i,\xi_i^*\geq 0 \qquad \text{for } i=1,\ldots,p\end{cases}\right\} \tag{2.38}$$

While conducting this minimization, the SVR optimizes the position of the ε-tube around the data as shown in Figure 2.4. Specifically, the optimization criterion in Eq. 2.38 penalizes those training data points whose y values lie more than ε distance away from the fitted function, $f(\mathbf{x})$. In Figure 2.4, the stated excess positive and negative deviations are illustrated in terms of the slack variables, ξ and ξ^*, respectively. These variables assume non-zero values outside the [ε, -ε] region. While fitting $f(\mathbf{x})$ to the training data, the SVR minimizes the training set error by minimizing not only ξ_i and ξ_i^*, but also $\|\boldsymbol{w}\|^2$ with the objective of increasing the flatness of the function or penalizing its over-complexity. This serves to avoid an under-fitting as also over-fitting of the training data.

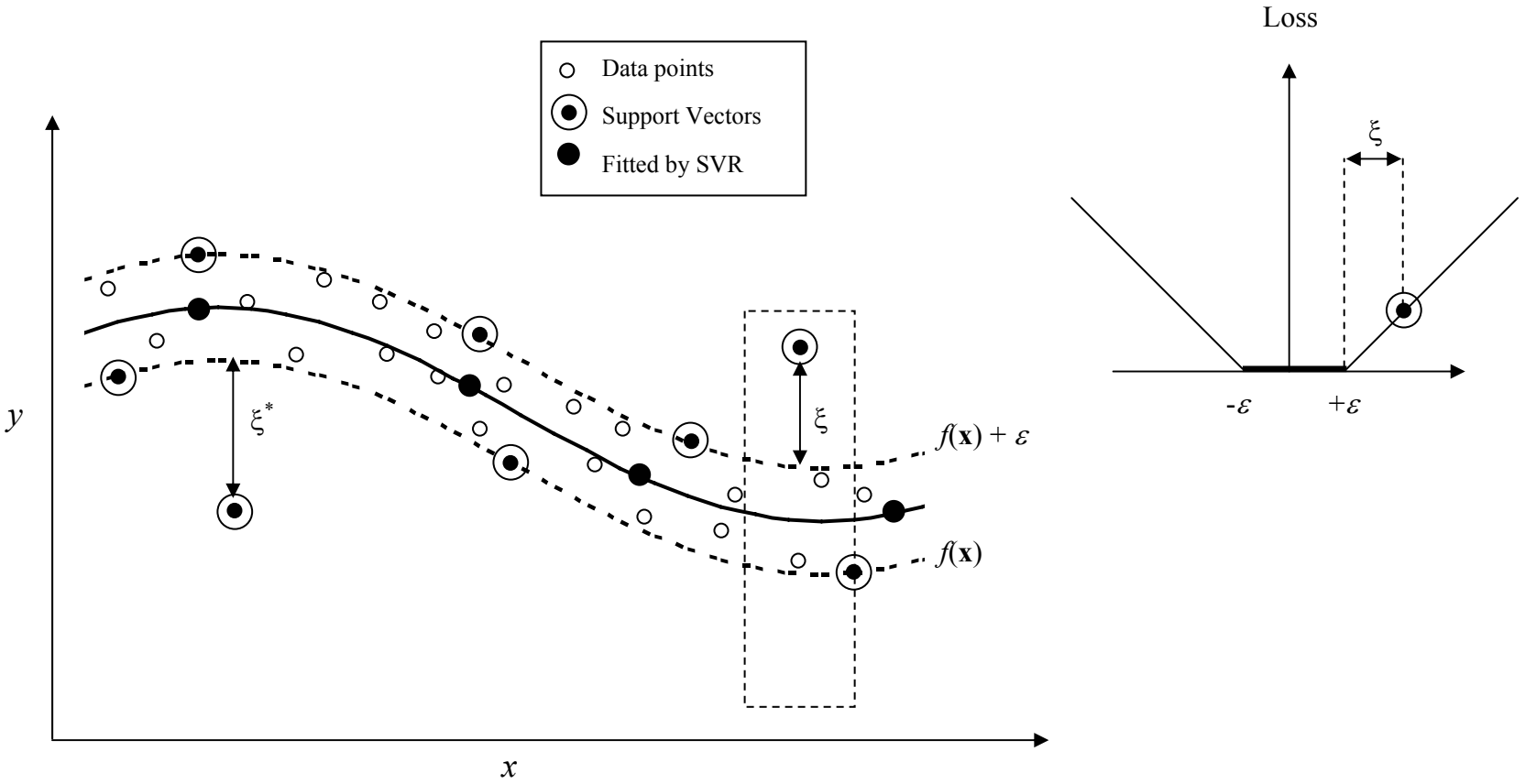

Figure 2.4: A schematic representation of the SVR using ε-insensitive loss function

It was demonstrated by Vapnik [1998] that the function defined below possessing a finite number of parameters can minimize the regularized risk functional in Eq. 2.37.

$$f\left(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*\right) = \sum_{i=1}^{p}\left(\alpha_i - \alpha_i^*\right)\left(\Phi\left(\mathbf{x}_i\right)\cdot\Phi\left(\mathbf{x}\right)\right) + b \tag{2.39}$$

where, α_i and α_i^* (both ≥ 0) are the coefficients (known as "Lagrange multipliers") pertaining to the input data vector $\mathbf{x}_i$ and satisfying

$\alpha_i\alpha_i^* = 0$, $i = 1, 2, \ldots, p$.

It can be noticed noted that both the optimization problem (Eq. 2.37 and Eq. 1.38), and its solution (Eq. 1.39) involves a computation of the dot product in the feature space, $\mathbf{\Phi}$. These computations become time consuming and cumbersome when $\mathbf{\Phi}$ is high-dimensional. It is however possible to use, what is known as the "kernel trick" to avoid computations in the feature space. This trick uses the Mercer's condition, which states that any positive semi-definite, symmetric kernel function, K, can be expressed as a dot product in the high-dimensional space. The advantage of using a kernel function is that the dot product in the feature space can now be computed without actually mapping the vectors, $\mathbf{x}$ and $\mathbf{x}_i$ into that space. That is, using a kernel function all the necessary computations can be performed implicitly in the input space instead of the feature space.

Although several choices for the kernel function K are available, the most widely used kernel function is the radial basis function (RBF) defined as,

$$K\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(\frac{-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\sigma^2}\right) \tag{2.40}$$

where, σ denotes the width of the RBF. A list of other possible kernel functions is given in Table 2.1.

Table 2.1: List of possible kernel functions [Dibike, 2000]

1.	Simple dot product	$K(x_i, x_j) = (x_i \cdot x_j)$
2.	Simple polynomial kernel: d, degree of polynomial	$K(x_i, x_j) = ((x_i \cdot x_j) + 1)^d$
3.	Vovk's real polynomial:	$K(x_i, x_j) = \frac{1-(x_i \cdot x_j)^d}{1-(x_i \cdot x_j)}$
4.	Radial basis function: λ is user defined	$K(x_i, x_j) = \exp(-\lambda \lvert x_i - x_j \rvert^2)$
5.	Two layer neural network: b and c are user defined	$K(x_i, x_j) = \tanh(b(x_i \cdot x_j) - c)$
6.	Linear splines:	$K(x_i, x_j) = \prod_{k=1}^{n} (x_i^k \cdot x_j^k)$
7.	Semi local kernel: d and σ are user defined	$K(x_i, x_j) = [(x_i \cdot x_j) + 1]^d \frac{\exp(-\lVert x_i - x_j \rVert^2)}{\sigma^2}$

We can now replace the dot product in Eq. 2.39 with a kernel function (Eq. 2.40) and write the general form of the SVR-based regression function as,

$$f(\mathbf{x}, \boldsymbol{w}) = f(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = \sum_{i=1}^{p} (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) K(\mathbf{x}, \mathbf{x}_i) + b \qquad (2.41)$$

where, the weight vector $\boldsymbol{w}$ is expressed in terms of the Lagrange multipliers $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}^*$. The values of these multipliers are obtained by solving the following convex quadratic programming (QP) problem.

Maximize:

$$R(\boldsymbol{\alpha}^*, \boldsymbol{\alpha}) = -\frac{1}{2} \sum_{i,j=1}^{p} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) K(\mathbf{x}_i, \mathbf{x}_j) - \varepsilon \sum_{i=1}^{p} (\alpha_i^* + \alpha_i) + \sum_{i=1}^{p} y_i (\alpha_i^* - \alpha_i) \qquad (2.42)$$

subject to constraints: $0 \le \alpha_i$, $\alpha_i^* \le C$, $\forall i$, and $\sum_{i=1}^{p}\left(\alpha_i^* - \alpha_i\right) = 0$. The bias parameter b in Eq. 2.41 can be computed as

$$b = \begin{cases} y_i - f\left(\mathbf{x}_i\right)_{b=0} - \varepsilon \text{ for } \alpha_i \in (0, C) \\ y_i - f\left(\mathbf{x}_i\right)_{b=0} + \varepsilon \text{ for } \alpha_i^* \in (0, C) \end{cases} \tag{2.43}$$

B. Interpreting the structure and coefficients of SVR model

A significant feature of the SVR is that the regression model and its parameters can be interpreted geometrically. In SVR, each training point is associated with it a pair (α_i and α_i^*) of parameters values. The α_i and α_i^* values have an intuitive explanation as forces pushing and pulling the regression function, $f(\mathbf{x}_i)$, towards the desired output, y_i [Muller et al., 1997]. Owing to the specific character of the QP problem defined in Eq. (2.42), only some of the regression coefficients, ($\alpha_i - \alpha_i^*$), assume non-zero values. The training input vectors, $\mathbf{x}_i$, with non-zero coefficients are termed "support vectors (SVs)". Alternatively, SVs are those training input-output data ($\mathbf{x}_i$, y_i) for which $\left|f\left(\mathbf{x}_i\right) - y_i\right| \ge \varepsilon$. Since they are the only points that determine the SVR-approximated function, the SVs are crucial data examples. In Figure 2.4, the SVs are depicted as the points lying on the surface and outside of the ε-tube. As the percentage of SVs decreases a more general regression solution is obtained. Also, a lesser number of computations are necessary to evaluate the output of a new and unknown input vectors when the percentage of SVs becomes smaller. The data points lying inside the tube are considered to be correctly approximated by the regression function. The training points with the corresponding α_i and α_i^* equal to zero have no influence on the solution to the regression task. If these points are removed from the training set, the solution obtained would still be same [Thissen et al., 2003]. This characteristic, owing to which the final regression model can be defined as a combination of a relatively small number of input vectors is known as "sparseness" of solution.

C. Tuning of SVR's algorithmic parameters

The prediction accuracy and generalization performance of an SVR-based model is controlled by two free parameters namely, C and ε. These parameters

therefore should be selected judiciously. Among two parameters, the former dictates a trade-off between the model complexity and the approximation error, while the latter determines the width of the ε-insensitive zone used for fitting the training data (see Fig. 2.4). The parameter C in essence determines the amount up to which the prediction errors beyond the magnitudes $\pm\varepsilon$ are tolerated. If the magnitude of C is too large (infinity), then the SVR minimizes only the empirical risk without regard to the model complexity. On the other hand, for a too small value of C, the SVR algorithm assigns an insufficient weightage to fitting the training data thereby improving the chances of a better generalization performance by the model [Drucker et al., 1997]. The tube width parameter ε can inversely affect the number of support vectors used to construct the regression function. As ε decreases, the percentage of training points marked as SVs (hereafter denoted as *%SV*) increases which in turn also enhance the complexity of the SVR model. With complex models there always exists a risk of over-fitting the training data and consequently a poor generalization performance by the model. On the other hand, a relatively better generalization performance is realized for large ε magnitudes at the risk of obtaining a high training set error. It may be noted that in ANNs and traditional regression, ε is always zero and the data set is not mapped into higher dimensional spaces. Thus, the SVR is a more general and flexible treatment of the regression problems [Chen et al., 2001]. A number of guidelines for the judicious selection of C and ε are provided by Cherkassky and Ma [2004].

2.2.3 Genetic Programming

An important AI based modeling technique known as Genetic Programming (GP) was proposed by Koza [1992, 1994]. Originally, the GP formalism was developed for generating automatically task-specific computer programs without manually coding them rigorously. The GP concept was later extended to automatically obtain a mathematical model that fits a given set of model's input-output data. It is closely related to an AI-based stochastic search and function optimization method viz. genetic algorithms (GA) which is described in another section 2.4.3. Both GP and GA are founded on the principles of natural selection and genetics followed by biologically evolving species. However, they differ in their applications. While the GA methodology is used for function

maximization/minimization, the GP technique discovers a system-specific form of a data fitting function and all of its necessary parameters or at least an approximation of these. The GP technique performs what is known as 'symbolic regression' to search and optimize the form and parameters of an appropriate data fitting function. Although conceptually attractive, the GP is relatively less explored AI-based modeling formalism as compared to the ANNs. The GP has been used in the steady-state modeling [McKay et al., 1997; Willis et al., 1997; Grosman, et al., 2004], dynamic modeling and time series prediction [Iba et al., 1993; Gray et al., 1996] process identification [Kulkarni et al., 1999] and process design [Lakshminarayanan et al., 2000]. In a novel application, the GP has been used to obtain the mechanistic map equations of simple chaotic systems such as logistic, Henon and Universal maps [Yadavalli et al., 1998]. In another contribution, Zhang and Muhlenbein [1993] used GP to optimize both the architecture and the connection weights of a feedforward artificial neural network. Their results indicated that given enough resources the GP methodology could determine minimal complexity networks. While this is an interesting result in itself, it fails to fully exploit the power of GP. Rather than manipulating neural network structures, there is potential to discover significantly more information about the underlying process characteristics by the direct use of symbolic regression.

Similar to GAs, that performs function minimization/maximization, the GP formalism uses *selection*, *crossover* and *mutation* operators to obtain the structure (form) and all the necessary parameters of a best-fitting linear/nonlinear function. Accordingly, the solution given by the two formalisms also differ. That is, while the GP searches, a model as a solution to a given data-fitting problem, GA searches and optimizes the decision variables that minimize/maximize a pre-specified objective function. In its procedure, the GAs manipulate only the numbers while GP manipulates symbols (structure of the data-fitting model) as also numbers (for obtaining parameters of the fitting function). The sequence of steps that the GP [Nandi, 2005] follows for obtaining a best-fitting model is given below (also refer Fig 2.10).

Begin

Initialize a random population of candidate solutions (models)

Fitness evaluation of individual models

Repeat

Genetic Operations

Selection

Reproduction

Crossover

Mutation

Fitness evaluation

Until the terminating criterion

Return results

End

Step 1: Initialization of population

GP paradigm first creates a random initial population of a number of potential candidate solutions to a given MISO modeling problem. Each candidate solution is represented using a tree like structure (see Fig. 2.5) that consists of two types of elements namely, the "functional" and the "terminal" elements [McKay, 1997]. A terminal represents operands of a model. These are leaves (nodes without branches) describing input variables or parameters of the data-fitting model. The functional elements are nodes (with branches) representing mathematical operators. A single tree represents the right hand side (RHS) of a data fitting function, $y = f(X, \alpha)$, where X is a vector of input variables and α denotes parameters of the function, f. Initialization of the population of candidate solutions to the problem is done as follows.

- Starting from its root, every node of a tree is chosen randomly either as a functional or a terminal element. Accordingly, the initial population is a blind random search of the solution space of an MISO modeling problem comprising function and parameter spaces.
- If a randomly chosen node is a terminal one, then a parameter or an input variable is randomly assigned to it.
- If the node chosen randomly comes out to be a functional one, i.e., an intermediate node, then a mathematical operator is chosen randomly, and that node is assigned a number of branches depending upon weather the operator is unary or binary. If the selected operator is a unary (e.g., *sine*, *cosine*, *tangent*, *exponential*, *log*, *ln*, etc) then the node is assigned a single branch in the level below the node. In the case of a binary operator (e.g., *addition*, *subtraction*, *multiplication*, *division*, etc.) the node fans out into two branches in the next level. When the depth of the tree reaches the pre-specified maximum depth d_{max}, then a terminal element is selected in that node. This way the initial individual candidate solutions are generated subject to a pre-specified maximum depth d_{max}.
- In Figure 2.5, a tree of depth equal to two is shown. It consists of three functional elements describing as many mathematical operators and three terminals (2.5, x_1 and x_2) representing function operands. This candidate solution can be interpreted as,

$$y = \sin(x_1) + (1.5 * x_2) \tag{2.44}$$

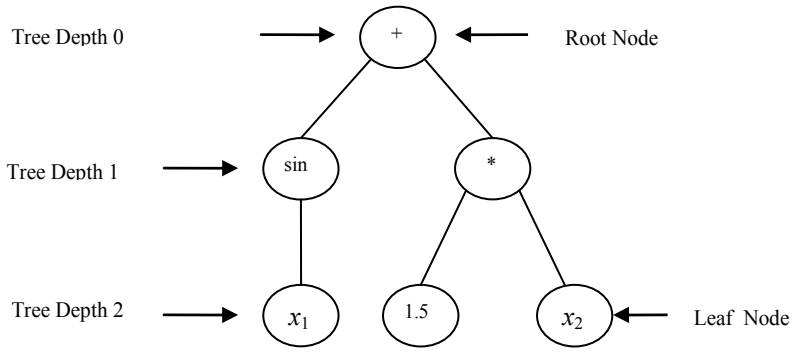

Figure 2.5: A simple equation tree

A candidate solution population is generated in a manner similar to Eq. (2.44). All the individual solutions in the population are syntactically consistent, valid and executable functions. Also, these solutions upon undergoing the genetic operations produce syntactically consistent, valid and executable candidate solutions.

Step 2: Fitness evaluation

Having created the initial population, each candidate solution thereof is assessed for its data-fitting ability. This is done using a "fitness" function and thereby evaluating the goodness of each candidate solution in approximating the relationship existing between the given set of inputs and the corresponding output. Fitness is a numeric value assigned to each member of a population to provide a measure of the goodness of a solution to the MISO modeling problem under study. Fitness functions are generally based upon the error between the actual and model predicted outputs (e.g., RMSE function). The error-based measures possess lower magnitudes for better solutions. In GP, it is desired to obtain fitness values that exhibit high magnitudes for better solutions during the evolutionary process. In order to modify the error-based performance index, a scaled inverse transformation is generally used. For symbolic regression problems, South et al. [1995] proposed the use of the correlation coefficient (CC) between the actual and model predicted outputs as an alternative to error-based fitness functions. Once fitness scores of all the candidate solutions in the current population are evaluated, the solutions are sorted in the decreasing order of fitness scores. The CC ranges between -1 (poor fit) to +1 (best fit). Accordingly, the solutions providing better approximations to the underlying MISO relationship are ranked higher in the fitness hierarchy.

Step 3: Selection

A number of selection methods have been suggested in the literature. These include *elitist* strategy, *Roulette Wheel* (RW) selection, *Tournament* selection and *fitness proportionate* selection [Deb, 2001; Deb et al., 2002;

Goldberg, 1989]. With the elitist scheme, the population is sorted into descending order according individual fitness values. The fittest M ($M \leq N$) individuals then undergo reproduction. Tournament selection involves random sampling (with replacement) of a fixed number of individuals from the parent population to form a subset. The fittest member of this relatively small subset is then chosen to reproduce, and the process is repeated as required. With fitness proportionate selection, an individual is sampled from the parent population (again with replacement) with a probability proportional to its fitness. Thus, if the ith individual in the parent population has a fitness f, the probability of this individual being chosen is,

$$\text{Probability (selection)} = \frac{f}{\sum f_i}, \qquad \text{where } i = 1,...,N. \tag{2.45}$$

Among the above-described techniques, the fitness proportionate selection appears to be a favored selection technique within the GA literature. The population that results upon selection is termed "mating" or "parent" pool

Step 4: Genetic operators

There exist two major genetic operators, namely *crossover* and *mutation* that are used frequently. These operators are implemented as given below. Implementation of steps 4(a) to 4(c) creates a new population of candidate solution that replaces the current one.

Step 4(a): Crossover

In this step, a crossover operation is conducted between pairs of candidate solutions from the current mating pool to produce two new candidate solutions termed as "offspring". In crossover, the genetic material (content of functional and terminal nodes) of a pair of parents is interchanged to produce offspring. Figure 2.6 and Figure 2.7 illustrate the crossover operation on a pair of parents. Here, the crossover nodes from the individual parents are selected randomly and the sub-trees originating from these nodes are mutually exchanged to form two new

candidate solutions (offspring). The two offspring created thereby replace the parents in the new generation.

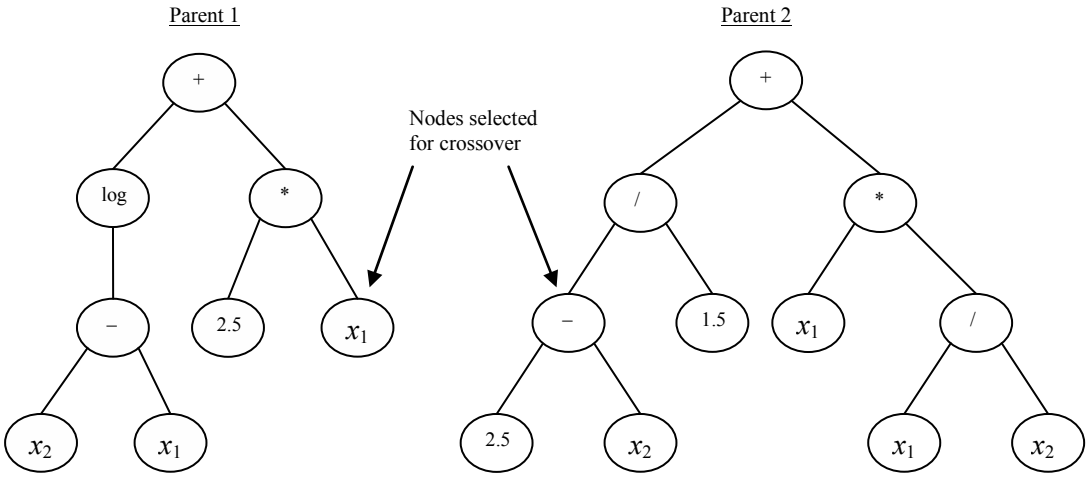

Figure 2.6: Parents selected for crossover and randomly selected crossover nodes.

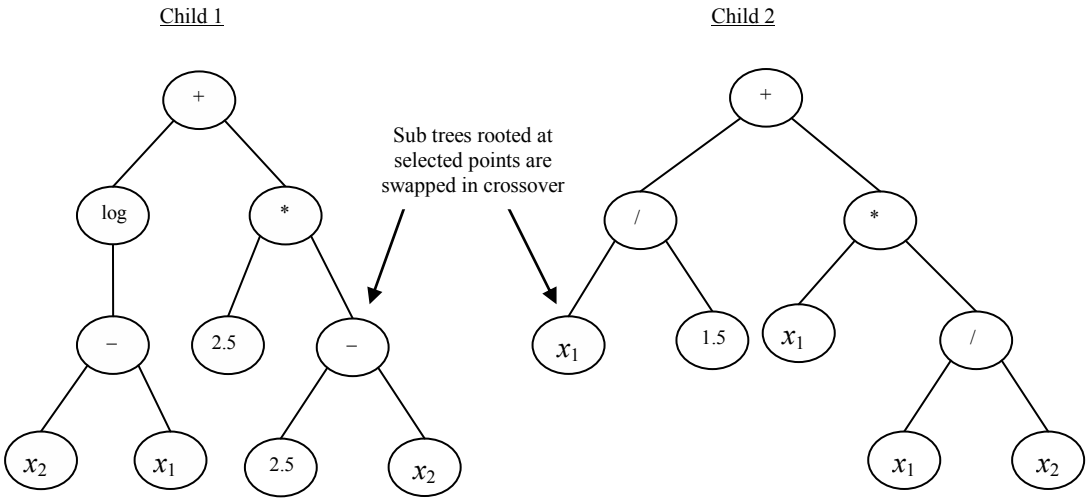

Figure 2.7: Offspring produced after the crossover operation.

Step 4(b): Mutation

Mutation operation is performed on post-crossover population. Candidate solutions of the population are individually subjected to mutation operation. Specifically, few nodes from a candidate solution tree are chosen randomly and

their contents are subjected to mutation. If a node happens to be an operator, then another randomly selected operator replaces it. If the chosen node contains an operand (i.e., a variable or a model parameter or a constant), then it is replaced by another randomly chosen operand. The principal aim of the mutation is to introduce a small change in the selected candidate solution. The new offspring created by the mutation operation replaces the candidate solution that has undergone mutation in the next generation. The mutation operation is illustrated in Figure 2.8.

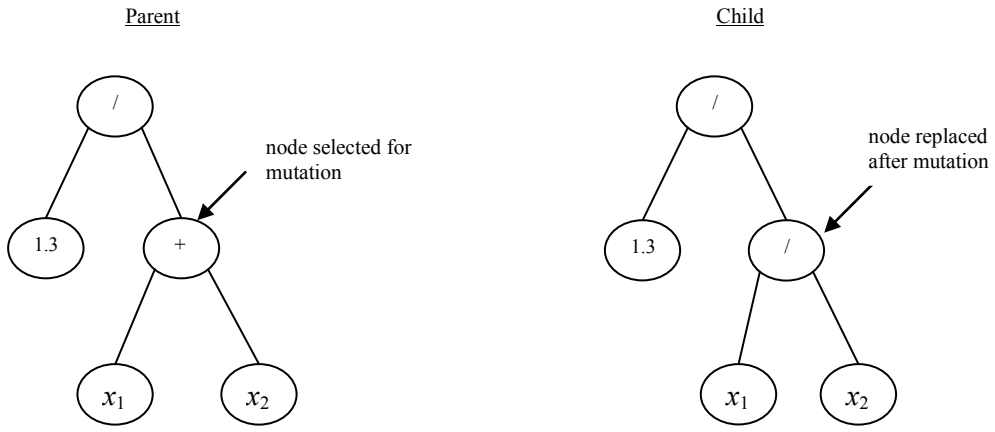

Figure 2.8: Example trees showing mutation operation

Upon executing the above-described three genetic operations on the current population, the resulting new generation population of offspring replaces the current one. Steps 2 to 4 are repeated over a large number of generations till a convergence criterion is satisfied. The commonly used criteria are, the GP has evolved over a pre-specified number of generations or the fitness value of the best candidate solution either remains constant or nearly constant over a pre-specified number of generations. The candidate solution tree with highest fitness at the convergence represents the best solution to the given modeling problem.

A significant drawback of the GP formalism is that in order to arrive at an appropriate fitting function it performs a global search of the function and the corresponding parameter space. Invariably, this search becomes time-consuming and numerically expensive. Also, the algorithm has a tendency to get stuck into a local minimum in the function and parameter spaces leading to a sub-optimal

convergence to a poorly performing fitting function. This happens as there is no mechanism for a candidate solution to escape once it gets stuck into a locally optimal solution. As a result, a good solution even if located in the neighborhood of a candidate solution remains unexplored. One remedy to overcome this problem is to subject the converged candidate solution to a nonlinear regression (NLR) method such as the Marquardt's algorithm with a view of fine-tuning. This however requires additional numerical effort. Also, the NLR approach does not fine-tune the structure of the candidate solution as it optimizes only its parameters. Another approach that is conceptually similar to that employed by the memetic algorithms (see Section 2.4.4) is to perform a local search in the neighborhood of each candidate solution or a number of high fitness scoring candidate solutions. This type of local search in both the functional and parameter space is expected to locate a better solution and/or speed-up the convergence. The AI Systems Group (AISG) at National Chemical Laboratory (NCL), Pune, India, has augmented the global search of the GP with a local search element to introduce an efficient symbolic regression formalism. The salient features of this method are described below.

Step 5: Local search

This is an advanced operator which essentially searches the solution tree's local neighborhood area to ultimately locate a better solution. The search could be applied at two tree locations namely function and parameter nodes. The details of these steps are as follows.

Step 5(a): Local search in function space

Each candidate solution of the population following the mutation operation is subjected to the local search in the function space. Here, a local search for an improved solution in the neighborhood of the candidate solution is carried out by slightly altering the functional form of that solution. In this step, the functional nodes of individual solutions are perturbed a specified number of times. This perturbation operation is conducted only on a specific number of functional nodes

which are selected according to some constraint. For example, the nodes in the maximum tree-depth and the ones in the depth equal to (maximum depth – 1) are selected to undergo the perturbation operation. Application of such a constraint on the selection of perturbation-undergoing nodes is done in order to bring out minor changes in the functional form of the tree which is akin to the local search in the function space of the candidate solution. The selected nodes are then subjected to perturbation operation for a fixed number of times and each time a new neighborhood solution with a slightly different functional form is generated. The resultant solution trees are then evaluated for their fitness and using Tabu search method (refer to Section 2.4.2), the best neighbor of the candidate solution is determined. This new solution tree then replaces the original un-perturbed solution tree in the next generation. Let us consider an example tree undergoing a local search in the functional space. The nodes selected for local search operation are shown in the Figure 2.9. Following a local search in the function space, the modified candidate solution population is subjected to the local search in parameter space.

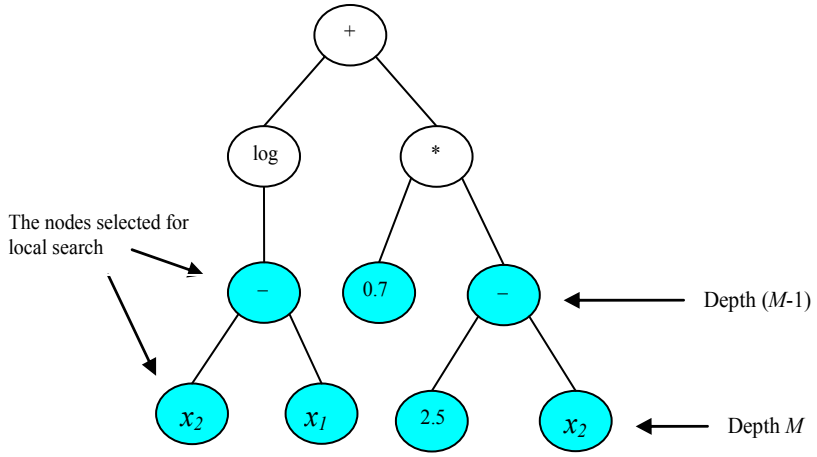

Figure 2.9: A candidate solution tree with nodes selected for local search operation

Step 5(b): Local search in the parameter space

Here, initially the set of parameters occurring in the maximum depth and one prior to that of each candidate solution tree selected and placed in an array. The individual array element of these parameters is then subjected to

Simultaneous Perturbation Stochastic Approximation (SPSA) [Spall, 1987, 1998a, b] (refer to Section 2.4.1) which is a simple yet numerically efficient stochastic parameter estimation formalism that mostly finds a locally optimal solution. Next, the fitness of the candidate solution with SPSA-optimized parameter is evaluated. If this fitness is higher than the fitness of the original-perturbed solution then the SPSA-optimized parameters replace the parameters in the original tree. The local search in the parameter space is illustrated in Figure 2.10. The stated search in the parameter space of a candidate solution is performed for a fixed number of times which equals that for the local search in the function space. The number of neighbors to be generated is a user-defined quantity.

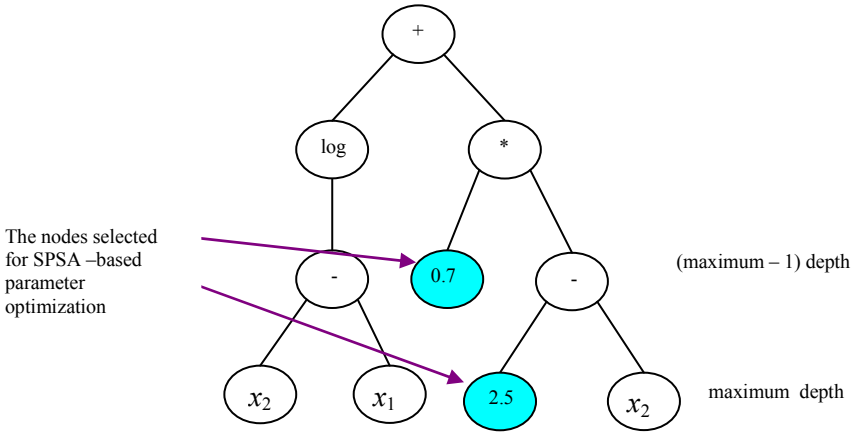

Figure 2.10: A candidate solution tree with nodes selected for SPSA-based parameter estimation

Following the local search, a decision must be made as to which members of the old population should die to make room for the next generation. Several methods may be adopted. The simplest procedure is to replace the entire old population so that there is a complete turnover for every cycle of the algorithm. However, it may be expedient to retain the fittest members of the old population in order to ensure the survival of structures that perform well. Thus a proportion P_{old}, of the original population is preserved, leaving $P_{gap} = (1 - P_{old})$ as a generation gap (the proportion to be replaced with new individuals). The above-described procedure is repeated for pre-specified maximum number of generations or till the fitness value stabilizes.

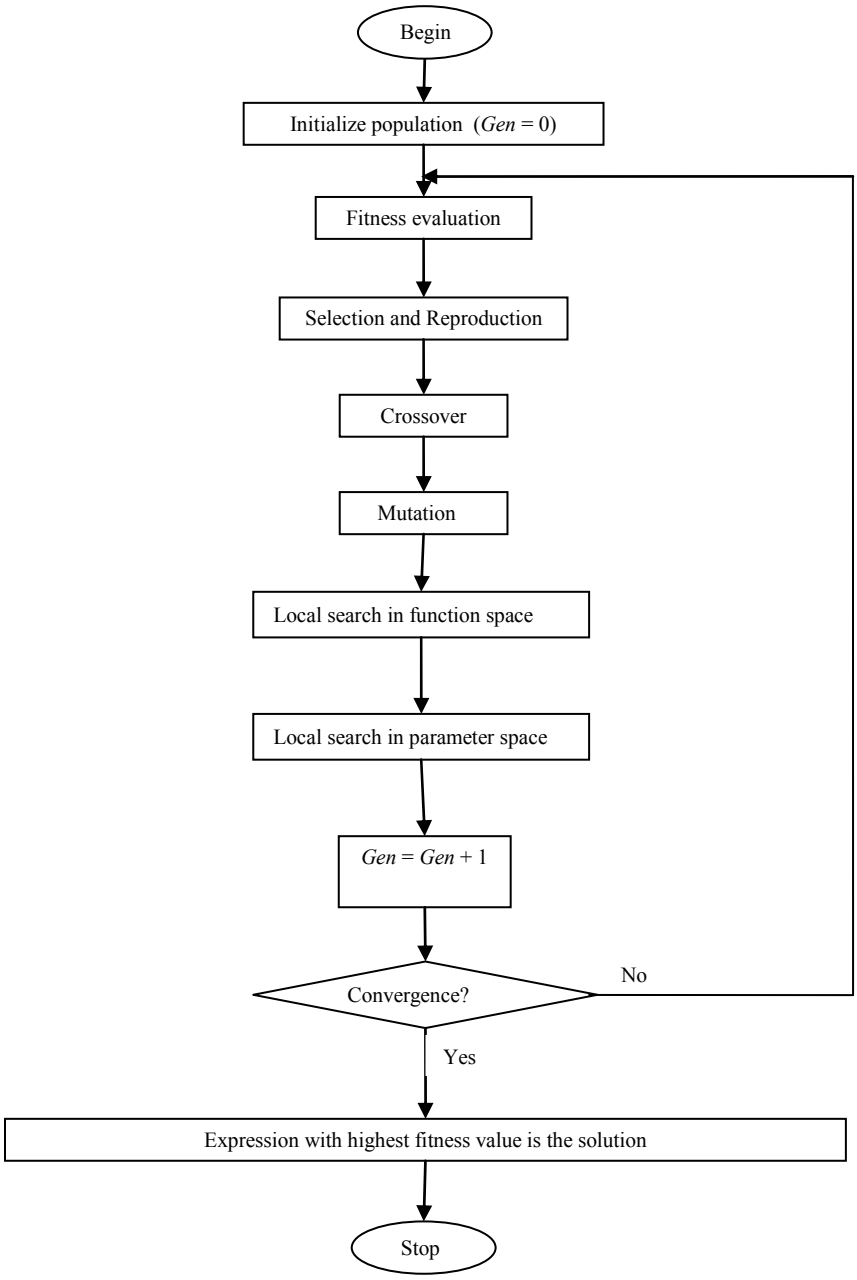

Figure 2.11: Flow chart of genetic programming

Thus in the new GP variant, the population of candidate solutions is refined each time before it enters the new generation. This saves the evolution from getting saturated (local minima entrapment) after a few generations (refer flow chart in Figure 2.11 for the complete operational flow designed for the GP with the local search).

2.3 CLASSIFICATION/CLUSTER ANALYSIS

Clustering is the classification of similar objects into different groups, or more precisely, the partitioning of a data set into subsets (called as “clusters”), so that the data in each subset (ideally) share some common trait - often “proximity” according to some defined distance measure. Clustering can be considered the most important unsupervised learning problem. As every other problem of this kind, it deals with finding a structure in a collection of unlabeled data. Data clustering is a common technique for statistical data analysis, which is used in many fields, including machine learning, data mining, pattern recognition, image analysis, chemo informatics and bioinformatics.

A *cluster* is a collection of objects which are “similar” between them and “dissimilar” to the objects belonging to other clusters.
We can illustrate this with a simple graphical example described in the following figure (Figure 2.12).

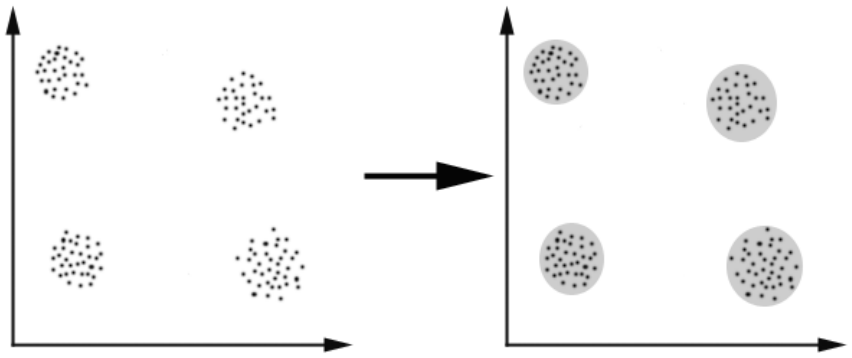

Figure 2.12: Simple graphical clustering example

From the example depicted in Figure 2.12, we can easily identify 4 clusters into which the data can be divided; the similarity criterion for defining a cluster is *distance*: two or more objects belong to the same cluster if they are “close” according to a given distance criterion (in this case Euclidean/geometrical distance).

2.3.1 K-Means Clustering

K-means is one of the simplest unsupervised learning algorithms that solve a clustering problem [MacQueen, 1967]. It follows a simple and easy procedure to classify a given data set into a certain number of clusters (assume *k* clusters) fixed *a priori*. The main idea is to define *k* centroids, one for each cluster. These centroids should be placed in an intelligent way since variability in their locations causes different results. Thus, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is complete and an early grouping is done. At this point it is necessary to re-calculate *k* new centroids as barley centers of the clusters resulting from the previous step. Having obtained these *k* new centroids, a new binding has to be done between the same data-set points and the nearest new centroid. Accordingly, a loop is generated as a result of which the *k* centroids change their location step by step until no more changes occur. In other words, centroids do not move any more.

Finally, this algorithm aims at minimizing an *objective function*; in this case a *squared error function* defined as

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2 \qquad (2.46)$$

where $\left\| x_i^{(j)} - c_j \right\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centre, c_j, which indicates the distance of *n* data points from their respective cluster centers.

The K-means algorithm is composed of the following steps:

1. Place *k* points into the space represented by the objects that are being clustered. These points represent initial group centroids.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the *k* centroids.
4. Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

Although it can be proved that the procedure will always terminate, the K-means algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. The algorithm is also significantly sensitive to the initial randomly selected cluster centers. The K-means algorithm can be run multiple times to reduce this effect. The K-means is a simple clustering algorithm that has been adapted to many problem domains.

2.3.2 Self Organizing Map

The *self organizing map* (SOM) [Kohonen, 1990] is a neural network that undergoes unsupervised learning (i.e., identifying classes in the absence of prior knowledge) and apart from nonlinear classification it is useful in projecting/visualizing high-dimensional data on to a low dimensional (i.e., 2-D or 3-D) space. The SOM possesses several attractive properties [Vasanto, 2000]: (i) it performs an ordered dimensionality-reducing mapping of the training data, (ii) the created map follows the probability density function of the data and is also robust to missing data, and (iii) the map is readily explainable, simple and easy for visualization. SOM has been successfully applied in various engineering applications [Kohonen et al., 1998] involving pattern recognition, image analysis, exploratory data analysis [Ultsch, 1993; Mao et al., 1995], process monitoring and control [Simula et al., 1995, Tryba et al., 1991] and fault diagnosis [Chan et al., 1999; Kang et al., 1999]. The other important application of SOM namely, classification (clustering), has been exemplified in a number of recent studies in genomics [Schneider, 1999; Wang et al., 2001; et al., 2002; Kasturi et al., 2003].

The SOM network architecture, as shown in Figure 2.13, consists of a two-dimensional array of units each of which is connected to all the p input nodes. It is also possible to use a grid of higher dimensions although such a grid is difficult to visualize conveniently. The SOM neural network architecture and its training method possess following properties: (i) an array of neurons, which as a function of its input of arbitrary dimensionality, calculates the outputs using a simple output function, (ii) a criteria to determine the "winner" neuron possessing the largest output, and (iii) an adaptive rule for updating the weights of the chosen neuron and its neighbors. The SOM training algorithm proposed by Kohonen [1990] has a very desirable property of topology preserving.

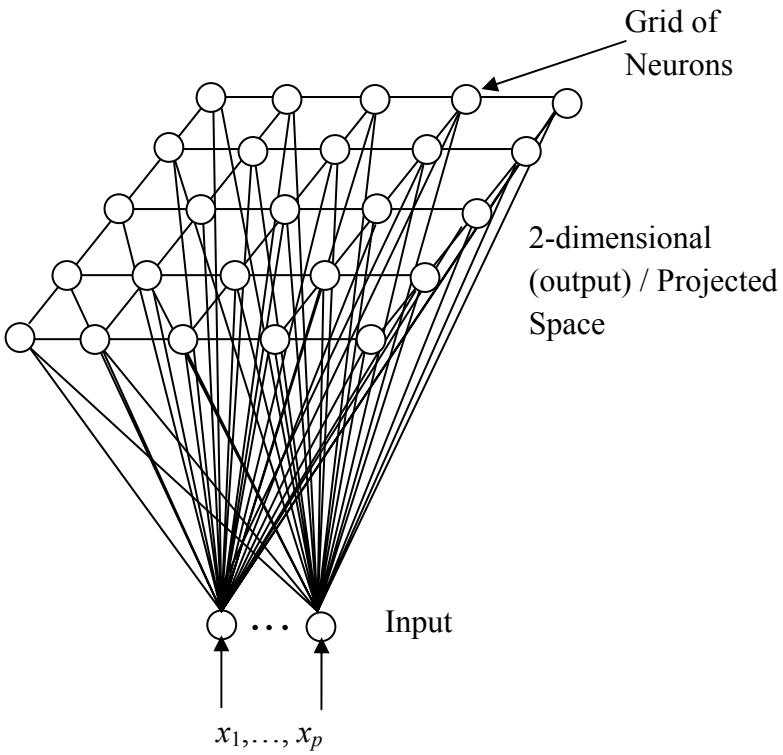

Figure 2.13: Schematic of Self-organizing Map

A. SOM training algorithm

Let $\mathbf{x}_k$, $k = 1,2,\ldots,N$ be the p-dimensional patterns (vectors) and $\boldsymbol{w}_{ij}$ be the p-dimensional weight vector associated with the processing element at the location (i, j) of the 2-D array. The stepwise procedure for training the SOM network is given below.

Step 1 (*Initialization*): Choose small random values for the initial weights, $w_{ij}(0)$, and fix the initial learning rate ($\hat{\alpha}_O$) and the neighbourhood.

Step 2 (*Determining the winner*): Select a sample pattern, $\mathbf{x}_k$, from the data set and determine the winner neuron (C_i, C_j) at time (iteration) t, using the minimum-distance Euclidean criterion.

$$\left\| \mathbf{x}_k - \boldsymbol{w}_{C_i C_j} \right\| = \min_{i,j} \left\| \mathbf{x}_k - \boldsymbol{w}_{i,j} \right\|; \; i = 1, 2,\ldots., L; \quad j = 1, 2,\ldots, L \tag{2.47}$$

where ||.|| refers to the Euclidean norm and L denotes the number of rows (as also columns) in the square 2D array.

Step 3 (*Weight updation*): Update all the weights according to the kernel-based learning rule;

$$\begin{aligned} \boldsymbol{w}_{ij}(t+1) &= \boldsymbol{w}_{ij}(t) + \hat{\alpha}(t) \| \mathbf{x}_k(t) - \boldsymbol{w}_{ij}(t) \| \quad \text{if } (i,j) \in N_{C_iC_j}(t) \\ &= \boldsymbol{w}_{ij}(t) \qquad\qquad\qquad\qquad \text{otherwise} \end{aligned} \tag{2.48}$$

where t denotes training iteration index; $N_{C_iC_j}(t)$ is the neighbourhood function of the winner unit (C_i, C_j) at iteration t, and $\hat{\alpha}(t) = \hat{\alpha}_0/(1+t)$ is the learning rate.

Step 4: Decrease the value of the learning rate, $\hat{\alpha}(t)$, by incrementing the iteration index, t, by unity and shrink the neighbourhood, $N_{C_iC_j}(t)$.

Step 5: Repeat steps 2 – 4 until the change in the weight values is less than the specified threshold, or the maximum number of iterations ($\hat{t}_{max}$) is reached.

It should be emphasized that the success of SOM training depends critically on the judicious selection of the main algorithm-specific parameters (i.e., $\hat{\alpha}(t)$ and $N_{CiCj}(t)$), initial values of the weight vectors, and the number of pre-specified training iterations, $\hat{t}_{max}$). These are commonly optimized using a heuristic procedure.

2.4 OPTIMIZATION FORMALISMS

The search for optimal and near-optimal solutions is an important problem in different areas of human activities including engineering, technology, manufacturing, business and finance. In chemical engineering/technology, the goal of process optimization is to obtain optimal process operating conditions, which lead to improved process performance, e.g., maximization of conversion, selectivity, etc., or minimization of reactor temperature, selectivity of an undesired product, operating cost, etc. Broadly, there are two different classes of optimization methods: *deterministic* and *stochastic*. Deterministic methods aim to

arrive at the optimum by approximating the local neighborhood of a given solution in the search space and moving to a better solution whenever possible. All gradient based methods and some line-search methods fall under this class. Gradient-based methods encompass math-programming formulations including linear, non-linear and discrete optimization models and their associated solution strategies. Heuristic methods such as tabu-search that allow for non-improving moves have been used with a great success, not so much as models in themselves but as alternative solution strategies under a deterministic framework. Enumerative methods that involve listing the entire solution space or a relevant neighborhood also fall under the category of the deterministic methods. These methods typically use *a priori* information about the solution space based on the past experience. The point of departure of stochastic methods from deterministic ones is that the former contain a random component in them. Here, solutions may be manipulated at random and the emphasis is on sampling the search space as widely as possible while at the same time trying to locate promising regions for further exploration. Simulated annealing, simultaneous perturbation stochastic approximation, random search and genetic algorithms fall under this broad class of methods.

Both deterministic and stochastic approaches [Vaidyanathan and El-Halwagi, 1994; Adjiman et al., 2000, Jayaraman et al., 2000, Yu et al., 2000] have been developed to address optimization problems in chemical engineering. Although a few deterministic approaches guarantee the global optimality of the final solution, they require specific formulations. On the other hand, stochastic algorithms cannot guarantee global optimality, but they can be readily and easily applied to many optimization problems. With appropriate parameters, they have a high probability of locating the globally optimal solution. Most of these existing stochastic approaches are only suitable for solving small to medium scale problems [Pörn et al., 1999].

Conventionally, gradient-based deterministic methods are used for process optimization. Most of these methods require that the objective function (to be minimized or maximized) should be smooth, differentiable and continuous. Many commonly used deterministic optimization formalism do not satisfy these criteria and therefore alternative optimization techniques need to be employed. AI-based

stochastic optimization techniques such as genetic algorithms and tabu search, overcome the above-stated drawbacks, and therefore are ideal for the optimization of chemical processes.

2.4.1 Simultaneous Perturbation Stochastic Approximation

Multivariate stochastic optimization plays a major role in the analysis and control of many real-world systems. In almost all large-scale practical optimization problems, it is necessary to use a mathematical algorithm that iteratively seeks out the solution because an analytical (closed-form) solution is rarely available. In the above spirit, the "s*imultaneous perturbation stochastic approximation* (SPSA)" method [Spall, 1987, 1998a,b] has been developed for attacking difficult multivariate optimization problems. The SPSA has recently attracted considerable attention in areas such as statistical parameter estimation, feedback control, simulation-based optimization, signal and image processing, and experimental design. The essential feature of SPSA–which accounts for its power and relative ease of implementation–is the underlying gradient approximation that requires only two measurements of the objective function regardless of the dimension of the optimization problem. This feature allows for a significant decrease in the cost of optimization, especially in problems with a large number of variables to be optimized.

The SPSA optimization methodology differs from the commonly employed deterministic gradient based techniques in the following aspects. Instead of directly evaluating the gradient with respect to each decision variable by perturbing it separately (as done in the standard two-sided finite difference approximation), the SPSA methodology approximates the gradient by perturbing all the decision variables simultaneously. Thus, irrespective of the number (K) of decision variables to be optimized, only two objective function measurements are necessary for the gradient approximation; this is in contrast to the finite-difference approximation, where $2K$ function measurements are necessary for the gradient evaluation. The implementation procedure of SPSA is an iterative that begins with a randomly initialized (guess) solution vector, $\hat{x}$. The SPSA technique stipulates the cost function, $C_{yr}(x)$, to be differentiable, since it searches for the minimum

point, x^*, at which the gradient of the objective function, g(x^*), attains zero magnitude.

$$g(x^*) = \left| \frac{\partial C_{yr}(x)}{\partial x} \right|_{x=x^*} = 0 \tag{2.49}$$

That is, at each SPSA iteration, the gradient is approximated by utilizing the numerically efficient simultaneous perturbation technique alluded to earlier. With these preliminaries, the stepwise procedure for SPSA implementation can be given as [Nandi et al., 2004]:

***Step* 1.** Set the iteration index, t to zero and choose randomly a K-dimensional guess solution vector, $\hat{x}_t|_{t=0}$

***Step* 2.** Compute the t-dependent values, A_t and Z_t, termed "gain sequences" using,

$$A_t = \frac{A}{(r+t+1)^{\eta}} \tag{2.50}$$

$$Z_t = \frac{Z}{(t+1)^{\beta}}, \tag{2.51}$$

where constants, A, Z, r, η and β assume nonnegative values. The optimal values of η and β are either 0.602 and 0.101 or 1.0 and 0.1667, respectively [Spall, 1998a].

***Step* 3.** Generate a K-dimensional perturbation vector, Δ_t, using Bernoulli ±1 distribution, where probability of occurrence of either + 1 or - 1 is 0.5; next, perturb all the K elements of the vector $\hat{x}_t$ simultaneously, as given by

$$\hat{x}_t^+ = \hat{x}_t + Z_t \Delta_t \tag{2.52}$$

$$\hat{x}_t^- = \hat{x}_t - Z_t \Delta_t \tag{2.53}$$

***Step* 4.** Using $\hat{x}_t^+$ and $\hat{x}_t^-$ as arguments, compute two measurements, that is, $C_{yr}(\hat{x}_t^+)$ and $C_{yr}(\hat{x}_t^-)$, of the objective function defined in Eq. 2.49.

***Step* 5.** Generate the simultaneous perturbation approximation of the unknown gradient, $\hat{g}_t(\hat{x}_t)$, using

$$\hat{g}_t(\hat{x}_t) = \left[\frac{C_{yr}(\hat{x}_t^+) - C_{yr}(\hat{x}_t^-)}{2Z_t}\right] \times \left[\Delta_{t1}^{-1} - \Delta_{t2}^{-1}, \ldots, \Delta_{tK}^{-1}\right]^T \quad (2.54)$$

where $\hat{g}_t(\hat{x}_t)$ is *k*-dimensional and Δ_{tK} refers to the k^{th} element (+1 or -1) of the perturbation vector, Δ_t .

***Step* 6.** Update estimate of the decision vector according to

$$\hat{x}_{t+1} = \hat{x}_t - A_t \hat{g}_t(\hat{x}_t) \quad (2.55)$$

Step **7.** Increment the iteration counter *t* to *t* + 1 ($1 \leq t \leq t_{max}$) and repeat steps 2-6 until convergence; the criterion for convergence could be that in successive iterations the decision variable values exhibit very little or no change.

2.4.2 Tabu Search

Tabu Search (TS) is a memory-based stochastic optimization strategy [Glover, 1986]. By considering historical information during the search process, TS has been reported to have a more flexible and effective search behaviour than other stochastic methods [Glover, 1986].

TS is a meta-heuristic approach that guides a neighborhood search procedure to explore the solution space in a way that facilitates escaping from local optima. Figure 2.14 shows a schematic of the algorithm. TS starts from an initial randomly generated solution. A set of neighbor solutions, $N(x)$, are constructed by modifying the current solution, x. The best one among them, x, is selected as the new starting point, and the next iteration begins. Memory, implemented with tabu lists, is employed to escape from locally optimal solutions and to prevent cycling. At each iteration, the tabu lists are updated to keep track of the search process. This memory allows the algorithm to adapt to the current status of the search, so as to ensure that the entire search space is adequately explored and to recognize when the search has got stuck in a local region. Intensification strategies are employed to search promising areas more thoroughly, while at the same time diversification strategies are employed to broadly search the entire feasible region, thus helping to avoid becoming stuck in local optima.

Finally, aspiration criteria is employed to override the tabu lists in certain cases. The details of each of the elements will be discussed in the following sub-sections.

A. Neighborhood search

TS explores the search space of feasible solutions by a sequence of moves [Glover and Laguna, 1997]. A move is an operation that changes the current solution to another solution. TS starts with an initial solution, say A_0. The initial objective function value, $f(x_1, x_2,\ldots,x_N)$, is initialized. A random change that is the product of a random number (between −1 and 1) and the magnitude of the search space along a dimension, say x_1, is added to the current solution. To ensure that the move does not cause the point to lie outside of the specified bounds for this dimension, a check is performed. If the move would exceed the bounds, the move is altered, so that it either is on the bound or remains at the location of the current solution. A similar random change for all remaining dimensions is then added. In this way, a neighbor solution, A_1, is determined. Similarly, a set of other neighbor solutions, A_2 to A_M, can be generated.

B. Tabu lists

When the best neighbor is not better than the current solution, it is classified as "tabu" and added to a recency-based tabu list. The tabu property remains active throughout a time period, called the "tabu tenure". As new solutions are added to the tabu list, older solutions will be released from the bottom. Thus, the recency-based tabu list stores the most recently visited solutions and forbids revisiting unpromising solutions for a specified number of iterations. In continuous solution space, the probability of visiting the same solution twice is very small. Although tabu lists only record specific solutions, the areas surrounding each of them are classified as tabu. The tabu area is empirically classified as 20% of the search range (centered at the current solution) along each dimension. The recency-based tabu list records solutions for a short time period; it is often called the short-term memory. Long-term memory is dependent on the frequency that a solution has been visited. The areas cover more solutions than other parts of the search space will be added to the frequency-based tabu list, which records solutions that have been searched around most often. The location of these frequency-based tabu regions are defined by the best neighbor solutions as they are determined. Once the maximum number of elements in the frequency-

based tabu list has been attained, the solution with the smallest frequency index will be replaced.

C. Aspiration criterion

In some cases, the best neighbor solution may appear in a tabu area. If so, the aspiration criterion will be checked to determine whether the best neighbor should be accepted despite being in a tabu area. When the best neighbor solution is at the some point, which is not as good as the best solution so far, it cannot be accepted as the starting point for the next iteration. Instead, the best non-tabu neighbor is used. The appropriate use of such criteria can be very important for TS to achieve its best performance.

Lin et. al. [2004], have designed an additional aspiration criterion which achieves a balance between intensified and diversified search. The aspiration criterion is based on a sigmoid function:

$$s(k) = \frac{1}{1 + e^{-\sigma(k - k_{center} \times M)}} \tag{2.56}$$

where k is the current iteration number and k_{center} determines at which point, $s(k) = 0.5$. A random number, $0 \leq P \leq 1$, is generated from a uniform distribution at each iteration. If P is greater than $s(k)$, the tabu property is active and the best non-tabu neighbor is used as a new starting point; if P is less than or equal to $s(k)$, the aspiration criterion overrides the tabu property. Thus, a restart operation (resulting from a frequency-based tabu list) will be canceled, or a neighbor solution which would otherwise be tabu (because of a recency-based tabu list) will be used to generate the new neighbor solutions.

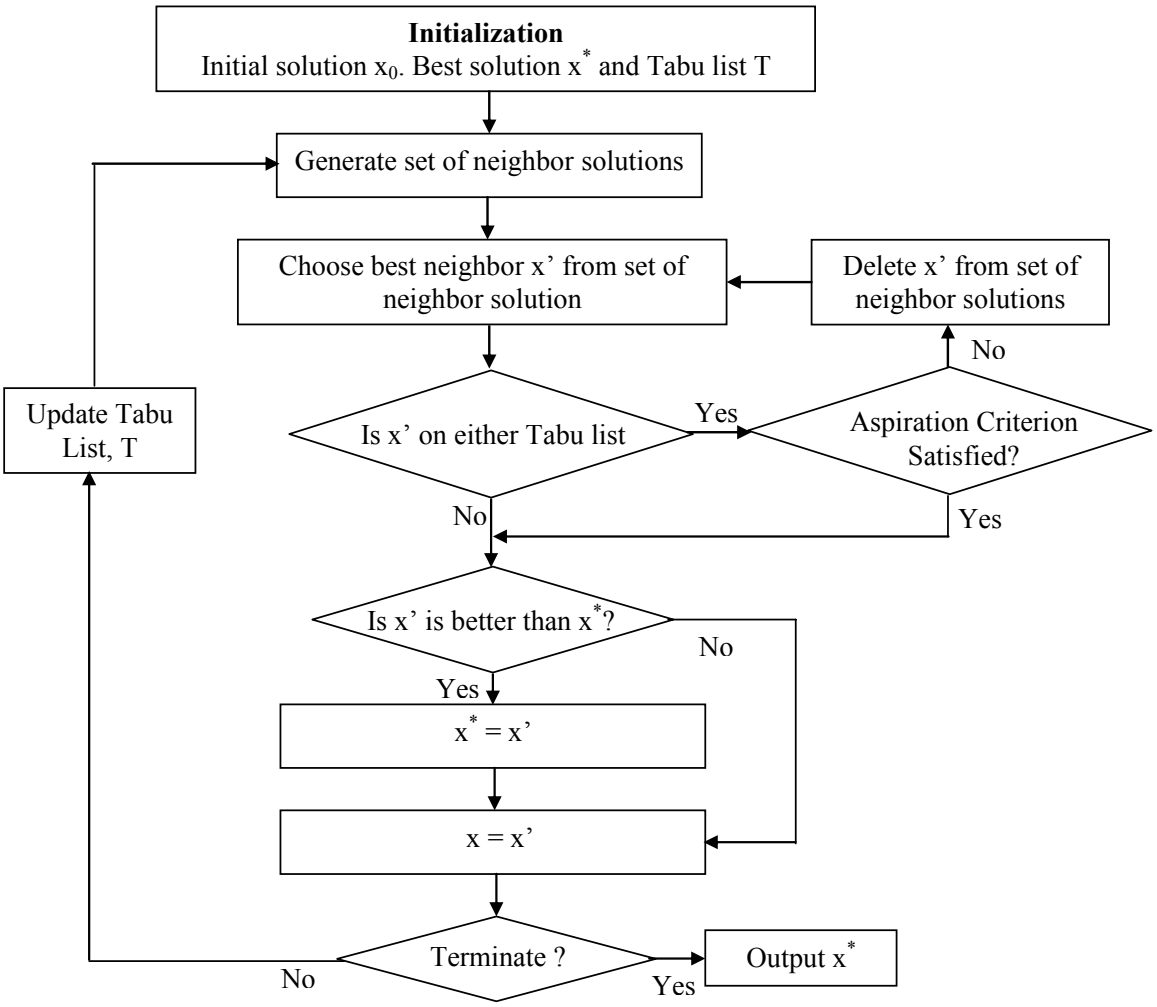

Figure 2.14: Flow chart of TS algorithm

D. Termination criterion

Because TS is a stochastic optimization approach, global optimality of the final solution cannot be guaranteed. In general, the longer the search process, the higher the probability of finding good solutions. The maximum time termination criteria has been widely used due to the ease of implementation [Patel et al., 1991; Tayal and Fu, 1999; Wang et al., 1999]. In most cases, this criterion is considered effective and practical; however, it risks wasting time in an unproductive search. Two primary conditions may cause this behavior. First, a solution very close to the global optimum may have already been located. Second, the algorithm may

have become trapped in a local solution. Thus, termination-on-convergence criteria have been implemented so that the search ceases at an "appropriate" time [Jain et al., 2001].

If the improvement over Γ generations is no larger than a threshold, δ, continued iterations are considered to be ineffective, and the search should be stopped. The termination criterion is mathematically expressed as follows:

$$\left| \frac{f_k(x) - f_{k-\Gamma}}{f_{k-\Gamma}(x)} \right| < \delta \tag{2.57}$$

where $\Gamma = \eta M$, δ is the ratio of the change in the value of the objective function, and η is the fraction of the maximum iterations possible over which the change in the objective function is compared.

2.4.3 Genetic Algorithms

Genetic a1gorithms (GAs) [Holland 1975; Goldberg, 1989] are artificial intelligence based stochastic methods which enforce the "survival of the fittest" paradigm of evolution along with the genetic propagation of characteristics principle followed by biological species. This brings to bear a balanced tradeoff between exploitation and exploration. Unlike traditional methods which move from point to point, an initial population of solutions constantly refined in a manner imitating selection and biological evolution, while discovering expectedly better solutions. GAs have been used with a great success in solving problems involving very large search spaces [Goldberg, 1989]. Owing to there attractive features, GAs are being increasingly used for solving diverse optimization problems in chemical engineering and technology [Nandi et al., 2002, Garcia et al., 1998]. The process of evolution involves the survival of the most adapted organism to the environment and the propagation of its characteristics to the next generation by reproduction. This phenomenon is referred to as "natural selection" and is primarily responsible for the evolution and modification of different species to suit the environment. The propagation of characteristics is brought forth by information carried in the genetic material of the species. In addition, gene-level operations such as recombination and mutation that occur during reproduction

lead to combinatorial diverse characteristics in the subsequent generations. By this, we imply the discovery of new characteristics that arise as a combination of different ones present in different individuals in the population. Recombination involves the exchange of parts of two parent chromosomes producing two different combinations. Mutation involves random changes at different loci in the gene, leading to inclusion of characteristics not present in the parent(s). The combined effects of *selection*, *recombination* and *mutation* not only produce fitter individuals but also lead to diversification of the population characteristics. This diversification leads to a state of perpetual novelty during evolution, where the evolving species is able to combat the changing environment successfully. One can immediately see how such a framework would be useful from an optimization viewpoint. If one were to treat the environment as an objective to be optimized and the individual in the environment as a point in the search space, evolution would become analogous to searching through a landscape of solutions.

In nature, the complete representation of any organism lies in its genetic code, which is a sequence of alphabets of amino acids. In a similar fashion, GAs model potential solutions as bit strings of alphabets that completely represent all the characteristics of the solution relevant to the optimization or search. The scheme of encrypting individuals as sequences of alphabets is referred to as the representation or "encoding" scheme. Different techniques are utilized for encoding a candidate solution as *binary*, *gray* and *real*. The decoded value of the solution is just the coordinate of the solution in the domain of the objective function. In nature, the fitness of an individual is its fitness or competitive edge in the environment. In the same way, in a GA the fitness of a solution is indicative of the value of the objective function, when it is evaluated at the decoded coordinate of the solution. In GA, natural selection occurs by choosing solutions with a probability proportional to their relative fitness values by some scheme. Recombination and mutation are performed on the encoded representation of the selected solutions in a manner analogous to chromosomal crossover and mutation. This process of selection and genetic operation is iterated until termination criterions are met. In the following, principle components and operators of a typical genetic algorithm is described.

A. Representation

The most important component in the GA procedure is the representation scheme for coding candidate solutions to a given optimization problem. All the solutions need to be represented in some form of genetic code. The most common encoding is a string representation where each solution is represented as a one-dimensional string of numbers or alphabets while other variations are also possible [Venkatasubramanian, 1994]. In a string representation, each coordinate or element of the string typically represents a component or piece of the candidate solution. Most problems in the GA literature use the binary encoding scheme where each loci of the string is drawn from a binary alphabet of "0" or "1". For function optimization where the solution space contains real numbers, the decoding scheme is generally binary to decimal conversion followed by mapping the resulting number into an appropriate domain. Once again, the mapping could be linear or exponential depending on scaling. For example, a binary string of the form (a_0, a_1, a_2,…,a_{N-1}) of length N is converted to its equivalent real value in its domain as follows.

$$D = \sum_{i=0}^{N-1} a_i \times 2^i \tag{2.58}$$

For instance, for $N = 4$ and a binary string (0 1 0 1), the solution is $D = 5$.

For multivariate domains, the complete binary representation of a point in the domain is obtained by placing the binary encoded string for each of coordinates/variables end to end in some predefined order. For example, a three dimensional coordinate (x_1, x_2 x_3) with $N = 3$ for each coordinate could be represented (1 0 1 1 1 0 0 0 1) where the first three bits encode x_1, the next three bits encode x_2, and the last three encode x_3. One of the drawbacks of the binary representation is that the number of bits used scales with the number of variables and precision of each variable. A floating point representation overcomes this by representing each variable in its negative real coded form. For function optimization problems, this gives a transparent and compact representation and can be utilized with special operator to handle constraints, [Michalewicz, 1995]. However, the drawback here is that specialized operators need to develop for the GA to exploit the solution space effectively.

B. Initialization

Initialization refers to the generation of the initial population of candidate solutions as well as the choice of some parameters of the population, such as its size. The preferred characteristics of an initial population are diversity and reasonable levels of fitness values. However, in practice, depending upon the application, generating an initial population varies from a random generation to careful choosing of candidates based on user's experience. Sometimes a few distinct and diverse solutions are chosen and assigning copies based on their fitness values to provide a good string population. In choosing the population size, a large population is normally preferred. But this leads to a large computation burden. Syswerda [1989] performed experiments with different control parameters including population size for the scheduling problem. The essential result is that an optimal choice of the population size tends to depend upon the nature of the domain, the representation, the evaluation and the genetic operators used. A re-initialization procedure was used by Goldberg [1989] wherein the GA was restarted every few generations with a new population containing the best solutions found so far and the remaining, if any, generated randomly. A variation of the fixed population size GA is the GA with varying population size (GAVaPS) [Arabas, 1994]. In this algorithm, the population is continuously augmented by the newly created products of recombination. However, the algorithm has a measure of age or lifetime of an individual beyond which the individual "dies' or is removed from the population. This lifetime, instead of the selection probability, is set proportional to the fitness of the individual. This means that fitter individuals live longer than the rest and the population is controlled by the death rate of the individuals.

C. Fitness evaluation

Once a population of candidate solutions has been created, it needs to be evaluated to determine its fitness in the environment. For an optimization problem, the environment is the objective function. Depending on how low (for minimization problems) or how high (for maximization problems) the objective function value for an individual is, its fitness should have a proportionally low or high value. The fitness evaluation procedure normally includes in itself the decoding scheme to map the individual from its stream based representation to its

domain of definition. In cases where the characteristics or a property of interest of the optimum is known, the fitness could be a scaled distance of the individual's property from that of the optimum. In some problems, one does not have a single objective but several to be optimized simultaneously as well as constraints to be satisfied by the solutions. One way of handling multiple objectives is to define a new objective function that is a "*weighted*" sum of all the objectives. Here, the choice of the weights can reflect the relative importance of optimizing different objectives. To handle constraints in a genetic algorithm, the objective function is usually augmented with a penalty term that "weighs" in the feasibility of the solution, i.e., if it satisfies all the imposed constrains. This is similar to a Lagrangian multiplier often used in nonlinear programming.

D. Selection

After all the candidates have been evaluated for their fitness values, the next step in the GA is the selection procedure. This involves algorithmic imposition of natural selection which enforces the survival of the fittest. The selection scheme has to make sure that the fitter individuals in the population are allotted more opportunities to reproduce and recombine to produce offspring. To this end, two different schemes are normally used.

a. Roulette Wheel (RW) Selection: In this scheme, once the fitness evaluation is completed, the population is sorted (ranked in the ascending order of fitness scores) and a running sum of the fitness is calculated for each member starting from the first one in the sorted list. The running fitnesses are normalized using the cumulative fitness of the entire population. A random number between 0 and 1 is drawn. The first member of the sorted list (beginning with the member with the lowest fitness) whose cumulative fitness is greater than the random number is selected. The RW selection scheme with a variable probability of selection across generations can also be used [Back, 1993].

b. Tournament Selection: In this scheme, a specified number, called the "tournament size", of members are chosen from the parent population and these enter competition for selection. The winner is decided based on the best fitness and allowed to enter the reproductive phase. This process is repeated sufficiently, along with recombination and mutation, to produce the offspring

population. This method slightly offsets the effects of a few large fitness solutions in the population biasing the selection scheme towards above-average solutions in general [Goldberg, 1989]. As opposed to the RW selection procedure, this is a static selection scheme where the probability of selection of a candidate remains fairly constant across generations.

E. Reproduction

Once an individual candidate solution has been selected from the current population, three basic genetic operators (*direct reproduction*, *crossover* and *mutation*) may be applied. The direct reproduction operator directly copies a member from the parent population to the next generation. In order to ensure that the survival of the fittest principle performs well, a member amongst the better fitness values is selected using the selection operator and is copied to some member having an inferior fitness value. In the process, the member having an inferior fitness value dies in order to make place for the member having a better fitness.

F. Crossover

Chromosomal crossover refers to the random recombination of parts of two chromosomes (the parents) to produce two new chromosomes (the offspring). This is a large-scale operator in the sense that it significantly perturbs the genotype of the parents. From an optimization viewpoint, the recombination operator tends to improve the combinatorial diversity by using the building blocks present in the population. In this manner, novel combinations of existing parts are discovered that could potentially lead to fitter individuals. The simplest abstraction of chromosomal crossover in genetic algorithms is the one point crossover. Here, a random cut-point is chosen along the length of the coded solution and the two parent chromosomes are split at this point. The tail portion (i.e., all the bit positions following the cut-point) of the two parents are exchanged to create two offspring chromosomes. Consider a string representation in which a "1" at the first position and a "0" at the last position is necessary for good solution quality. Consider a string of length L which possesses this feature. One-point crossover selects the cut-point with uniform probability. This implies that the probability that this feature will be lost is 1.0 when the other parent in question does not posses a "1" in the first position or a "0" in the last. Strings with certain

positions fixed at certain values (0s or 1s in the binary case) and the rest arbitrarily assigned are called *schemas* and these form the basic building blocks that the GA manipulations. The order of the schema refers to the distance or length (in bits) between fixed positions in the schema. It is clear from the above example that one-point crossover is biased toward lower-order schema, i.e., when the fixed positions are closer together. This implies that beneficial characteristics brought forth by high order schema are lost more regularly. This characteristic of one-point crossover is called positional bias [Eshelman et al., 1989]. To overcome this drawback, variations of crossover exist in the literature [Syswerda, 1989] and two of the common ones are *two-point* crossover and *uniform* crossover. In two-point crossover, two random cut-points are chosen and the portions of the encoded representations of the parents between these cut-points are mutually exchanged. A uniform crossover is the generalized form of crossover where chromosomal exchanges happen between parents, across multiple (the number chosen randomly) cut-points. The recombination operator has a probability associated with it which dictates how often it is used. The probability of crossover is typically set to a high value (around 99%) for binary coded representation. A random number is drawn and whenever it falls below the crossover probability, two individuals (selected using one of the selection schemes described in the previous section) are allowed to undergo crossover. If the random number test fails, the chosen individuals are duplicated and placed in the offspring population.

G. Mutation

The recombination operator is handicapped by the fact that it combines only what is already present in the population. For instance, let us suppose that all individuals in the initial population contain zero in the second bit position of their genotype and that any reasonably good genotype should have the value of "1" at that position. Applying recombination alone on this population will not lead to any genotype with "1" in the second position (assuming we use the above-mentioned crossover operators). This would eventually lead to convergence of the algorithm to a poor local optimum. Hence, to be effective, the GA needs an influx of characteristics extraneous to the population. This is provided by the mutation operator.

Mutation is applied by randomly flipping the bits from zero to one or vice versa with a certain probability. For a simple GA using binary encoding, mutation is normally applied after crossover and with a low probability (around 1%). This is since with high probabilities mutation tends to destroy the good features (schema) brought forth by recombination and selection. Mutation is implemented by drawing out successive random numbers (corresponding to each bit) and flipping the bit whenever the number falls below the mutation probability. However, this method offers uniform mutation where there is no distinction between the positions of the bits. When applying GAs to optimization in the domain of real numbers coded in binary format, non-uniform mutation may sometimes be preferred. Under non-uniform mutation, the more significant bits of the solution (that cause large changes in the decoded value of the solution) are mutated at different probabilities compared to the less significant ones.

This section describes the implementation and properties of recombination and mutation operators. However, the nature and probabilities of the operators discussed above is generally valid only for a simple genetic algorithm with binary representation. A lot of research has been undertaken to find optimal settings for operator probabilities [Grefenstette, 1986] including different adaptive schemes [Fogarty, 1989; Srinivas and Patnaik, 1994]. Evolutionary strategies typically use several operators and in competition with one another, as opposed to the use of mutation after crossover in classical GAs. They may also contain modules to tune the operator probabilities dynamically [Davis, 1989; Schwefel, 1984]. The nature of these specialized operators is domain dependent and dependent also on the representation used. For instance, constrained optimization with real coded representation uses specialized operators to ensure that the generated offspring satisfy the imposed constraints. Other specialized operators have also been constructed depending on the application domain [Syswerda, 1989]. When appropriate knowledge is available, heuristic hill-climbing and gradient based operators have been used in conjunction with purely stochastic ones [Wright, 1991; Bhandari, 1994].

H. Pros and cons of genetic algorithms

In contrast to more traditional numerical techniques, which iteratively refine a single solution vector as they search for optima in a multi-dimensional

landscape, genetic algorithms operate on entire populations of candidate solutions in parallel. In fact, the parallel nature of a GA's stochastic search is one of the main strengths of the genetic approach. This parallel nature implies that GAs are much more likely to locate a global peak than traditional techniques, because they are much less likely to get stuck at local optima. In addition, due to the parallel nature of the stochastic search, the performance is much less sensitive to initial conditions, and hence and GA's convergence time is rather predictable. In fact, the problem of finding a local optimum is greatly minimized because GAs, in effect, make hundreds, or even thousands, of initial guesses. This implies that a GA's performance is at least as good as a purely random search. In fact, by simply seeding an initial population and stopping there, a GA without any evolutionary progression is essentially a Monte Carlo simulation.

As appealing as a GA may seem, the parallel nature of the stochastic search is not without consequences. Although the prospects of finding global optima make it robust, the convergence of a GA is usually slower than traditional optimization techniques. In fact, with a good initial guess close to the global optimum, a numerical technique will likely be much faster, and more accurate, than a genetic search because, in essence, the GA will be wasting time testing the fitness of sub-optimal solutions. Furthermore, due to the stochastic nature of GAs, the solution, although more likely to estimate the global optimum, will only be an estimate. It must be realized that GAs will only by chance find an exact optimum, whereas traditional gradient methods will find it exactly, assuming, of course, they find it at all. The user must then determine whether the solution found by a GA is close enough. In many cases, it will be, but the question of 'How close is close enough?' is somewhat arbitrary and application-dependent.

I. Real coded genetic algorithm

Binary coded strings for the representation solution have dominated GA research since there are theoretical results that show them to be the most appropriate ones [Goldberg, 1991a], and as they are amenable to simple implementation. However, the GA's good properties do not stem from the use of a bit string [Antonisse, 1989]. For this reason, the path has been laid toward the use of non-binary representations more adequate for each particular application problem. One of the most important representation is *real number* representations,

which would seem particularly natural when optimization problems with variables in continuous search space are tackled. Therefore, a chromosome is a vector of floating point numbers whose size is kept the same as the length of the vector, which is the solution to the problem. GAs based on the real number representation is called *real-coded* GAs (RCGAs). The use of real coding initially appears in specific applications, such as in [Lucasius et al., 1989] for chemometric problems, and in Davis [1989] for the use of meta operators in order to find the most adequate parameters for a standard GA. Subsequently, RCGAs have been mainly used for numerical optimization on continuous domains [Write, 1991; Michalewicz, 1992].

The use of real parameters makes it possible to use large domains (even unknown domains) for the decision variables, which is difficult to achieve in binary implementations where increasing the domain would mean sacrificing precision, assuming a fixed length for the chromosomes. Also, when using real parameters, the capacity to exploit the *graduality* of the function with continuous variables, where the concept of graduality refers to the fact that, slight changes in the objective the variables correspond to a slight change in the objective function. Using real coding the representation of the solutions is very close to the natural formulation of many problems. Most real-world problems may not be handled using binary representations and an operator set consisting only of binary crossover and binary mutation [Davis, 1989]. The reason is that nearly every real-world domain has associated domain knowledge that is of use when one is considering a transformation of a solution in the domain. Davis [1989] believes that the real-world knowledge should be incorporated into the GA, by adding it to the decoding process or expanding the operator set. Real coding allows the domain knowledge to be easily integrated into the RCGA for the case of problem with non-trivial restrictions.

Since the basic string of GA is coded in real numbers in RCGAs, crossover and mutation operators need to be reformulated. Few of them are listed below.

J. Crossover operators for RCGAs

Let us assume that $C_1 = (c_{1,1},c_{2,1},\ldots,c_{n,1})$ and $C_2 = (c_{1,2},c_{2,2},\ldots,c_{n,2})$ are two chromosomes that have been selected to apply the crossover operator.

(i) Simple crossover [Write, 1991; Michalewicz, 1992]**:**

A position $i \in \{1,2,...,n-1\}$ is randomly chosen and the two new chromosomes (offsprings) are build as shown below.

$$C_1^{\text{new}} = (c_{1,1},c_{2,1},\ldots,c_{i,1},\, c_{i+1,2,\ldots,}\, c_{n,2}) \tag{2.59}$$

$$C_2^{\text{new}} = (c_{1,2},c_{2,2},\ldots,\, c_{i,2,}\, c_{i+1,1,\ldots,}c_{n,1}) \tag{2.60}$$

(ii) Arithmetic crossover [Michalewicz, 1992]**:**

Two offspring, $C_k^{\text{new}} = (c_{1,k},\ldots,\, c_{i,k,\ldots,}c_{n,k})$ $k = 1,2$, are generated, where $c_{i,1} = \lambda\, c_{i,1} + (1 - \lambda)\, c_{i,2}$ and $c_{i,2} = \lambda\, c_{i,2} + (1 - \lambda)\, c_{i,1}$. λ is a constant (uniform arithmetical crossover) or varies with regard to the number of generations made (non-uniform arithmetic crossover).

K. Mutation operators for RCGAs

Let us suppose $C = (c_1, c_2,\ldots, c_n)$ is a chromosome and $c_i \in [a_i, b_i]$ a gene to be mutated. Next, the gene, c_i^{new}, resulting from the application of different mutation operator is shown below.

(i) Random mutation [Michalewicz, 1992]**:**

c_i^{new} is a random (uniform) number from the domain $[a_i, b_i]$.

(ii) Non-uniform mutation [Michalewicz, 1992]**:**

If this operator is applied in a generation t, and g_{max} is the maximum number of generations then

$$c_i^{\text{new}} = \begin{cases} c_i + \Delta(t, b_i - c_i) & \text{if } \tau = 0 \\ c_i - \Delta(t, a_i - a_i) & \text{if } \tau = 1 \end{cases} \tag{2.61}$$

with τ being a random number which may have a value of zero or one, and

$$\Delta(t, y) = y\left(1 - r^{\left(1 - \frac{t}{g_{\max}}\right)^b}\right), \tag{2.62}$$

where r is a random number from the interval [0,1] and b is a parameter chosen by the user, which determines the degree of dependency on the number of iterations. This function gives a value in the range [0, y] such that the probability of returning a number close to zero increases as the algorithm advances. The size of the gene generation be lower with the passing of generations. This property causes this operator to make a uniform search in the initial space when t is small, and very locally at a later stage, favoring local tuning.

In short, the principle features possessed by GAs are as follows:

(i) They require only scalar values and not the second and/or first order derivatives of the objective functions, thus they are not numerically expensive;
(ii) GAs are capable of handling nonlinear and noisy objective functions;
(iii) GAs perform global searches and thus are more likely to arrive at or near the global optimum;
(iv) They do not impose preconditions, such as smoothness, differentiability and continuity on the form of the objective functions.

2.4.4 Memetic Algorithm

Although used widely, the principle drawback of GAs is that since the formalism performs a global search of the solution space, it can take a long time to converge even if the optimal solution lies in the neighborhood of a candidate solution. A genetic algorithm related methodology that overcomes the stated problem is known as "*Memetic algorithms* (MAs)" [Moscato, 1992, 1999]. The most attractive feature of MA, which makes it efficient nonlinear optimization formalism, is that it combines a local search heuristics with the population-based global search conducted by the GAs. This feature helps in significantly speeding up the convergence to the optimal solution.

The MA likewise, GA based on the concept of evolution. However, whereas the GA models the biological evolution, the MA models the cultural

evolution or the evolution of ideas [Moscato, 1999]. The principle difference between this model and the biological model is that an idea can be improved upon or modified by its owner. This improvement is obtained by incorporating a local search mechanism into the global search conducted by the genetic algorithm. A unit of chromosome in the memetic approach is referred to as a *meme* rather than a *gene*, as referred to in the GA approach. For searching the solution space locally, the Tabu search algorithm has been employed and thereby assisting in escaping local minima and visiting promising neighborhood solutions. The basic concept of Tabu Search as described by Glover [Glover, 1989, 1990] is "a meta-heuristic superimposed on another heuristic". It aims at avoiding entrainment in cycles by forbidding or penalizing those moves, which in the subsequent iteration explore those points (solutions) in the solution space that were visited previously (hence the name "Tabu"). A stepwise procedure for the MA is described as follows:

1. Initialize candidate solution population randomly (Generation index, *Gen* = 0)
2. Perform the global search (as in GAs) as follows.
   (a) Evaluate fitness of all the solutions using a fitness function and rank the solutions in the decreasing order of the fitness scores.
   (b) Select pairs of parent solutions with high fitness randomly.
   (c) Perform crossover between parents strings.
   (d) Perform mutation over crossed-over strings.
3. Perform a local search on the candidate solution population obtained from the global search as follows.
   (a) Perturb each candidate solution to obtain a pre-specified number of neighborhood solutions.
   (b) Select the best solution in the neighborhood.
   (c) Discard the best solution if already present in the Tabu list (this list contains a fraction of the candidate solutions visited in the past).
   (d) Apply the aspiration criterion to the discarded solution, and accept the solution if it obeys the aspiration criterion.

(e) GOTO (a) until the termination criteria is not satisfied for local search.

4. Increase generation index: $Gen = Gen + 1$.

5. Check for convergence.

(i) If converged (that is fitness score of the best solution no longer increases substantially or the algorithm has performed a fixed number of iterations), then stop.

(ii) If not then go to step 2.

6. Solution with the highest fitness score is the optimal solution.

Applying a local search in the vicinity of a candidate solution assists in locating promising solutions in the neighborhood and thereby substantially improving the convergence speed of an MA for multimodal problems. To design an efficient search algorithm, the application of Tabu should be carefully considered [Lin et al., 2004]. If we want to efficiently utilize the global search ability of an MA, then we must reduce the computation time spent in the Tabu-based local search. This can be realized by restricting the number of neighborhood solutions examined by the local search procedure.

2.5 DIMENSIONALITY REDUCTION FORMALISMS

Huge amounts of data comprising values of operating and output variables is generated and archived continuously during a typical process operation. These data inherently contain instrumental and measurement noise. Thus, it becomes necessary to treat/preprocess the data with a view to reducing their dimensionality. Dimensionality reduction, apart from generating noise-free data also helps in constructing parsimonious process models possessing an excellent generalization capability.

The most widely used dimensionality reduction technique is Principle Component Analysis (PCA). Being a linear technique, PCA fails to capture nonlinear correlations that are frequently present among real life process data. To overcome this problem a number of novel nonlinear dimensionality reduction techniques have been recently introduced [Tenenbaum et. al., 2000]; these are:

- AANN (Auto-associative Neural Network)
- SAMMANS Mapping and SAM-ANN
- CCA (Curvilinear Component Analysis)
- LLE (locally linear embedding)

2.5.1 Principle Component Analysis

The PCA method [Geladi and Kowalski, 1986] extracts linear relationships existing among the variables of a data set. PCA was first introduced in statistics by Pearson [1986], who formulated the analysis as finding "lines and planes of closest fit to systems of points in space". PCA was briefly mentioned by Fisher and MacKenzie [Fisher and MacKenzie, 1923] as more suitable than analysis of variance for the modeling of response data. Fisher and MacKenzie also outlined the nonlinear iterative partial least squares (NIPALS) algorithm, later rediscovered by Wold [Wold, 1966]. Hotelling [Hotelling, 1933] further developed PCA to its present stage. The PCA decomposes a single data set comprising measurements of linearly correlated variables into a transformed variable set defining the eigenvectors of the covariance of the data and the associated parameters. In essence, PCA generates a set of pseudo-measurements (called 'scores' or 'latent variables'), which are linearly independent (uncorrelated). The important feature of the PCA is that successive latent variables capture decreasing amount of variability in the data.

To illustrate the PCA method, consider a two dimensional matrix, $X(I,J)$, defining I measurements of J variables. The PCA decomposes, X, into matrices of latent variables and the corresponding parameters (known also as "loadings") as given by:

$$X = TP' + E \tag{2.63}$$

where, matrix X is assumed to be mean-centered (mean = 0) and variance-scaled (i.e. the standard deviation of elements of each column is unity); T (I,J) denotes the matrix of J principal component (PC) scores (each column of matrix T signifies a principal component); P' refers to the transpose of the loading matrix, $P(J,J)$, and E denotes the residuals. In the event of linearly correlated variables,

first R principle component scores capture a large amount of variance in the data, and thus Eq. (1.63) can be rewritten as

$$X = \sum_{r=1}^{R} t_r (p_r)' + E' \tag{2.64}$$

where, t_r denotes the I-dimensional rth score vector; p_r refers to the transpose of the rth J- dimensional loading vector, p_r, and E' denotes the residual matrix. It can be seen from Eq. (2.64) that the original ($I \times J$) dimensional data matrix, X, can now be represented in terms of R number of I-dimensional score vectors. Since R is smaller than J, the original data can be represented in terms of a smaller matrix. The sum of squares of elements of a score vector (t_r) is related to the eigenvalue (also known as "trace") of that vector and it serves as a measure of the variance captured by the rth principle component. It thus follows that larger the magnitude of trace, more significant is the respective principal component.

2.5.2 Curvilinear Component Analysis

Curvilinear component analysis (CCA) was proposed by Demartines and Herault [1995] as an improvement to the Kohonen self-organizing maps [Kohonen, 1989] (also see Section 2.3.2); the output is not a fixed lattice but a continuous space able to take the shape of the submanifold. As it turns out, the projection part of CCA is similar in its goal to other nonlinear mapping methods, such as multidimensional scaling (MDS) [Shepard et al., 1965] and Sammon's nonlinear mapping (NLM) [Sammon, 1969], in that it minimizes a cost function based on the inter-point distances in both input and output spaces. The primary objective of CCA is to generate a revealing representation of the original data in a lower dimensional feature space so as to prepare a foundation for the further clustering of the input data [Demartines et al., 1997]. The CCA operates on the principle of preserving distances in its input and output (projected) spaces. That is, all the possible distances between points in the input space are expected to match the corresponding distances between the points in the reduced dimensional projected space. However, in the case of nonlinearly correlated input data, it may not be possible to preserve distances of large magnitudes since the task necessitates unfolding of the manifold to effect dimensionality reduction in the

projected space. This difficulty can be addressed by preserving at least smaller (i.e. local) distances, which then leads to stretching of the larger distances (known as "global unfolding"). For achieving the preservation of local distances, the CCA employs a neighborhood function, which fulfills the condition of preservation of smaller distances, while relaxing the condition for larger distances [Buchala et al., 2004].

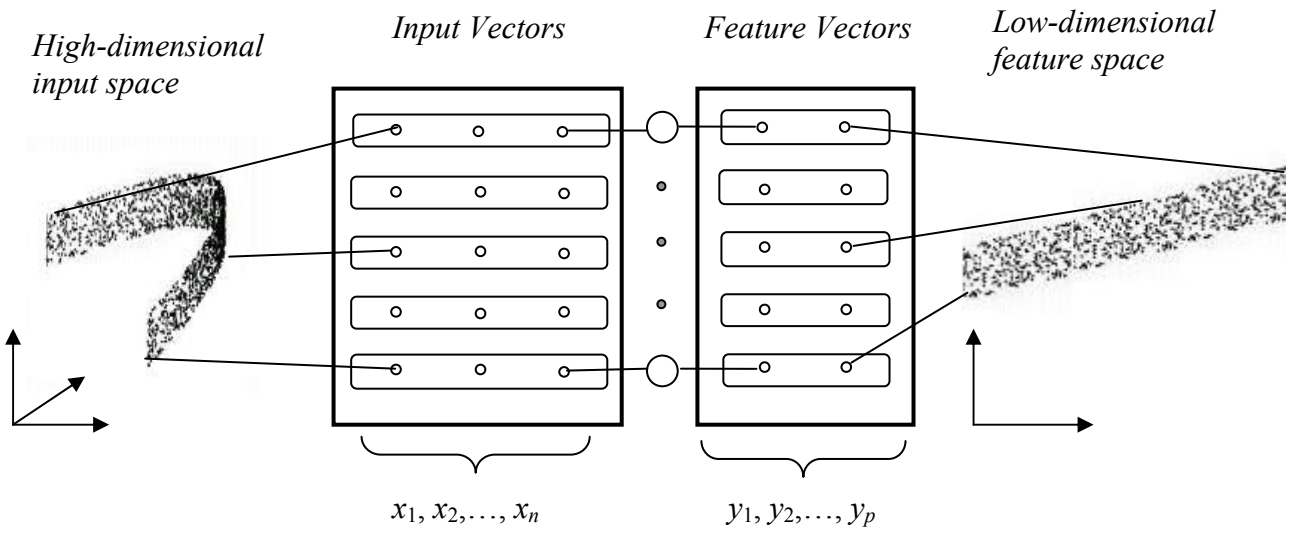

Figure 2.15: A schematic of the CCA network

The CCA can be considered as a self-organizing neural network (see Figure 2.15) that performs two tasks: (i) vector quantization (VQ) of the submanifold in the data set (input space), and (ii) nonlinearly projecting the quantized vectors onto the output space. A vector quantizer maps *n*-dimensional vectors in the vector space, $\Re^n$, into a finite set of vectors thereby reducing the original data set to a smaller but still representative set to work with. After training, the CCA network has the ability of generalization owing to which it can continuously map any new point in the forward or backward direction. The CCA as shown in Figure 2.15 performs the VQ and nonlinear projection tasks separately using two layers of connections. The first network layer performs vector quantization on the data set and the second layer (known as the "projection layer") conducts topographic mapping of the quantized vectors. The projection layer is a free space, which takes

the shape of the submanifold of the data [Buchala et al., 2004]. The principal advantages of the CCA method over other dimensionality reduction algorithms are as follows [Demartines et al., 1997]:

(i) It uses a new cost function that enables unfolding of strongly nonlinear or even closed structures.

(ii) The cost function allows selection of the scale at which the unfolding of the submanifold is performed.

(iii) Significantly faster as compared to other nonlinear feature extraction formalisms owing to a new implementation method that necessitates computation of only a few distances.

(iv) It allows interactivity whereby the user has control over the minimized function itself – specifically on the scale at which the distances have to be preferably respected.

The training algorithm for the CCA network was proposed as an improvement to the Kohonen's self-organizing map (refer Section 2.3.2) wherein the output is not a fixed lattice but a continuous space capable of taking the shape of the manifolds of the input data. In what follows the procedural details of CCA training are described.

Let $\{\mathbf{x}_i\}$; $i = 1, 2,\ldots, N$, be the set of data vectors ($\mathbf{x}_i = [x_{i1}, x_{i2},..., x_{in}]^T$) in an n-dimensional input space, and $\{\mathbf{y}_i\}$ be the corresponding vectors ($\mathbf{y}_i = [y_{i1}, y_{i2},..., y_{ip}]^T$) in the p-dimensional ($p < n$) feature space. Accordingly, each of the n neurons (processing elements) in the CCA network has two weight vectors $\mathbf{x}_i$ and $\mathbf{y}_i$ associated with it. During training of the network, the processing elements (PEs) in the first layer force the input vectors to become the prototypes of the distribution using any standard vector quantization methods [Ahalt et al., 1990]. The output layer PEs are required to construct a nonlinear mapping of the input vectors. This objective is fulfilled by minimizing the structure differences between the quantized and output spaces. The structure differences can be described in terms of the Euclidean distances and the corresponding quadratic cost function (E) to be minimized for reducing the data dimensionality from n to p is given as,

$$E = \frac{1}{2}\sum_i \sum_{j \neq i} \left[X_{ij} - Y_{ij}\right]^2 F\left(Y_{ij}, \lambda_y\right) \tag{2.65}$$

where, $X_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$ describes the Euclidean distance between vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, and $Y_{ij} = d(\mathbf{y}_i, \mathbf{y}_j)$ refers to the corresponding distance in the output space. The objective of minimizing E is to force Y_{ij} to match X_{ij} for each possible vector pair (i, j). Since, a perfect match between X_{ij} and Y_{ij} is not possible while mapping to a lower dimensional space, a weighting function $F(Y_{ij}, \lambda_y)$ is used to favor the conservation of the local topology. For preserving this topology, a bounded and monotonically decreasing weighting function such as the decreasing exponential or sigmoid function is commonly chosen. The weighting function assigns greater weightage to points lying closer in the output space.

In the beginning, the set of p-dimensional output vectors, $\{\mathbf{y}_i\}$, are initialized randomly to small magnitudes. The minimization of E with respect to the vectors, $\mathbf{y}_i$, is performed by following the procedure outlined by Demartines et al. [1997]. This procedure temporarily fixes one of the $\mathbf{y}_i$ vectors and moves all the other $\mathbf{y}_j$ vectors around it without a concern to the interactions among the $\mathbf{y}_i$ vectors. The updating rule for a $\mathbf{y}_j$ vector to effect minimization of E is given as [for the detailed discussion see Demartines et al., 1997]:

$$\Delta \mathbf{y}_j(i) = \alpha(t)\nabla_i E_{ij} = -\alpha(t)\nabla_j E_{ij} = \alpha(t)F(Y_{ij}, \lambda_y)(X_{ij} - Y_{ij})\frac{\mathbf{y}_j - \mathbf{y}_i}{Y_{ij}}, \forall\ j \neq i \quad (2.66)$$

where, i refers to the index of a randomly chosen vector; $\nabla_i E_{ij}$ represents the gradient of E with respect to $\mathbf{y}_i$, and $\alpha(t) = \frac{\alpha_0}{(1+t)}$ denotes the learning rate that decreases with time and λ_y is a constant that signifies the neighborhood criteria. The optimized $\mathbf{y}_j$-updation rule in Eq. (2.66) is numerically efficient, and its implementation results in the output vectors eventually converging to L number of prototypes ($\mathbf{y}_{i^*}$, i = 1, 2,…, L) in a certain number (<100) of training iterations. The CCA training algorithm can now be briefly summarized as:

Step 1: Initialize Y_{ij} in the projected space by using the linear PCA.

Step 2: Compute Euclidean distances $d(\mathbf{x}_i, \mathbf{x}_j)$ in the input space.

Step 3: Compute distances $d(\mathbf{y}_i, \mathbf{y}_j)$ in the projected space.

Step 4: Update all the projected vectors according to the equation (Eq. 2.66).

Step 5: Decrease the value of λ_y with the iteration index, t.

Step 6: Repeat steps 3-5 until the change, $\Delta \mathbf{y}_j(i)$, in the projected space is less than a pre-specified threshold or the maximum number of iterations (t_{max}) is reached.

The CCA is an efficient nonlinear dimensionality reduction technique although other formalisms such as the SOM are necessary for classification and projection if the dimensionality of the projected space is very high (>3).

2.5.3 Autoassociative Neural Networks

The autoassociative neural network (AANN) is an efficient nonlinear principle component analysis formalism [Kramer, 1991, 1992; Leonard and Kramer, 1993; Kuespert and McAvoy, 1994]. It performs auto-association and in this process also conducts nonlinear feature extraction and dimensionality reduction of a multivariable data set. In auto-association, an input-output nonlinear mapping is performed in a manner such that the desired output of the network is same as its input. The AANN conducts the auto-association in a novel manner whereby the input information is initially compressed using a small number of hidden layer neurons, following which decompression of the compressed information is effected. This way, an AANN yields as its output, an approximation of its input. Unlike the widely employed three-layered MLP neural network, an AANN (see Figure 2.16) comprises five layers namely *input*, *mapping*, *bottleneck*, *demapping* and *output* layers. This network architecture performing nonlinear identity mapping from a J-dimensional input space to the J-dimensional output space can also be viewed as a hybrid of two MLP networks, each with a single hidden layer. The first MLP network (known as "compression" network) (Figure 2.17a) consists of the *input*, *mapping* and *bottleneck* layers, respectively, while the second network (known as "decompression" network) (Figure 2.17b) consists of *input*, *demapping* and *output* layers. The input to the first network (rows of matrix, X) is known, whereas its output is unknown. In

contrast, for the second network, the input is unknown but the output is known. These two networks are combined in a manner such that the output of the first network is the input to the second one. Typically, the number of nodes (NH_2) in the bottleneck layer is much smaller than the number of input and mapping layer nodes. In an AANN, the outputs of nodes in the bottleneck layer represent the nonlinear principal components. As its name signifies, the compression network performs compression of an input vector, **x**, and this compressed version forms the output of the bottleneck layer nodes. The second network accepts (as its input) the output from the bottleneck layer and decompresses it so as to produce an approximation of the vector, **x**, at the output layer nodes.

The commonly utilized method for training an AANN is the error-back-propagation algorithm [Rumelhart et al., 1986]. The AANN is trained to obtain an identity mapping, which involves adjusting the network weights in a manner such that the network outputs become identical to the inputs. A small error between the input and its reconstruction (AANN output) ensures that the output from the bottleneck layer contains a compact representation of the input data set. In order to uncover nonlinearities between the input data variables, the neurons in the mapping and demapping layers should use a nonlinear transfer function such as the logistic sigmoid.

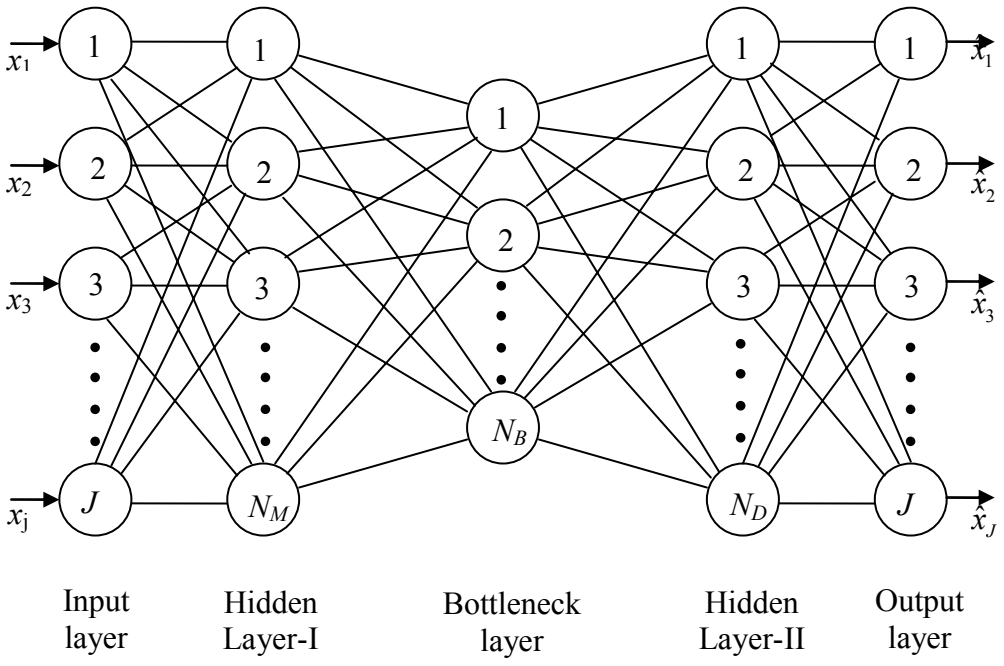

Figure 2.16: The schematic representation of architecture of AANN

The number of nodes in the mapping and demapping layers of an AANN are problem-dependent and require a heuristic optimization. However, too many nodes in these layers may lead to an over-fitted network owing to which it may not generalize well. Thus, the same precautions as outlined for the three-layered MLP network (see Section 2.2.1) must be exercised while creating an optimal AANN. Apart from their use in dimensionality reduction and extraction of non-linear principal components, the AANNs can also be utilized for denoising of input vectors [Kuespert and McAvoy, 1994].

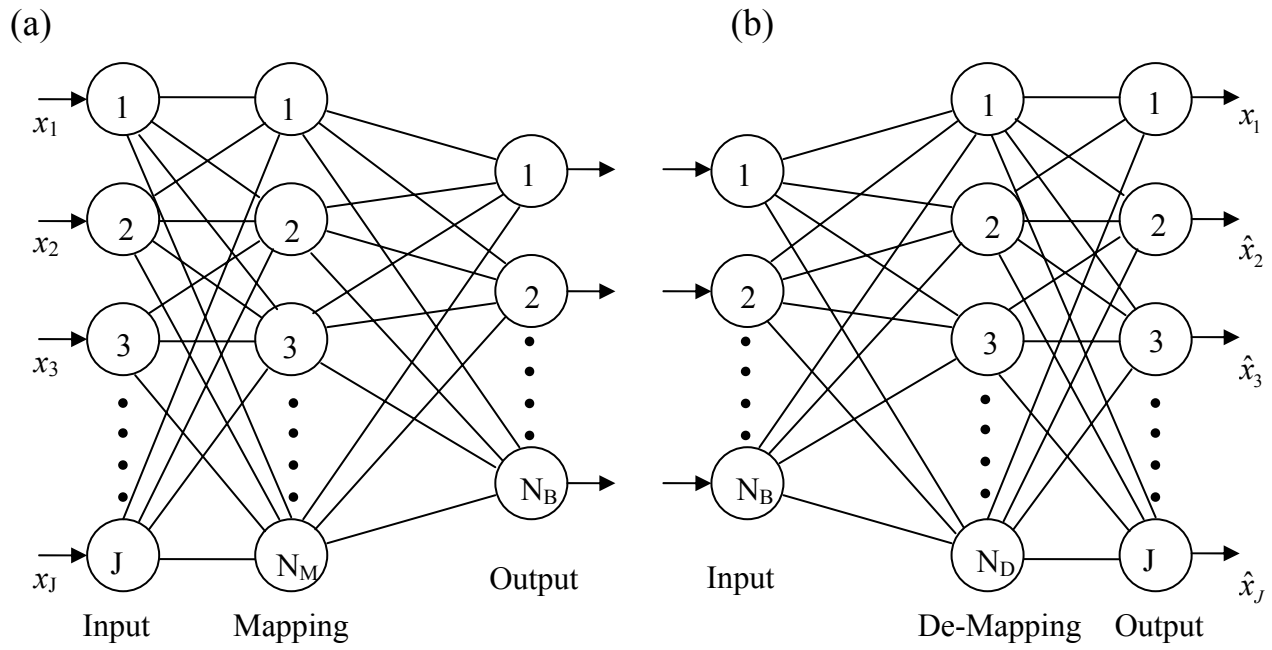

Figure 2.17: Compression and decompression networks performing: (a) mapping, and (b) de-mapping of the input data

Two approaches are possible for extracting the nonlinear principal components (PCs) using an AANN. In the first approach, known as "simultaneous extraction," the AANN training is conducted by taking as many nodes in the bottle-neck layer as the desired number of nonlinear PCs. Thus, a single AANN extracts all the desired number of nonlinear PCs. However, this method though effective for dimensionality reduction, the nonlinear PCs that it extracts could still be correlated. To minimize the correlation existing between the nonlinear PCs, the second approach known as "sequential extraction" is used. In this approach, the

number of nodes in the bottleneck layer always equals one. The corresponding NLPCA procedure involves sequential training of several AANNs (each with a single bottle-neck node) extracting only one nonlinear PC at a time. Here, the first AANN is trained to obtain the first nonlinear PC; the residual matrix, formed by the difference between the input and the output of the first AANN is then used to train the second AANN to obtain the second nonlinear PC and so on. Thus, the sequential extraction approach requires training a number of AANNs equal to the desired number of nonlinear principal components. In this method, each AANN extracts a different feature in the input data. As a result, nonlinear PCs obtained by this method are expected to be less correlated and more orthogonal to each other than the PCs given by the simultaneous extraction method.

2.5.4 Locally Linear Embedding

Dimensionality reduction by locally linear embedding (LLE) [Roweis and Saul, 2000] involves identifying the underlying structure of the manifold, while projections of the data by PCA map faraway data points to nearby points in the plane. Similar to PCA, the LLE algorithm is simple to implement, and its optimization does not involve local minima while at the same time it is capable of generating highly nonlinear embeddings. The LLE algorithm is based on simple geometric intuitions. Consider a dataset consisting of N real-valued vectors X_i, each of dimensionality D, sampled from some smooth underlying manifold. Provided that there exist sufficient data so that the manifold is well-sampled, it is expected that each data point and its neighbors lie on or close to a locally linear patch of the manifold. Thus, it is possible to characterize the local geometry of these patches by linear coefficients that reconstruct each data point from its neighbors. The simplest LLE formulation identifies K nearest neighbors per data point, as measured by the Euclidean distance; neighbors are identified by choosing all points within a ball of fixed radius. Reconstruction errors are then measured by the cost function:

$$C(W) = \sum_i | X_i - \sum_j W_{ij} X_j |^2 \tag{2.67}$$

This adds up the squared distances between all the data points and their reconstructions. The weights W_{ij} represent the contribution of the j^{th} data point to the i^{th} reconstruction. To compute the weights W_{ij}, the cost function $C(W)$, is minimized subject to two constraints: (i) that each data point X_i is reconstructed only from its neighbors, which forces $W_{ij} = 0$ if X_j does not belong to this set and, (ii) that the rows of the weight matrix sum to one i.e. , $\Sigma W_{ij} = 1$. The optimal weights W_{ij} subject to these constraints are found by solving a least squares minimization problem.

Consider that the data lie on or near a smooth nonlinear manifold of dimensionality $d << D$. Then, assuming good approximation, there exists a linear mapping—comprising a translation, rotation, and rescaling—that maps the high dimensional coordinates of each neighborhood to global internal coordinates on the manifold. According to the design reconstruction, weights W_{ij}, reflect intrinsic geometric properties of the data that are invariant exactly to such transformations. Hence, their characterization of local geometry in the original data space should be equally valid for local patches on the manifold. Specifically, the same weights W_{ij} that reconstruct the i^{th} data point in D dimensions should also reconstruct its embedded manifold coordinates in d dimensions. The LLE essentially constructs a neighborhood-preserving mapping based on the above idea. In the final step of the algorithm, each high dimensional observation X_i is mapped to a low dimensional vector Y_i representing global internal coordinates on the manifold. This is done by choosing d-dimensional coordinates Y_i to minimize the embedding cost function:

$$\phi(Y) = \sum_i | Y_i - \sum_j W_{ij} Y_j |^2 \qquad (2.68)$$

Similar to the cost function in Eq. (2.67), this cost function is based on locally linear reconstruction errors, however here the weights, W_{ij} are fixed while optimizing the coordinates Y_i. The embedding cost in Eq. (2.68) defines a quadratic form involving vectors Y_i. Subject to the constraints which make the problem well- posed, the cost function can be minimized by solving a sparse $(N \times N)$ eigenvector problem, whose bottom d non-zero eigenvectors provide an ordered set of orthogonal coordinates centered on the origin. While the reconstruction weights for each data point are computed from its local neighborhood—independent of the weights for other data points—the embedding

coordinates are computed by an $(N \times N)$ eigen solver, which is a global operation that couples all data points in connected components of the graph defined by the weight matrix. The different dimensions in the embedding space can be computed successively which is performed simply by computing the bottom eigenvectors from Eq. (2.68) one at a time. However, the computation is always coupled across data points. In this manner, the algorithm leverages overlapping local information to discover the underlying global structure.

Implementation of the LLE algorithm is fairly straightforward, since the algorithm has only one free parameter namely the *number of neighbors per data point* (K). Once neighbors are chosen, the optimal weights W_{ij} and coordinates Y_i are computed by the standard linear algebra methods. The algorithm involves a single pass through the three steps detailed below and finds the global minima of the reconstruction and embedding costs in Eqs. (2.67) and (2.68). In an event when the neighbors outnumber the input dimensionality $(K>D)$, the least squares problem for finding the weights does not possess a unique solution. In such cases, a regularization term—for instance, one that penalizes the squared magnitudes of the weights—must be added to the reconstruction cost. This is done by adding regularization parameter to the algorithm to break the degeneracy. For simplicity of the algorithm, the parameter is set to a constant value or it is set automatically.

LLE algorithm:

(1) Compute the neighbors of each data point, X_i;

(2) Compute the weights W_{ij} that best reconstruct each data point X_i from its neighbors, minimizing the cost in Eq. (2.67) by constrained linear fits;

(3) Compute the vectors Y_i that best reconstructed by the weights W_{ij}, minimizing the quadratic form in Eq. (2.68) by its bottom nonzero eigenvectors.

2.5.5 Sammon's Mapping and ANN-based Sammon's Mapping

The Sammon's mapping (SM) is a multidimensional-scaling (MDS) based useful tool in nonlinear pattern recognition practice [Sammon, 1969]. It maps a dataset of dimensionality D, onto a non-linear subspace of d dimensions (where d

< *D)*, preserving as well as possible the inter-pattern distances. The SM is often used to visualize high-dimensional data in two or three dimensions although it can be used to map a high-dimensional dataset to any low-dimensional space, i.e. the output is not restricted to only two or three dimensions. The SM algorithm performs a mapping to a *d*-dimensional space by minimizing the following error function (also known as "Sammon's stress", E_{SAM}):

$$E_{SAM} = \frac{1}{\sum_{i<j}^{N} D_{ij}} \sum_{i<j}^{N} \frac{\left(d_{ij} - D_{ij}\right)^2}{D_{ij}} \tag{2.69}$$

where, d_{ij} (D_{ij}) refers to the distance between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ ($i \neq j$) in the low (high) dimensional space.

Equation (2.69) expresses how well the distances in the output space (i.e., *d*-dimensional) fit the distances in the original *D*-dimensional input space, giving more weightage to the small distances. Minimization of E_{SAM} is performed by using a classical gradient descent technique, in which a search is conducted in the direction opposite of the gradient of $E_{SAM}(X)$.

A. Gradient decent algorithm of Sammon's Mapping

1. Define an initial configuration $X^{(0)}$. at iteration *t*:
2. Compute the distances d_{ij} for the current configuration $X^{(t)}$, one such refinement will be required.
3. ∇E_{SAM} : gradient vector of function $E_{SAM}(X)$, evaluated at X:

$$(\nabla E)_k = \frac{\partial E(X)}{\partial x_\mu^r}, \quad k = (\mu - 1)d + r\,; \qquad k = 1,2,\ldots,N_d \tag{2.70}$$

4. $X^{(t+1)}$ is obtained by: $X^{(t+1)} = X^{(t)} - \alpha(t)\nabla E^{(t)}$, (2.71)
5. STOP if E_{SAM} has converged, else GOTO step 2.

A significant disadvantage of Sammon's mapping algorithm is that it can not be generalized to yield a mapping of new or previously unseen data points. That is, when a new point has to be mapped, the entire mapping exercise must be repeated. To overcome this drawback, Mao and Jain [1995] proposed the artificial neural network (ANN) based algorithm known as SAMANN for Sammon's mapping. In this paradigm, a feed-forward neural network such as the multi-layer

perceptron (see Section 2.2.1) with an unsupervised learning rule is utilized to minimize the Sammon's stress. Once trained properly, the SAMANN weight parameters can be used to perform dimensionality reduction and feature extraction of newer/unseen data.

B. ANN-based implementation of Sammon's mapping (SAMANN)

The basic principle underlying SAMANN is the usage of a feed-forward artificial neural network (FFNN) to interpolate and extrapolate the Sammon's mapping, for overcoming the associated generalization problem. In the study by Mao and Jain [1995], a specific back-propagation like learning algorithm (SAMANN) is developed to allow a standard FFNN to learn the Sammon's mapping in an unsupervised way (see Figure 2.18). In each learning step, two points ($\mathbf{x}_i$, $\mathbf{x}_j$) are presented to the SAMANN and the outputs of each neuron are stored for both points. The distance between SAMANN's output vectors is evaluated and an error measure can be defined in terms of this distance and the distance between the points in the original D-dimensional input space. From this error measure, a weight update rule can be derived for its minimization. Since no output examples are necessary, this is an unsupervised training algorithm.

Let $\mathbf{x} = (x_1, x_2,\ldots,x_D)$ be a D-dimensional input vector. We denote the output of the j^{th} unit in SAMANN's layer, l, by $y_j^{(l)}$, j =1, 2,….n_l, l = 1, 2,…,L, where n_l is the number of units in layer l, L is the number of layers, and $y_j^{(0)} = \mathbf{x}_j$, j =1, 2,…D. The weight on connection between unit i in SAMANN's layer l -1, and unit j in layer l, is represented by $w_{ij}^{(l)}$. We denote $w_{0j}^{(l)}$ as the bias for the j^{th} unit in the l^{th} layer and $y_0^{(l)} = 1.0$. The sigmoid activation function, $g(h)$, whose range is (0.0,1.0) is used for each unit, where h is the weighted sum of all the inputs to the units. Thus, the output of the j^{th} unit in layer l can be written as

$$y_j^{(l)} = g\left(\sum_{i=0}^{n_l} w_{ij}^{(l)} y_i^{(l-1)}\right), \quad l = 1, 2, \cdots, L. \tag{2.72}$$

where i and j are the two pattern indices. For simplicity we denote input vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ as μ and ν, respectively.

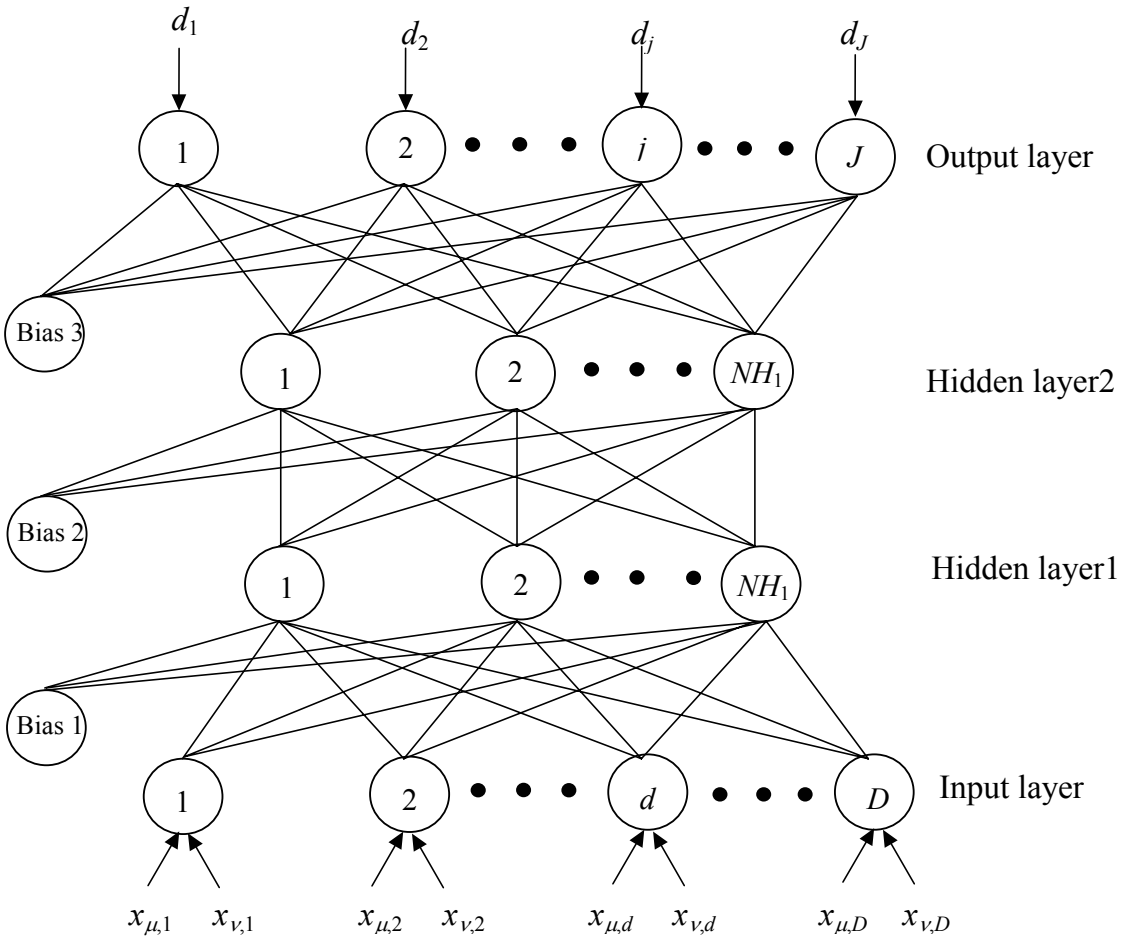

Figure 2.18: SAMANN Architecture

And define distances between the corresponding output vectors as

$$d(\mu,\nu)=\left\{\sum_{k=1}^{m}\left[y_k^{(L)}(\nu)-y_k^{(L)}(\nu)\right]^2\right\}^{1/2} \tag{2.73}$$

Let, $$\hat{\lambda}=\frac{1}{\sum_{\mu=1}^{n-1}\sum_{\nu=\mu+1}^{n}D(\mu,\nu)} \tag{2.74}$$

where, $\hat{\lambda}$ is independent of the network and can be computed before hand. The error function is defined as,

$$E_{\mu\nu}=\hat{\lambda}\frac{\left[D(\mu,\nu)-d(\mu,\nu)\right]^2}{D(\mu,\nu)} \tag{2.75}$$

$$E = \sum_{\mu=1}^{n-1} \sum_{\nu=\mu+1}^{n} E_{\mu\nu} \tag{2.76}$$

Note that the $E_{\mu\nu}$ is proportional to the interpattern distance changes between the patterns μ and ν, due to the projection from the D-dimensional feature space to the d-dimensional projected space. Therefore, $E_{\mu\nu}$ is more appropriate for the pattern-by-pattern based weight updating rules. The updating rule has been derived for the multilayer feed-forward neural network, which minimizes the Sammon's stress based on the gradient descent method. For the output layer ($l = L$):

$$\frac{\partial E_{\mu\nu}}{\partial w_{jk}^{(L)}} = \left(\frac{\partial E_{\mu\nu}}{\partial d(\mu,\nu)} \right) \left(\frac{\partial d(\mu,\nu)}{\partial \left[y_k^{(L)}(\mu) - y_k^{L}(\nu) \right]} \right) \cdot \left(\frac{\partial \left[y_k^{L}(\mu) - y_k^{L}(\nu) \right]}{\partial w_{jk}^{(L)}} \right) \tag{2.77}$$

$$\delta_k^{(L)}(\mu,\nu) = -2\lambda \frac{D(\mu,\nu) - d(\mu,\nu)}{D(\mu,\nu)\, d(\mu,\nu)} \left[y_k^{(L)}(\mu) - y_k^{(L)}(\nu) \right] \tag{2.78}$$

and,

$$\Delta_{jk}^{(L)}(\mu) = \delta_k^{(L)}(\mu,\nu) \left[1 - y_k^{(L)}(\mu) \right] y_k^{(L)}(\mu) \tag{2.79}$$

$$\Delta_{jk}^{(L)}(\nu) = \delta_k^{(L)}(\mu,\nu) \left[1 - y_k^{(L)}(\nu) \right] y_k^{(L)}(\nu) \tag{2.80}$$

where $\delta_k^{(L)}(\mu,\nu)$ is the change in the output scales by the normalized interpattern distance change when patterns μ and ν are presented to network. As will be seen later, $\Delta_{jk}^{L}(\mu)$ and $\Delta_{jk}^{L}(\nu)$ are back-propagated to the layer (L-1).

$$\frac{\partial E_{\mu\nu}}{\partial w_{jk}^{(L)}} = \Delta_{jk}^{(L)}(\mu)\, y_j^{(L-1)}(\nu) - \Delta_{jk}^{(L)}(\nu)\, y_j^{(L-1)}(\nu). \tag{2.81}$$

The updating rule for the output layer is

$$\Delta w_{jk}^{(L)} = -\eta \frac{\partial E_{\mu\nu}}{\partial w_{jk}^{(L)}} = -\eta \left(\Delta_{jk}^{(L)}(\mu)\, y_j^{L-1}(\mu) - \Delta_{jk}^{(L)}(\nu)\, y_j^{(L-1)}(\nu) \right) \tag{2.82}$$

where η is the learning rate. Similarly, we can obtain the general weight updating rules for all the hidden layers, $l = 1, \cdots, L-1$

$$\Delta w_{ij}^{(l)} = -\eta \frac{\partial E_{\mu\nu}}{\partial w_{ij}^{l}} = -\eta\left(\Delta_{ij}^{(l)}(\mu)\, y_i^{l-1}(\mu) - \Delta_{ij}^{(l)}(\nu)\, y_i^{(l-1)}(\nu)\right) \tag{2.83}$$

where,
$$\Delta_{ij}^{(l)}(\mu) = \delta_j^{l}(\mu)\left[1 - y_j^{(l)}(\mu)\right] y_j^{l}(\mu) \tag{2.84}$$

$$\Delta_{ij}^{(l)}(\nu) = \delta_j^{l}(\nu)\left[1 - y_j^{(l)}(\nu)\right] y_j^{l}(\nu) \tag{2. 85}$$

and,
$$\delta_j^{(l)}(\mu) = \sum_{k=1}^{m} \Delta_{jk}^{(l+1)}(\mu)\, w_{jk}^{(l+1)} \tag{2.86}$$

$$\delta_j^{(l)}(\nu) = \sum_{k=1}^{m} \Delta_{jk}^{(l+1)}(\nu)\, w_{jk}^{(l+1)} \tag{2.87}$$

Analogous to the standard back-propagation learning algorithm, $\delta_j^{(l)}(\mu)$ and $\delta_j^{(l)}(\nu)$ are changes in layer l back-propagated from its successive layer, l +1, for patterns μ and pattern ν, respectively. As can be seen from equations (2.82) and (2.83), to update weights, we need to present a pair of patterns to the network instead of one pattern at a time as in the standard back-propagation learning algorithm. To do this, we can either build two identical networks or just store all the outputs of the first pattern before we store the second pattern. In order to improve the convergence speed and to reduce the possibility of entrapment into a local minimum, a momentum term similar to the standard back-propagation algorithm [Rumelhart et al., 1986] can be added to the weight updation equations (2.82) and (2.83).

SAMANN unsupervised back propagation algorithm

1. Initialize weights randomly in the SAMANN network
2. Select a pair of input pattern randomly and present them to the network one at a time; evaluate the network in a feed-forward fashion.
3. Updates weights using (1.86) and (1.87) in the usual back-propagation fashion beginning with the output layer weights.
4. Repeat steps (2) - (3) a large number of times.
5. Present all the patterns in the training set and evaluate the output of the network; compute Sammon's stress and check convergence; if the value of Sammon's stress is below a pre-specified threshold or the number of

iterations (over steps 2-5) exceeds the prespecified maximum number, then STOP; else go to step 2.

The number of hidden layers and the number of neurons in each hidden layer in the multilayer SAMANN should be chosen heuristically albeit judiciously. To achieve the representation power of the Sammon's algorithm, a network with at least $d \times N$ free parameters should be used, where $d \times N$ is the number of variables in Sammon's algorithm. Thus, $d \times N$ becomes a lower bound for the total number of free parameters. This lower bound becomes very large when the number of patterns is large. Random initialization of SAMANN is preferred here [Lerner et al., 1999] as the PCA based initialization – which was used in the classification experiments by Mao and Jain [1995] – was found to yield very similar maps to those of the PCA based feature extractor.

2.5.6 Fuzzy Curves and Surfaces

Modern day chemical processes are complex entities with a diverse set of equipment. Thus, a large number of factors influence the reaction and mass and heat transfer phenomena in these processes. As a result, the corresponding mathematical models also comprise a large number of input variables. All the input variables are not equally important since their influence on the model outputs (*response*/*dependant* variables) vary significantly with some of these showing only negligible effects. Thus, identifying the important (influential) variables from among a number of input variables becomes necessary for securing a parsimonious yet accurate and reliable process model.

A. Review of prior input selection techniques

The most commonly used input selection and dimensionality reduction techniques are those that perform linear transformations, such as *principal component analysis* (PCA) [Ramos et al., 1986]. The major advantage of linear transformations is that they are computationally efficient, easy to design, and typically possess closed-form solutions [Wold et al,. 1987]. However, the PCA and other similar methods extract only linear relationships among the input data

variables and thus they do not perform satisfactorily when input variables are nonlinearly correlated, which is a common feature of high-dimensional data sets.

There exist two general methods namely *sensitivity analysis* (SA) and *mean square error* (MSE), for ranking and selecting the important, nonlinearly correlated inputs of an ANN model. In the SA methodology, sensitivity coefficient matrix (SCM) is computed based on the partial derivatives of ANN outputs with respect to each input. The procedure for computing the sensitivity coefficient matrix for a single hidden layer is outlined by Zurada et al. [1994] and Engelbrecht et al. [1995], while that for a two hidden layer network can be found in Sung [1998]. The MSE methodology can be implemented using either the *forward selection* (FS) or *backward elimination* (BE) methods. In the first stage of the forward implementation [Lin and Cunningham, 1996, 1998; Sugeno and Yasukawa, 1993] a separate single input–single output (SISO) ANN model is employed for approximating the nonlinear relationship existing between each of the input variables and an output variable. Thus, for a system comprising N inputs, as many ANN models are developed following that the MSE between the desired (target) and the model predicted output is evaluated. The particular input whose model yields the smallest MSE magnitude is then selected as the most significant input. Next, this input is used along with each of the remaining inputs to construct (N-1) number of two input–single output models. Here, the specific pair of inputs leading to the smallest MSE magnitude is chosen to be the two most important inputs. This input pair is then utilized in combination with each of the remaining (N-2) inputs, to construct (N-2) number of three input–single output models and thereby determining the three most important inputs. The said procedure is repeated till one of the two criteria gets satisfied: (i) the MSE has reached the pre-specified threshold, and (ii) the MSE does not decrease with the addition of a new input. In the worst case scenario, identification of important inputs requires a total of (N(N+1)/2) models [Lin et al., 1996].

In the BE method for input identification [Takagi and Hayashai, 1991] all the N inputs are first employed to construct an N input–single output ANN model using an appropriate training algorithm such as the error back-propagation [Rummelheart et al., 1986]. Next, each of the N inputs is ignored in turn to develop N number of (N-1) input–single output models. The specific input whose

exclusion does not lead to an increase in the MSE is eliminated as an unimportant input. This input elimination procedure is continued till all the unimportant inputs are removed. Likewise in the forward selection method, the BE approach needs development of at most $N(N+1)/2$ models.

Among the above-stated three commonly employed input selection methods; the sensitivity analysis is computationally least costly since the sensitivity matrix with respect to all the inputs can be evaluated by constructing a single ANN model that uses all available inputs. In contrast the FS and BE methods are computationally expensive since the other two techniques (FS and BE) require development of up to $N(N+1)/2$ models. This is due to the fact that the overall computational effort in developing ANN models by taking even a single input at a time increases dramatically with increasing number of inputs and the size of the training data set. The large number of models that needs to be developed for identifying important inputs makes FS and BE methods practically unattractive.

B. Fuzzy curves and surfaces

A fuzzy logic based method that overcomes the problems of a large computational effort and local minima associated with the MSE approach has been introduced by Lin et al. [1996, 1998]. This method comprising the computation of fuzzy curves and surfaces avoids the process of ANN-based modeling completely and it automatically and quickly isolates the significant independent input variables for use in the development an entire gamut of non-linear models. The fuzzy curve and fuzzy surface techniques allow a rapid development of accurate nonlinear models for large, complex, poorly defined systems [Lin et al., 1996].

The fuzzy logic based method for ranking the inputs comprises two stages. In the first stage, fuzzy curves are computed followed by the evaluation of fuzzy surfaces. Lin et al. [1995, 1996] have developed a fuzzy curve method for establishing the relationship between inputs and the output and thereby identifying the most important inputs.

Consider an input-output data set D, comprising P training patterns of N dimensional inputs $\mathbf{x} = (x_1, x_2, \ldots, x_k, \ldots, x_P)$ and the corresponding output y. Let x_{ik}

and y_k denote the i^{th} variable in the k^{th} ($k \leq P$) pattern and the corresponding output, respectively. The fuzzy curves algorithm can be briefly described as given below.

1. For each input x_i in the k^{th} pattern in the data set D, compute the fuzzy membership function (FMF), $\mu_{i,k}$.

$$\mu_{i,k}(x_i) = \exp\left(-\left(\frac{x_{i,k} - x_i}{b}\right)^2\right) \tag{2.88}$$

The FMF can be interpreted as fuzzy rules for the output y with respect to each input x_i [Lin et. al., 1996):

$$\text{IF } x_i \text{ is } \mu_{i,k}(x_i) \text{ THEN } y \text{ is } y_k \tag{2.89}$$

Thus, for P training patterns, we get P fuzzy rules for each input variable. The parameter b, in the Eq. (2.88), controls the width of the membership function and its magnitude is chosen arbitrarily between 10% and 20% of the range of the specific (i.e., i^{th}) input variable [Lin et al., 1995].

2. Defuzzify the fuzzy membership functions $\mu_{i,k}(x_i)$, using the center of area method to produce a fuzzy curve, C_i, for each input variable x_i.

$$C_i(x_i) = \sum_{k=1}^{p} \frac{\mu_{i,k}(x_i) y_k}{\sum_{k=1}^{p} \mu_{i,k}(x_i)} \tag{2.90}$$

3. Rank the importance of input variables on the basis of the range covered by their individual fuzzy curves. The range (R_{C_i}) of fuzzy curve is calculated as

$$R_{C_i} = C_i^{\max} - C_i^{\min} \tag{2.91}$$

where, $C_i^{\max}$ and $C_i^{\min}$ respectively refers to the maximum and minimum of the fuzzy curve (C_i) for the i^{th} input. Next, the input variables are ranked according to the decreasing R_{C_i} values i.e., the input variable with the height (lowest) R_{C_i} representing the most (least) important input. Ranking can also be done by visual inspection of fuzzy curves in the $x_i - C_i(x_i)$ space. An

input variable with a flat fuzzy curve is identified as less important since it has little influence on the output.

4. Generate the first stage fuzzy surfaces (i.e., 2-*D* fuzzy curves) according to:

$$S_I(x_i, x_j) = \frac{\sum_{k=1}^{p} \mu_{i,k}(x_i)\mu_{j,k}(x_i) y_k}{\sum_{k=1}^{p} \mu_{i,k}(x_i)\mu_{j,k}(x_i)} \tag{2.92}$$

where, x_i and x_j are any two input variables. Evaluate importance of a combination of value of the corresponding performance index $R_S^{i,j}$ which is defined as,

$$R_I^{i,j} = \frac{1}{P\sigma_y} \sum \left(S_I(x_i, x_j) - y_k\right)^2 \tag{2.93}$$

where σ_y refers to the variance of the p number of outputs evaluated as

$$\sigma_y = \frac{\sum_{k=1}^{p} (y_k - \bar{y})^2}{P} \tag{2.94}$$

A smaller magnitude of $R_S^{i,j}$ indicates that the input variable x_i and x_j are more independent.

5. Generate second stage fuzzy surface using following equation

$$S_{II}(x_i, x_j) = \sum \left(S_I(x_i, x_j) - y_k\right)^2 \mu_{i,k}(x_i)\mu_{j,k}(x_j) \tag{2.95}$$

The performance index for the IInd stage fuzzy surface is given as:

$$R_{II}^{i,j} = \frac{1}{P\sigma_y^2} \sum \left(S_{II}(x_i, x_j) - \sigma_y\right)^2 \tag{2.96}$$

The single performance index combining the performance indexes of the Ist and IInd stage fuzzy surfaces is identified as

$$R^{i,j} = \frac{R_I^{i,j}}{1 + R_{II}^{i,j}} \tag{2.97}$$

The second stage fuzzy surfaces use a different principle than the first stage surfaces. These are based on the idea that if the relationship between an input x_i and the output y is random then a local estimate of the variance of y in x_i-y space will be nearly equal to the global value σ_y. On the other hand, if x_i is indeed related to y, then the local variance estimate is expected to differ significantly from the global variance [Lin et al., 1998].

C. Advantages of the fuzzy curves and surfaces

The fuzzy curves and surfaces (FCS) method for identifying significant inputs has following advantages:

(i) Unlike the commonly used forward selection and backward elimination methods that use ANNs, the FCS method does not use a complex nonlinear formalism for modeling a system.

(ii) There exists only one free parameter namely width that needs tuning.

(iii) Unlike ANN-based input identification, the FCS does not suffer from local minima problems.

(iv) The maximum number of models to be developed in the FCS-based input identification is far less than the forward selection and backward elimination methods which makes it numerically efficient.

2.6 CONCLUSION

To summarize, this chapter provides an overview of artificial intelligence based formalisms used in modeling, optimization, classification, dimensionality reduction and input selection. The formalisms and the corresponding algorithm described here have been extensively utilized in the subsequent chapters for conducting several studies involving modeling, classification, optimization, fault detection and diagnosis, monitoring, dimensionality reduction, input selection, etc. of several chemical engineering/technology systems. In some studies, the algorithms presented in this chapter are improvised in the sense to make it applicable for chemical and biochemical systems.

2.7 REFERENCES

1. Abe, T., S. Kanaya, M. Kinouchi, Y. Ichiba, T. Kozuki, and T. Ikemura (2003), Informatics for Unveiling Hidden Genome Signatures, *Genome Research* 13(4): 693–702
2. Adjiman, C. S., I. P. Androulakis, and C. A. Floudas (2000), Global optimization of mixed-integer nonlinear problems. *AIChE Journal*, 46(9), 1769–1798.
3. Admoaitis R. A., R. M. Farber, J. L. Hudson, I. G. Kevrekidis, M. Kube and A. S. Lapedes (1990), Application of neural nets to system identification and bifurcation analysis of real world experimental data. Los Alamos National Laboratory, *Technical Report* LA-UR-90-5 15.
4. Ahalt, A., A. K. Krishnamurthy, P. Chen, and D. E. Melton (1990), Competitive learning algorithms for vector quantization. Neural Networks. *Interactive Pattern Recognition*. New York: Marcel Dekker, 1978. 3: 277–290.
5. Antonisse, J., A new interpretation of schema notation that overturns the binary encoding constraint. *Proc. Of third int. Conf. on genetic Algorithms*, ed. J. David Schaffer, Morgan Kaufmann, San mateo, CA, 1989, 86–91.
6. Arabas, J., Z. Michalewicz, and J. Mulawka (1994), GA Vs PS – A genetic algorithm with varying population size, *Proceedings of the 1st IEEE International Conference on Evolutionary Computing*, eds. Z. Michatewicz, D. Schaffer, H. P. Schewefel, D. Fogel, and H. Kitano, IEEE Service Center, Orlando, FL, Vol. 1, 73–78.
7. Back, T. and H.P. Schwefel (1993), An overview of evolutionary algorithms for. parameter optimization, *Evol Comput.*, 1 (1), 1–23.
8. Bhandari, D., N. R. Pal and S. K. Pal, Directed mutations in genetic Algorithms, *Inf. Sci.*, 1994, 79, 251–270.
9. Bhat N., P. Minderman, T. McAvoy and N. S. Wang (1990), Modeling chemical process systems via neural computation. *IEEE Control Syst. Msg.*, 8, 24–30.
10. Bishop, C. M. (1994), *Neural networks and their applications*. Rev. Sci. Instru., 65, 1803.

11. Buchala S., N. Davey, R. J. Frank, T. M. Gale, M. J. Loomes, W. Kanargard (2004), Gender classification of face images: The role of global and feature-based information. *INCONIP 2004*. 763–768.
12. Bulsari, A. B. (Ed.) (1995), *Neural Networks for Chemical Engineers*, Elsevier, Amsterdam.
13. Burges, C. (1998), A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2, 1–47.
14. Chan, C. W., Cheung, K. C., Jin, H., and Zhang, H. Y. (1999), A constrained Kohonen network and its application to sensor fault detection. In *Automatic Control in Aerospace* 1998. Proceedings volume from the 14th IFAC Symposium. Elsevier Science, Kidlington, UK, 249–54.
15. Chen, B. J., M. W. Chang, C.J. Lin (2001), Load forecasting using support vector machines: A study of EUNITE competition 2001. Report for *EUNITE* competition for smart adaptive systems. Available at http://www.eunite.org
16. Chen, S., C.F.N. Cowan, P.M. Grant (1991), Orthogonal least-squares learning algorithm for RBFNs, *IEEE Trans. Neural Networks* 2 (2), 302–309.
17. Cherkassky, V., Y. Ma (2004), Practical selection of SVM parameters and noise estimation for SVM regression, *Neural Networks* 17 (1) 113–126.
18. Cybenko G. (1989), Approximation by superposition of a sigmoidal function. Math. *Control. Sig. Syst.*, 2, 303–314.
19. Davis, L. (1989), Adapting Operator Probabilities in Genetic Algorithms, in *Proceedings of the 3rd International Conference on Genetic Algorithms*, eds. J. D. Schaffer and Morgan Kaufmann, San Mateo, CA, , 61–69.
20. Demartines, P. and J. Herault (1995), CCA: Curvilinear component analysis, in *GRETSI'95*, Juan-les-pins, France, Sept.
21. Demartines, P., and J. Herault, Curvilinear Component Analysis: A Self-Organizing Neural Network for Nonlinear Mapping of Data Sets, *IEEE trans. on neural networks*, vol. 8(1).
22. Dibike, Y. B., S. Velickov, D. Solomatine (2000), Support vector machines: Review and applications in civil engineering. *Proc. of the 2nd Joint Workshop on Application of AI in Civil Engineering*, Cottbus, Germany.

23. Donat J. S., N. Bhat and T. J. McAvoy (1990), Optimizing neural net based predictive control. *Proc. Am. Control Conf.*, 2466–2471.

24. Drucker, H., C.J.C. Burges, L. Kaufman, A. Smola, V. Vapnik (1997), Support vector regression machines, In: M.C. Mozer, M.I. Jordan, and T. Petsche (Eds.), *Advances in Neural Information Processing Systems* 9, MIT Press, Cambridge, MA, 155–161.

25. Edgar, T. F., Himmelblau, D. M., & Lasdon L. S. (2001), *Optimization of chemical processes*. McGraw-Hill.

26. Eshelman, L. J., R. A. Caruana, and J. D. Schaffer (1989), in 'Proceedings of the 3rd International Conference on Genetic Algorithms', eds. J. D. Schaffer and Morgan Kaufmann, San Mateo, CA, 10–19.

27. Fahlman, S. E. (1988), An empirical study of learning speed in backpropagation networks, Technical Report CMU-CS-88-162, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA.

28. Fisher R. and W. MacKenzie (1923), Studies in crop variation. II. The manurial response of different potato varieties, *Journal of Agricultural Science*, 13, 311–320.

29. Fogarty, T. C. (1989), Varying the Probability of Mutation in the Genetic Algorithm , in *Proceedings of the 3rd International Conference on Genetic Algorithms*, eds. J. D. Schaffer and Morgan Kaufmann, San Mateo, CA, 104–109.

30. Freeman, J. A., Skapura, D. M. (1991), *Neural networks: algorithms, applications and programming techniques*, Addison-Wesley, Reading, MA.

31. Garcia, S., E. P. Scott (1998), Numer. *Heat Trans.* (Part-A), 33, p. 135.

32. Geladi, P., B.R. Kowalski (1986), Partial least-squares regression: a tutorial, *Anal. Chim. Acta*, 185, 1–17.

33. Glover, F. (1986), Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 5, 533–549.

34. Glover, F., and Laguna M., (1997), *Tabu search*. Boston: Kluwer Academic Publishers.

35. Glover, F. (1989), Tabu Search-Part I. *ORSA J. Comput.* **1**, 190–206.

36. Glover, F. (1990), Tabu Search-Part II. *ORSA J. Comput.* **2**, 4–32.

37. Goldberg, D. E. (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA.
38. Goldberg, D. E. (1991), Real coded genetic algorithms, alphabets and blocking, *Complex Systems*, 5, 139–167.
39. Goldberg, D. E. (1989), Sizing Populations for Serial and Parallel Genetic Algorithms, in *Proceedings of the 3rd International conference on Genetic Algorithms*, eds. J. Schaffer and Mogan Kaufmann, San Mateo, CA, 70–79.
40. Gray, G. J., Murray-Smith, D. J., Li, Y., and Sharman, K. C. (1996), Nonlinear model structure identification using genetic programming and a block diagram oriented simulation tool. *Electronic Letters*, 32, 1422–1424.
41. Grefenstette, J. J. (1986), Optimization of control parameters for genetic algorithms, *IEEE Trans. Syst. Man. and Cyber.*, 16, 122–128.
42. Grosman, B., D.R. Lewin (2004), Adaptive genetic programming for steady-state process modeling, *Computers and Chemical Engineering* 28, 2779–2790.
43. Haesloop D. and B. R. Holt (1990), A neural network structure for systems identification. *Proc. Am. Control conf.*, 2460–2465.
44. Haykins, S., (1999), *Neural networks: a comprehensive foundation (2nd edn.)*: Prentice Hall, New Jersey.
45. Hernandez E. and Y. Arkun (1990), Neural network modeling and an extended DMC algorithm to control nonlinear systems. *Proc. Am. Control Conf.*, 2454–2459.
46. Hertz, J., A. Krogh, R.G. Palmer (1991), *Introduction to the Theory of Neural Computation*, Addison-Wesley, Redwood City, CA.
47. Holland, J. H. (1975), *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI.
48. Hoskins J. C. and D. M. Himmelblau (1988), Artificial neural network models of knowledge representation in chemical engineering. *Computers and Chem. Engg.*, 12, 881–890.
49. Hotelling, H. (1933), Analysis of a complex of statistical variables into principal components, *Journal of Educational Psychology*, 24, 417–441 and 498–520.

50. Iba H., T. Kurita, H. Garis, and T. Sato (1993), System Identification Using Structured Genetic Algorithms, *Proc. of the 5th Int. Conf. on Genetic Algorithms*, Urbana-Champaign, USA, 276–286.

51. Jain, B. J., Pohlheim, H., & Wegener J. (2001), On termination criteria of evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, San Francisco, CA: Morgan Kaufmann Publishers.

52. Jayaraman, V. K., Kulkarni, B. D., Karale, S., and Shelokar, P. (2000), Ant colony framework for optimal design and scheduling of batch plants. *Computers and Chemical Engineering*, 24, 1901–1912.

53. Joost, M., and W. Schimann (1998), Speeding up backpropagation algorithms by using cross{entropy combined with pattern normalization. *International Journal of Uncertainty, Fuzziness and Knowledge based Systems* (IJUFKS), 6(2):117–126.

54. Kang, P. and Birtwhistle, D. (1999), Self-organizing map for fault detection. *Intelligent engineering systems through artificial neural networks*, 9:685–690.

55. Kasturi, J., R. Acharya, and M. Ramanathan (2003), An information theoretic approach for analyzing temporal patterns of gene expression, *Bioinformatics*. 19(4): 449–458.

56. Kohonen, T. (1990), The self-organizing map. *Proceedings of the IEEE* 78:1464–1480.

57. Kohonen, T., E. Oja, O. Simula, A. Visa, and J. Kangas (1998), Engineering applications of the self-organizing map. *Proc. IEEE*, 84(10) p.27.

58. Kohonen, T. (1989), *Self-Organization and Associative Memory*, 3rd ed. Berlin: Springer-Verlag.

59. Koza J. R. (1994), *Genetic Programming II*, MIT Press.

60. Koza J. R. (1992), *Genetic Programming: On the programming of computers by means of natural selection*, MIT Press,.

61. Kramer, M. A., (1991) Nonlinear principal component analysis using autoassociative neural networks, *AIChE J.* 37, 223–243.

62. Kramer, M.A. (1992), Autoassociative neural networks. *Comput. Chem. Eng.* 16 (4), 313–328.

63. Krogh, A., J. A. Hertz (1995), A simple weight decay can improve generalization, in: J.E. Moody, S.J. Hanson, R.P. Lipmann (Eds.), *Advances in Neural Information Processing Systems*, Vol. 4, Morgan-Kauffmann, San Mateo, CA, 950– 957.

64. Kuespert, D. R. and McAvoy, T. J. (1994), Knowledge extraction in chemical process control. *Chem. Eng. Comm.* 130, 251–264.

65. Kulkarni, B. D., Tambe, S. S., Dahule, R. K., Yadavalli, V. K. (1999), Consider Genetic Programming for Process Identification. *Hydrocarbon Processing*, 78 (7), 89–97.

66. Kulkarni, S. G., A. K. Chaudhary, S. Nandi, S. S. Tambe and B. D. Kulkarni (2004), Modeling and monitoring of batch processes using principal component analysis (PCA) assisted generalized regression neural networks (GRNN). *Biochemical Engineering Journal*, 18 (3), 193–210.

67. Lakshminarayanan, S., Fujii, H., Grosman, B., Dassau, E., & Lewin, D. R. (2000), New product design via analysis of historical databases. *Computers and Chemical Engineering*, 24(2–7), 671–676.

68. Lapedes A. and R. Farber (1987), Nonlinear signal processing using neural networks: prediction and system modeling. Los Alamos National Laboratory Technical Report LAUR, 87, 2662.

69. Leonard, J. A. and Kramer, M. A. (1993), Diagnosing dynamic faults using modular neural nets. *IEEE Expert* 8, 44–53.

70. Lerner, B. (1999), A Comparative Study of Neural Network Based Feature Extraction Paradigms, *Pattern Recognition Letters*, vol. 20, 7–14.

71. Levin E. (1990), Modeling time varying systems using a hidden control neural network architecture. *Proc. Sixth Yale Workshop on Adaptive and Learning Systems*, 127–l 32.

72. Lin Y., Cunningham G.A. (1995), A new approach to fuzzy-neural system modeling, *IEEE Transactions on Fuzzy Systems*. 3. (2), 190–198.

73. Lin, B., and Miller D. C. (2001), Improvement to the performance of tabu search. In *Proceedings of the AIChE Annual Meeting. Reno*, NV.

74. Lin, B., D. C. Miller (2004), Tabu search algorithm for chemical process optimization, *Comps and Chem. Engg.*, 28, 2287–2306.

75. Lin, Y., G. A. Cunningham III, S. V. Coggeshall (1996), Input variable identification - fuzzy curves and fuzzy surfaces, *Fuzzy Sets and Systems*, 82, 65–71.

76. Lin, Y., G. A. Cunningham III, S. V. Coggeshall, R. D. Jones (1998), Nonlinear system input structure identification: two stage fuzzy curves and surfaces. *IEEE Transactions on Systems, Man, and Cybernetics*, Part A 28(5): 678–684.

77. Lucasius, C. B. and Kateman G. (1989), Application of genetic algorithms in chemometrics, *application of genetic algorithms in chemometrics*. ed. J. David Schaffer, Morgan Kaufmann, San mateo, CA, 170–176.

78. MacQueen, J. B. (1967), Some Methods for classification and Analysis of Multivariate Observations, *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, University of California Press, 1, 281–297.

79. Mao, J. and Jain, A. K. (1995), "Artificial Neural Networks for Feature Extraction and Multivariate Data Projection", *IEEE Trans. Neural Networks*, vol. 6, 296–317.

80. McKay, B., Willis, M. J., & Barton, G. W. (1997). Steady-state modeling of chemical processes using genetic programming. *Computers and Chemical Engineering*, 21(9), 981–996.

81. Michalewicz, Z. (1995). A survey of constraint handling techniques in evolutionary computation methods. In *Proceedings of the 4th Annual Conference on Evolutionary Programming*. Cambridge, MA: MIT Press.

82. Michalewicz, Z. (1992), *Genetic Algorithms + Data Structures = Evolution programs*. Springer-verlag, New York.

83. Michalewicz, Z. (1995), Genetic algorithms numerical optimization and Constraints, in *Proceedings of the 6th International Conference on Genetic algorithms*, cd., L. J. Eshelman and Morgan Kaufmann, San Mateo, CA, 151–158.

84. Moody, J., C. J. Darken (1989), Fast learning in networks of locally-tuned processing units, *Neural Computation*, 1, 281–294.

85. Moscato P. (1992), A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems, in *Parallel Computing and*

Transputer Applications, M. Valero, E. Onate, M. Jane, J.L. Larriba, and B. Suarez, Eds. Amsterdam, The Netherlands: IOS Press, 177–176.

86. Moscato P. and Cotta C. (1999), A gentle Introduction to Memetic Algorithms, in *Handbook of Metaheuristics*, ed Glower F. & Kochenberger G., Kluwer, 1–56.
87. Moscato, P. and C. Cotta (1999), A gentle introduction to memetic algorithms. Glower and G. Kochenberger, Editors, *Handbook of Metaheuristics*, Kluwer, 1–56.
88. Muller, K. R., A. Smola, G. Ratsch, B. Schölkopf, J. Kohlmorgen, V. Vapnik (1997), Predicting time series with support vector machines. In: W. Gerstner, A. Germond, M. Hasler, and J.D. Nicoud (Eds.), *Artificial Neural Networks ICANN'97*, Berlin. Springer, Lecture Notes in Computer Science. 1327, 999–1004.
89. Nadaraya, E. A. (1964), On estimating regression, *Theory of Probability and its Application*, 9, 141–142.
90. Nandi, A. K. (2005), Fault detection using Genetic Programming, *Mechanical system and signal processing*, 19 (2), 271–289.
91. Nandi, S., P. Mukherjee, S. S. Tambe, R. Kumar and B. D. Kulkarni (2002), Reaction modeling and optimization using neural networks and genetic algorithms: case study involving TS-1 catalyzed hydroxylation of benzene, *Ind. Engg. Chem. Res.*, 41, 2159–2169.
92. Narenda K. S. and K. Parthasarathy (1990), Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Networks*, 1, 4–27.
93. Neal R. M. (1996), *Bayesian learning for neural networks*, New York: Springer Verlag,.
94. Park J. and I. W. Sandberg (1991), Universal approximation using radial-basis function networks. *Neur. Comput.*, 3,246–257.
95. Patel, A. N., Mah, R. S. H., & Karimi, I. A. (1991). Preliminary design of multiproduct noncontinuous plants using simulated annealing. *Computers and Chemical Engineering*, 15(7), 451–469.
96. Pearson, K. (1901), On lines and planes of closest fit to systems of points in space, *Philosophical Magazine*, (6) 2, 559–572.

97. Pörn, R., Harjunkoski, I., and Westerlund, T. (1999). Convexification of different classes of non-convex MINLP problems. *Computers and Chemical Engineering*, 23, 439–448.

98. Psichogios D. C. and L. H. Ungar (1991), Direct and indirect model based control using artificial neural networks. *Ind. Engg Chem. Res.*, 30, 2564.

99. Reifman, J., J. Vitela (1994), Accelerating learning of neural networks with conjugate gradients for nuclear power plant applications, *Nucl. Technol.* 106, 225–241.

100. Riedmiller, M. and H. Braun (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. Proceedings *IEEE Int. Conf. on Neural Networks*, IEEE Press, 586–591.

101. Riedmiller, M. (1994), Advanced supervised learning in multi-layer perceptrons { from backpropagation to adaptive learning algorithms. *Computer Standards and Interfaces*, 16, 265–278.

102. Roweis S. T. and Saul L. K. (2000), Nonlinear Dimensionality Reduction by Locally Linear Embedding , *Science*, vol 290, pp 2323–2326.

103. Rumelhart, D., Hinton G., Williams, R., 1986. Learning representations by backpropagating error. *Nature*, 323, 533–536.

104. Rutkowski, L., and K. Cpałka (2003), Flexible neuro-fuzzy systems, *IEEE Trans. Neural Networks*, 14, 554–574.

105. Sammon, J. W. (1969), A nonlinear mapping algorithm for data structure analysis, *IEEE Trans. Comput*., C-18, 401–409.

106. Schimann, W., M. Joost, and R. Werner (1993). Comparison of optimized backpropagation algorithms. In Verleysen, editor, *Proceedings European Symposium on Artificial Neural Networks, ESANN '93*, 97–104, Brussels,.

107. Schneider, G., (1999). How many potentially secreted proteins are contained in bacterial genome. *Gene*. 237:113–121.

108. Schölkopf, J. C., J. Platt, A. J. Shawe-Taylor, Smola; Williamson R. C. (2001), Estimating support of a high-dimensional distribution. *Neural Comput.*, 13, 1443–1471.

109. Schwefel, H. P. (1984), A family of non-linear optimization techniques based imitating on some principles of organic evolution. *Ann. Operations Res.*, 1, 165–167.

110. Shepard, R. N. and J. D. Carroll (1965), Parametric representation of nonlinear data structures, in Proc. *Int. Symp. Multivariate Anal.*, P. R. Krishnaiah, Ed. New York: Academic, 561–592.

111. Simula, O. and J. Kangas (1995). Neural networks for chemical engineers, vol-6 of Computer-Aided Chemical Engineering, Chapter 14, *Process Monitoring and visualization using self-organizing maps*. Elsevier, Amsterdam.

112. Smola, A., B. Schölkopf, K. R. Müller (1998), The connection between regularization operators and support vector kernels. *Neural Networks*, 11, 637–649.

113. South M. C., S. McConnel, M. T. Tham, M. J. Willis (1995), Data Analysis Via Symbolic Regression, *Trans. IChemE*.

114. Spall, J. C. (1987), "A Stochastic Approximation Technique for Generating Maximum Likelihood Parameter Estimates," *Proc. of the Amer. Cont. Conf, AACC, Evanston*, IL, 1161.

115. Spall, J. C. (1998b), An Overview of the Simultaneous Perturbation Method for Efficient Optimization, *Johns Hopkins APL Tech. Dig.*, 19, 482.

116. Spall, J. C. (1998a), Implementation of the Simultaneous Perturbation Algorithm for Stochastic Optimization, *IEEE Trans. Aerosp. Electron. Syst.*, AES-34, 817.

117. Specht, D. F. (1991), A general regression neural network, *IEEE Trans. Neural Networks*, 2, 568–576.

118. Srinivas, M. and L. M, Patnaik (1994), Adaptive probabilities of crossover and mutation in genetic algorithms, *IEEE. Trans. Syst. Man. Cyber.*, 24 (4), 656–667.

119. Syswerda, G. (1989), Schedule optimization using genetic algorithms, in *Handbook of Genetic Algorithms*, ed. L. Davis, Van Nostrand Reinhold, New York, chapter 21, 332–349.

120. Syswerda, G. (1989), Uniform Crossover in Genetic Algorithms, in *Proceedings of the 3rd International conference on Genetic Algorithms*, eds. J. D. Schaffer and Morgan Kaufmann, San Mateo, CA, 2–9.

121. Tambe, S. S., B. D. Kulkarni and P. B. Deshpande (1996), *Elements of Artificial Neural Networks with selected applications in Chemical*

Engineering, and Chemical & Biological Sciences, Simulations & Advanced Controls, Louisville, KY.

122. Tayal, M. C., and Fu, Y. (1999). Optimal design of heat exchangers: A genetic algorithm framework. *Industrial Engineering and Chemical Research*, 38, 456–467.

123. Tenenbaum, J. B., DeSilva, V. and Langford, J. C. (2000), A global geometric framework for nonlinear dimensionality reduction, *Science*, 290, 2319–2323.

124. Thissen, U., R. Van Brakel, A.P. De Weijer, W.J. Melssen, L.M.C. Buydens (2003), Using support vector machines for time series prediction, *Chemom. Intell. Lab. Syst.* (1–2) 35–49.

125. Tryba, V. and K. Goser (1991). *Self-organizing feature maps for process control in chemistry*. In T. kohonen, K. Makisara, O. Simula, and J. kangas, editors, Artificial neural Networks, Amsterdam, Netherlands, 847–852.

126. Ultsch, A. (1993), Self-organizing feature maps for monitoring and knowledge acquisition of a chemical process. In S. Gielen and B. Kappen, editors, *proc. ICANN'1993, Int. Conf. on Artificial neural networks*, London, UK. Springer 864–867.

127. Vaidyanathan, R., and El-Halwagi, M. (1994). Global optimization of nonconvex nonlinear programs via interval analysis. *Computers and Chemical Engineering*, 18(10), 889–897.

128. Vapnik, V., S. Golowich, A. Smola (1996), Support vector method for function approximation, regression estimation and signal processing. *Adv. in Neural Inform. Proces. Syst.*, 9, 281–287.

129. Vapnik, V. (1998), *Statistical Learning Theory*, John Wiley, New York.

130. Vapnik, V. (1995), *The Nature of Statistical Learning Theory*, Springer Verlag, New York.

131. Vasanto, J., E. Alhoniemi, J. Himberg, K. Kiviluto, and J. Parvinainen, (1999). *Self-organizing map for data mining in Matlab: the SOM Toolbox*. Simulation news, Europe, 25–54.
Also see http://www.cis.hut.fi/project/somtoolbox

132. Venkatasubramanian, V., K. M. Chan, and J. M. Caruthers (1994). *Comput. Chem. Eng.*, 18, 833–844.

133. Venkatasubramanian. V., R. Vaidyanathan and Y. Yamamoto (1990), An analysis of the learning, recall, and generalization characteristics of neural networks for process fault diagnosis. *Computers chem. Engg.*, 14, 699–712.

134. Wah, B. W., and Wang, T. (2000). Tuning strategies of constrained simulated annealing for nonlinear global optimization. *International Journal of Artificial Intelligence Tools*, 9(1), 3–25.

135. Wang, C., Quan, H., & Xu, X. (1999). Optimal design of multiproduct batch chemical process using tabu search. *Computers and Chemical Engineering*, 23, 427–437.

136. Wang, H. C., J. Badger, P. Kearney, and M. Li. (2001). Analysis of Codon Usage Patterns of Bacterial Genomes Using the Self-Organizing Map. *Molecular Biology and Evolution* 18:792–800.

137. Watanabe K., I. Matsuura, M. Abe, M. Kubota and D. M. Himmelblau (1989), Incipient fault diagnosis of chemical processes via artificial neural networks, *AK'kE JI* 35, 1803–1812.

138. Watson, G. S. (1964), Smooth regression analysis, *Sankhya Series A*, 26, 359–372.

139. Widrow, B., and Hoff, M. E. (1960). Adaptive switching circuits. *IRE WESTCON Connection Record*, 4, 96–104. Reprinted in [Anderson 1988], 265.

140. Willis M. J. C. DiMassimo, G. A. Montague, M. T. Tham and A. J. Morris (1991), Artificial neural networks in process engineering. *ZEE Proc.-D.* 138, 256–266.

141. Willis, M. J., Hiden, H., Hinchliffe, M., McKay, B., & Barton, G. W. (1997). Systems modeling using genetic programming. *Computers and Chemical Engineering*, 21(1), 1161–1166.

142. Wold, H. (1966), Nonlinear estimation by iterative least squares procedures, in F. David (Editor), *Research Papers in Statistics*, Wiley, New York, 411–444.

143. Wright, A. H. (1991), Genetic Algorithms for Real Parameter Optimization, in *Foundations of Genetic Algorithms*, First Workshop on the Foundations of Genetic Algorithms and Classifier Systems, eds. G. Rawlins and Morgan Kaufmann, San Mateo, CA, 205–218.

144. Xu, L., A. Krzyzak, E. Oja (1993), Rival penalized competitive learning for clustering analysis, RBF net, and curve detection, *IEEE Trans. Neur. Networks* 4, 636–648.

145. Ydstie B. E. (1990), Forecasting and control using adaptive connectionist networks. *Computers and Chem. Engg*., 14,683–599.

146. Yu, H., Fang, H., Yao, P., & Yuan, Y. (2000). A combined genetic algorithm/simulated annealing algorithm for large scale system energy integration. *Computers and Chemical Engineering*, 24, 2023–2035.

147. Zhang B. and H. Muhlenbein (1993), Genetic Programming of Minimal Neural Nets Using Occam's Razor, *Proc. of the 5th Int. Conf. on Genetic Algorithms*, Urbana-Champaign, USA, 342–349.

148. Levenberg, K. (1944), A Method for the Solution of Certain Non-Linear Problems in Least Squares. *Quart. Appl. Math.* 2, 164–168.

149. Marquardt, D. (1963), An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *SIAM J. Appl. Math.* 11, 431–441.

`

CHAPTER 3

PROCESS MODELING

3.1 INTRODUCTION

In this chapter, we have designed, developed or improvised some of the AI and ML based algorithms for building chemical/biochemical process models used in process monitoring, performance enhancement, output predictions, softsensors and model input identification. In the first study, optimal noise superimposition strategy is presented for building more accurate and generalized process models for Reliance Ind. Ltd. To show the generic nature of the suggested strategy (optimal noise superimposition), in another study, the strategy is used for a few more chemical processes. In the third study, the ANN model is developed for the estimation of the gross calorific value of Indian coals. In the next study, a novel ML-based formalism, known as support vector regrssion, is explored to develop softsensors for biochemical processes. The SVR is also utilized for modeling of biochemical systems.

3.2 ARTIFICIAL INTELLIGENCE BASED SOFT-SENSORS FOR MONITORING PRODUCT PROPERTIES IN POLYETHYLENE PROCESS*

Properly designed artificial intelligence based soft-sensors for the quality control variables has helped in improving the performance of the polyethelene (PE) plants while cutting down the need for frequent laboratory analyses of the PE product. Deployment of softsensors has also reduced the extent of an off-spec product and aided better and faster process related decision-making.

Introduction

Sharp excursions in the natural gas prices and an increasing trend of feedstock prices in the last few years have led to almost flat and even negative margins for the global polyolefin industry. Unstable and decreasing end-product selling prices resulting from overcapacities have compounded the problem further. A welcome respite from the severe downturn in margins and financial performance has been offered to the polyolefin industry by a forecast, which anticipates a supply-demand imbalance favoring demand over the next few years [Nahas et al., 1992]. Accordingly, polyolefin manufacturers are gearing up for the impending significant upturn in the polyolefin business by raising the production rates of existing plants, reducing transition time between product changes and minimizing off-spec production.

The Reliance Industries Limited (RIL) (a "Fortune Global 500" company) has two plants at Hajira, India, that produce polyethylene (PE) using the solution polymerization of ethylene technology wherein α-olefins are utilized as the co-monomers. The plants are capable of producing various PE grades comprising homo-polymers, co-polymers and ter-polymers, for meeting the requirements of a wide spectrum of consumer applications. Depending upon the resin to be

---

* Badhe, Y. P., J. Lonari, S. S. Tambe, B. D. Kulkarni, Neelamkumar Valecha, S. V. Deshmukh, B. Shenoy and S. Ravichandran, *Hydrocarbon Processing*, March-2007.

manufactured, process conditions are specified and these are controlled using a sophisticated distributed control system (DCS). The quality of the final polyethylene product is measured in terms of three quality control (QC) variables, namely, *stress exponent*, *density* and *melt flow index* (MFI).

3.2.1 Monitoring of Polyethylene Process

Frequent monitoring of the properties of a PE product is a prerequisite for ensuring consistent and reliable production of a high quality product. For polyethylene, various product grades have closely matching values of properties such as density. It is therefore necessary that quality monitoring mechanisms are accurate, reliable, robust and capable of fast response. Often, on-line hardware sensors are not available for monitoring the product properties and thus these have to be determined using time consuming laboratory analyses. In the event of a process malfunction or an operation under sub-optimal process conditions, the plant continues to produce – till the results of the laboratory tests become available to the operator – a product of an off-spec quality.

What are the problems?

Determination of PE's three QC variables namely, stress exponent, density and melt flow index, involves an analytical laboratory procedure that takes 30 to 45 minutes. This means that a confirmation that the product of a desired quality is being produced comes only after the stated time period. If the laboratory measurements of the three QC variables indicate any deviation from their desired magnitudes, then process conditions need a readjustment. A significant amount of off-spec product may be generated during the course of the laboratory analysis and fine-tuning of process conditions. This is clearly undesired since it has adverse economical implications. The stated difficulty can be overcome by implementing a technological solution in the form of an accurate and robust mathematical model capable of real-time predictions of the three QC variables. In recent years, software based sensors (soft-sensors) that satisfy these requirements have been recommended for monitoring the QC variables.

3.2.2 Suggested Approaches

There exist two approaches namely *phenomenological* and *empirical*, for developing process models. The PE process has complexities such as the relationships between process variables are highly nonlinear and interactive, process dynamics can last for many hours and is through-put dependent, and control variables are often measured in laboratory which is a time consuming process. Also, the involved product grade transitions represent some of the most safety, reliability and cost-critical procedures done in the manufacturing environment [Karagoz et al., 1990]. Being inherently complex and nonlinear, the PE process is difficult to model phenomenologically, which involves development of a "first-principles" model expressed in terms of mass, momentum and energy balance equations. Specific difficulties encountered in the phenomenological modeling of the PE process are: (i) the large number of costly experiments required for studying the effect of influential process variables and parameters on the QC variables, (ii) insufficient knowledge of the underlying physico-chemical phenomena (e.g. reaction kinetics and, heat and mass transport mechanisms), and (iii) stupendous amount of time intensive simulation effort to arrive at a reasonable model.

The second approach to modeling of the PE process is to utilize classical regression methods to formulate empirical models representing the dependency of the QC variables on the key process operating variables. This approach however suffers from that the form of the data-fitting needs to be specified *a priori* before estimation of the function parameter. This is a difficult task since in the PE process multiple variables influence the polymerization phenomenon nonlinearly and the precise nature of their interactions is not fully known.

3.2.3 What is the Solution?

The above-stated difficulties associated with the regression based empirical modeling can be overcome by utilizing an artificial intelligence based modeling formalism known as *Artificial Neural Networks* (see Section 2.2.1). In process engineering, ANNs have been used in diverse applications such as steady-state and dynamic modeling, fault detection and diagnosis, process identification, nonlinear model based control and process optimization (see e.g., Bhat and

McAvoy [1990]; Hernandez and Arkun [1992]; Nahas et al., [1992]; Ramasamy et al., [1995]; Tendulkar et al., [1998]; Nandi et al., [2001] and reviews by Narendra and Parthasarathy [1990]; Hunt et al., [1992]; Agarwal [1997]). Owing to their significant nonlinear function approximation property, ANNs have also been used to develop soft-sensor models [Desai et al., 2005].

3.2.4 Softsensors for Process Monitoring

Soft-sensors are software based sophisticated monitoring systems, which can relate less accessible and infrequently measured process variables with those measured easily and frequently. Once developed, a softsensor model can be readily used for predicting in real-time, the values of less accessible and difficult-to-measure process variables. Softsensors are useful in control and monitoring of chemical processes where owing to the unavailability of appropriate hardware sensors, the values of important QC variables are not available continuously. Availability of accurate, reliable and robust softsensor models assist in reducing product variability and sampling frequency, minimizing an off-spec product formation and enabling process operators and engineers in running the process optimally. Accordingly, Reliance Industries Ltd., decided to explore the possibility of developing ANN-based softsensors for predicting the values of three polyethylene QC variables in real-time and this section details the case history of the development and deployment of these softsensors.

The performance of soft-sensors in accurately predicting the magnitudes of the QC variables depends upon the availability of reliable hardware sensors for monitoring the easily accessible process variables and also on the mathematical and/or statistical techniques used in the correlation and interpretation of process data. The advantages of using ANNs for the development of soft-sensor models for the PE process are: (i) the ANNs are capable of efficient approximation of the nonlinear relationship(s) existing between the operating and the QC variables exclusively from the corresponding historic process data, (ii) for model fitting, *a priori* knowledge of the data-fitting function is not unnecessary since ANNs use a generic nonlinear function for approximating the relationships between the model inputs and outputs, and (iii) models can be built without considering the detailed phenomenological knowledge (kinetics, heat and mass transport mechanisms, etc.)

underlying the process. In essence, given historic process data, an ANN-based soft-sensor model is capable of identifying and capturing the "cause-effect" relationship between PE plant's operating variables (model inputs) and the QC variables (model outputs). Subsequently, given new operating conditions, the soft-sensor models can be used for predicting in real-time the values of the three QC variables.

3.2.5 ANN-Based Softsensor Development

In any set of real-world process data, presence of instrumental noise and/or measurement errors is unavoidable. Thus, the data in their raw noisy form must be used for building the softsensor models. Presence of noise and/or errors in the data creates a threshold limit for the prediction accuracy and generalization performance of the ANN-based softsensor model. Inaccuracies in model predictions, if significant, cannot be tolerated since a number of process control and policy decisions are based on the model predictions. Thus, it is critically important that an ANN model possesses not only excellent prediction accuracy, but also a good generalization property. This property enables an ANN model to predict accurately the outputs corresponding to a new set of model inputs.

The input-output example data set used for training (i.e., fitting) an ANN model is only a finite subset of samples selected from a population of a large number of input-output patterns that can be monitored. Conventionally, ANN models are trained using a suitable parameter (weight) adjustment algorithm that minimizes a pre-specified cost (error) function. For instance, the most widely used error-back-propagation (EBP) algorithm [Rumelhart et al., 1986] (refer to Section 2.2.1A) conducts minimization of the root-mean-squared error (RMSE) function. Often, an ANN model constructed solely on the basis of minimization of the RMSE with respect to the example set outputs is incapable of generalization. Such an inability to generalize arises from an over-fitting of the model. This happens when: (i) the ANN is trained over an excessively large number of training iterations (known as *over-fitting*), and (ii) the number of parameters (weights) to be fitted by an ANN are too high when compared to the number of input-output example samples available for network training (*over-parameterization*). There exist a number of approaches to avoid over-fitting and thereby improving the

generalization performance of an ANN model. The commonly used method to avoid over-fitting is to partition the example input-output dataset into two subsets, namely *training* and *validation* sets. While the ANN weights are adjusted based on the training set error, the validation set is used to continuously monitor the generalization performance of the ANN model undergoing training. Here, the prediction error with respect to the validation set outputs is utilized for selecting the optimal model. A shortcoming of the validation set approach is that it is effective when both the training and validation sets are large and representative [An, 1996]. Else, the chosen ANN model is likely to be biased towards the validation set. Another approach for improving the generalization performance of an ANN model is to superimpose noise (also known as "jitter") on the inputs and outputs of the example set. The assumption underlying the noise-superimposition strategy is that for a well-posed modeling problem a precise solution exists and a small noise addition to the data should produce only small variations in the solution. Thus, for a given example data set, additional network training patterns can be generated by superimposing noise to the input-output elements of the example set patterns. With addition of noise, the resultant ANN-approximated function is defined over continuous ranges of the input plane which assists in increasing the smoothness of the ANN fitted function and thereby improving the generalization ability of the ANN model.

While creating an enlarged noise-superimposed dataset, the noise magnitude must be small since a large amount of noise would clearly distort the intrinsic relationship between the inputs and outputs, while a too small noise amount will lead to insignificant changes of no consequence. It may also be noted that in a nonlinearly behaving system, such as the PE process, the sensitivity with which changes in an input variable affect the output variable, differs significantly. Owing to these factors, it becomes necessary to add varying extent of noise to each input and output data element of the example set. Determining the exact amount of noise to be added to each input-output variable is a tricky issue, which has been successfully addressed by a methodology proposed by Kulkarni et al. [2002]. Specifically, this method generates an enlarged noise-superimposed data set wherein multiple patterns are synthesized by adding an optimal amount of noise to variables of each input-output pattern of the example set. The distinguishing feature of the proposed scheme is that it determines the optimal

amount of noise to be superimposed on each input and output variable of the example set. Here, the task of determining the optimal magnitude of variable-specific noise to be superimposed has been addressed by using a novel optimization formalism known as *Genetic Algorithms* (GA) [Holland, 1975, Davis, 1987, Goldberg, 1989] (refer Section 2.4.3). The GA is an efficient method for searching optimal solutions pertaining to noisy, multi-modal and non-convex objective functions.

3.2.6 Development of Softsensors Using the Proposed Approach

Consider an example process input-output data matrix, $Z(I, J)$, comprising elements, z_{ij}, (i = 1, 2,…, I; j = 1, 2,…, J), wherein first N elements of an ith row (i.e., z_{ij}; i = 1,2,…,I; j = 1,2,…,N) represent model inputs and the next K ($K = J - N$) elements (i.e., z_{ij}; i = 1,2,…,I; j = N+1, N+2,…, N+K) represent the desired (target) model outputs. For the PE process, model inputs describe the process operating variables and parameters while outputs describe the QC variables. The objective of the GA-based optimization is to create an enlarged noise-superimposed data set, $\hat{Z}$, such that when it is used for training an ANN, it yields a model possessing improved prediction accuracy and generalization performance.

The J-dimensional decision vector to be optimized by the GA comprises noise tolerances defined as: $\varepsilon = [\varepsilon_1, \varepsilon_2, ..., \varepsilon_J]^T$. The first N elements of this vector describe the noise tolerance (%) values for the model inputs and the next K elements (N+1 to N+K) represent the noise tolerances for the model outputs ($J = N + K$). These variable-specific tolerances are used to characterize a set of probability density functions (PDF), which are utilized to sample noisy copies of the input-output measurements of the example set, Z. The type of noise to be superimposed on the example data elements is considered to be Gaussian. The procedure of developing an optimal ANN-model using an enlarged noise-superimposed input-output example set is described in Kulkarni et al. [2003].

Soft-sensors for the PE plants

For affording a comparison, the softsensor models were developed using both the original (i.e., non-noise-superimposed) historic example process data sets and their noise-superimposed enlarged versions obtained using the GA formalism. Specifically, sixteen models were developed depending upon the PE plant (I or II),

catalyst operating mode and the co-monomer species used in the PE production. All these models were of multiple input – single output (MISO) type (i.e., $K = 1$). The break-up of the softsensor models and the corresponding output QC variable is given below.

- Models 1 to 5: Stress exponent
- Models 6 to 10: Density
- Models 11 to 16: MFI

For softsensor development, steady-state process data collected over a number of months of PE process operation were utilized. Depending upon the model, the number of model inputs varied between 12 and 14; these comprised various reaction dependent variables and parameters such as ethylene conversion, temperatures at different reactor locations, flow rates, concentrations of reactants and temperature differences within the reactor.

The GA-based optimization of noise tolerances was conducted using following GA-specific parameter values.

- Precision for binary coding of a tolerance variable: 10 bits
- Maximum number of generations over which GA was evolved: 50
- Crossover probability: 0.9
- Mutation probability: 0.05
- Population size: 15

All soft-sensor models were developed using a two hidden layer multi-layer perceptron (MLP) network (see Figure 2.1) and its training was performed using the error-back-propagation (EBP) algorithm with the momentum term [Rumelhart et al., 1986]. To create an optimal soft-sensor model, the effect of MLP's structural parameters (number of nodes in each hidden layer) and EBP-algorithm specific parameters (i.e., learning rate, η, and momentum coefficient, α) was rigorously studied. Also, the effect of random weight initialization on the root-mean-squared error (RMSE) minimization was examined by performing multiple training runs with different seed values of the pseudo-random number generator. The magnitude of the enlargement factor (M) used in creating the enlarged noise-superimposed data sets varied between 10 and 50. Prior to MLP training, the data sets (non-noise-superimposed and noise-superimposed enlarged ones) were partitioned into training and validations sets in 80:20 ratio randomly.

The detailed procedure for obtaining an optimal MLP model can be found, for instance, in Bishop [1994], Freeman and Skapura [1991], Tambe et al. [1996] and Nandi et al. [2001].

Softsensor performance

The prediction and generalization performance of the softsensor models for the three QC variables namely stress exponent, density and MFI, was evaluated in terms of two statistical measures namely RMSE, and the squared coefficient of correlation (R^2) between the model predicted and the corresponding desired outputs. These quantities were evaluated for both training and validation sets.

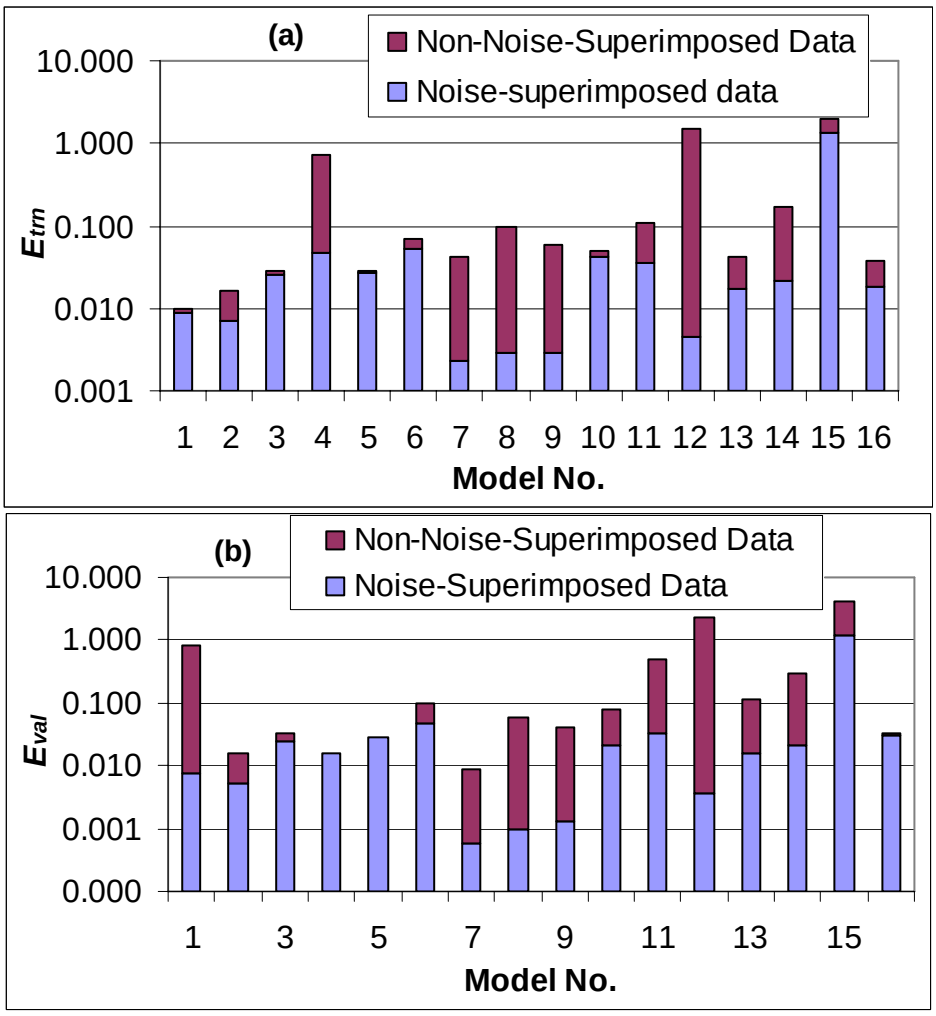

Figure 3.1: (Panel a) Comparison of RMSE values corresponding to the soft-sensor models based on noise-superimposed and non-superimposed training data sets. (Panel b) same as panel (a) but for validation set data

As can be noticed, the RMSE values pertaining to the models trained on the noise-superimposed enlarged data sets are consistently smaller when compared with those obtained using the original non-noise-superimposed data sets. The extent of RMSE reduction is in general significant; in some cases it is as high as 99% (see Table 3.1 to Table 3.3).

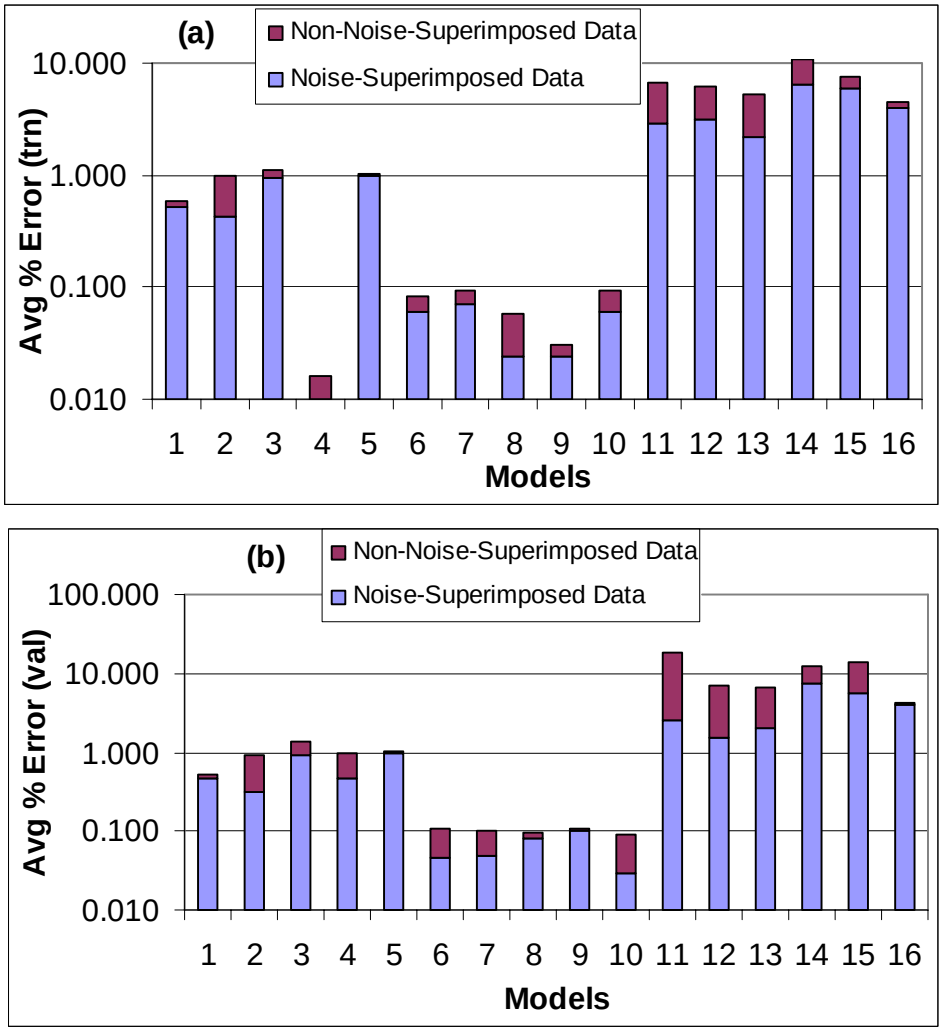

Figure 3.2: (Panel a) Comparison of average percentage error values corresponding to the softsensor models based on noise-superimposed and non-superimposed training data sets. (Panel b) same as panel (a) but for validation sets

Similar results are also observed with average percentage error for training and validation sets (see Figure 3.2*a* and Figure 3.2*b*). The bar chart comparing the R^2 values corresponding to the predictions of all sixteen softsensor models using noise-superimposed and non-noise-superimposed data are portrayed in

magnitudes when compared with the R^2 values pertaining to the non-noise-superimposed data.

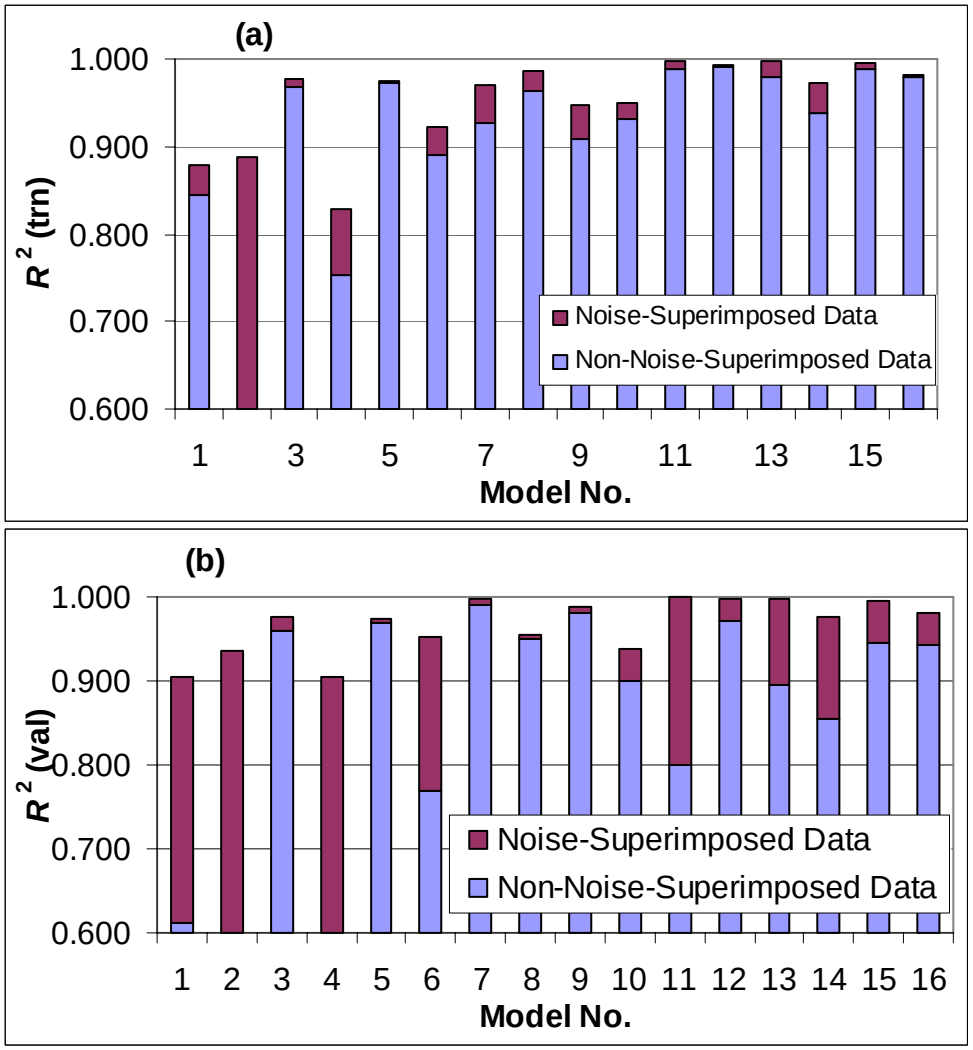

Figure 3.3: (Panel a) Comparison of the squared of correlation coefficient (R2) values corresponding to the soft-sensor models based on noise-superimposed and non-superimposed training sets. (Panel b) for validation data sets

From the R^2 magnitudes listed in Table 3.1 to Table 3.3, it is seen that the usage of noise-superimposed data has consistently yielded R^2 values greater than 0.9. High and comparable values of R^2 and low and comparable values of RMSE for both the training and validation sets indicate that the models trained on the noise-superimposed data possess excellent prediction accuracy and generalization performance. The above–described softsensor models have been installed on the

distributed control systems (DCS) of RIL's two PE plants and are yielding satisfactory real-time predictions of the three QC variables.

Table 3.1: Prediction and generalization performance of soft-sensor models for stress exponent

Data Set*	Training data set		Validation data set	
	RMSE (% Reduction)	**R^2 (% Improvement)**	**RMSE (% Reduction)**	**R^2 (% Improvement)**
Z_1	0.010	0.845	0.832	0.613
$\hat{Z}_1$	0.009 (12.34)	0.879 (4.15)	0.008 (99.07)	0.904 (47.50)
Z_2	0.016	0.438	0.015	0.46
$\hat{Z}_2$	0.007 (55.12)	0.888 (102)	0.005 (64.74)	0.935 (102)
Z_3	0.029	0.967	0.033	0.96
$\hat{Z}_3$	0.025 (14.25)	0.976 (0.92)	0.025 (24.04)	0.976 (1.64)
Z_4	0.707	0.753	0.016	0.494
$\hat{Z}_4$	0.046 (93.45)	0.828 (9.912)	0.015 (4.75)	0.904 (82.9)
Z_5	0.029	0.972	0.028	0.968
$\hat{Z}_5$	0.028 (5.15)	0.974 (0.228)	0.027 (2.17)	0.975 (0.65)

* Z_i : Non-noise-superimposed data set for *i*th stress exponent model;

$\hat{Z}_i$: Noise superimposed enlarged set for *i*th stress exponent model

Table 3.2: Prediction and generalization performance of soft-sensor models for density

Data Set*	Training data set		Validation data set	
	RMSE (% Reduction)	**R^2 (% Improvement)**	**RMSE (% Reduction)**	**R^2 (% Improvement)**
Z_1	0.068	0.891	0.100	0.769
$\hat{Z}_1$	0.052 (22.94)	0.922 (3.42)	0.046 (53.70)	0.952 (23.85)
Z_2	0.043	0.927	0.009	0.99
$\hat{Z}_2$	0.002 (94.59)	0.969 (4.53)	0.001 (93.36)	0.998 (0.834)
Z_3	0.100	0.964	0.061	0.95
$\hat{Z}_3$	0.003 (97.10)	0.987 (2.35)	0.001 (98.41)	0.956 (0.53)
Z_4	0.060	0.908	0.042	0.98
$\hat{Z}_4$	0.003 (95.26)	0.948 (4.36)	0.001 (96.90)	0.989 (0.89)
Z_5	0.050	0.931	0.082	0.90
$\hat{Z}_5$	0.041 (17.05)	0.951 (2.08)	0.0206 (74.88)	0.937 (4.04)

* Z_i : Non-noise-superimposed data set for *i*th density model;

$\hat{Z}_i$: Noise superimposed enlarged set for *i*th density model

Table 3.3: Prediction and generalization performance of soft-sensor models for MFI

Data Set*	Training data set		Validation data set	
	RMSE (% Reduction)	**R^2 (% Improvement)**	**RMSE (% Reduction)**	**R^2 (% Improvement)**
Z_1	0.109	0.988	0.493	0.801
$\hat{Z}_1$	0.036 (66.64)	0.999 (1.07)	0.033 (93.39)	0.998 (24.7)
Z_2	1.524	0.99	2.320	0.972
$\hat{Z}_2$	0.005 (99.70)	0.994 (0.40)	0.004 (99.84)	0.998 (2.65)
Z_3	0.041	0.98	0.117	0.895
$\hat{Z}_3$	0.017 (58.04)	0.997 (1.71)	0.016 (86.37)	0.997 (11.45)
Z_4	0.173	0.939	0.290	0.856
$\hat{Z}_4$	0.021 (87.86)	0.972 (3.54)	0.020 (92.97)	0.976 (14.08)
Z_5	1.954	0.988	4.142	0.944
$\hat{Z}_5$	1.316 (32.63)	0.995 (0.65)	1.222 (70.49)	0.995 (5.34)
Z_6	0.037	0.98	0.032	0.943
$\hat{Z}_6$	0.018 (51.05)	0.982 (0.20)	0.031 (4.19)	0.98 (3.96)

* Z_i : Non-noise-superimposed data set for *i*th MFI model;

$\hat{Z}_i$: Noise superimposed enlarged set for *i*th MFI model.

3.2.7 Benefits of the Soft-sensor Models

The model-based real-time knowledge of the three product quality variables has served following important purposes, namely: (i) assistance in reducing product variability, (ii) reduction in the frequency of product sampling and thereby lab analyses, and (iii) the plant management no longer has to wait for lab results while making a switch from an off-spec to prime quality production and vice versa.

To summarize, this study presents a case history of the development of ANN-based softsensor models for predicting the three critical QC variables namely stress exponent, density and melt flow index, of Reliance Industries Limited's two polyethylene plants at Hajira, India. A new generic strategy capable of substantially improving the prediction accuracy and generalization performance of the ANN models was specially designed and implemented for the development of the above-stated softsensors. This strategy envisages creation of optimal noise-superimposed data sets for training and validation of the softsensor models. The softsensors developed thereby are capable of predicting the values of the three QC variables with satisfactory accuracy. This has resulted in the reduction of product sampling frequency and consequent laboratory analyses, and also allowed the plant management in taking "informed" decisions while switching from an off-spec to prime production and vice-versa.

3.3 PERFORMANCE ENHANCEMENT OF ARTIFICIAL NEURAL NETWORK BASED MODELS IN PRESENCE OF NOISY DATA*

Owing to their significant nonlinear function approximation capability, ANNs have been widely used for developing nonlinear empirical process models. The major advantage of ANN-based models is that multiple input—multiple output (MIMO) nonlinear relationships can be developed easily and exclusively from the historic process data (known as "example set"). A significant difficulty arises in ANN-based modeling when the example input-output data contain unavoidable instrumental noise and/or measurement errors. In such situations, the resultant ANN model exhibits suboptimal prediction accuracy and poor generalization performance. Accordingly, a generic method [Kulkarni et al., 2002] is used in this section for improving the prediction accuracy and generalization performance of ANN models. The methodology envisages creation of an enlarged noise-superimposed data-set from the example set, which is then utilized in training the ANN model. In this "noise-superimposition" methodology, the Gaussian noise of a specific tolerance is added to each input-output element of the example data set. Typically, a large number of input-output patterns are sampled in this manner from the Gaussian probability distribution function. The underlying principle in the presented methodology is that the super-imposed noise in the data constrains the ANN to be less sensitive to variations in the input data and the resultant smoothing effect is beneficial in improving the ANN's generalization performance. It is necessary that the strength of the variable-specific superimposed noise is optimal. This issue has been addressed by using an optimization strategy that optimizes the input-output variable-specific tolerance values of the Gaussian super-imposed noise. The efficacy of the noise-superimposition formalism in developing optimal ANN models has been

---

* Badhe Y. P, S. S. Tambe and B. D. Kulkarni, Presented in the "*First Indo-US Joint Meeting in a Global Environment*," organized by Indian Institute of Chemical Engineers and American Institute of Chemical Engineers held at The Grand Hyatt, Mumbai, during 28-30 Dec. 2004.

successfully established by conducting two case studies involving modeling of a catalytic and CSTR process.

3.3.1 Introduction

Any real-world process data sets are always associated with instrumental noise and/or measurement errors. There exists a number of efficient methodologies for removing noise from the raw process data; these methodologies though effective for denoising dynamic data with non-varying statistical properties, they are not effective – owing to their varying statistical properties – for denoising, for instance, steady-state process data. Thus, the data in their raw noisy form must be used for building ANN models. The presence of noise and/or errors in the data used for training a steady-state ANN model creates a threshold limit for the prediction accuracy and generalization performance of the model. Also, the noise present in the desired (target) output values increases the danger of over-fitting of ANN models. This effect is more pronounced in regions where the function to be learned by the ANN model is steep. The inaccuracies in model predictions, if significant, cannot be tolerated since a significant number of control and policy decisions regarding the process operation are based on the predictions made by the model. Thus, it is critically important that an ANN model possesses not only excellent prediction accuracy, but also a good generalization property.

In one of the established methods for improving the generalization performance of an ANN model, noise is added to the inputs and outputs of the example set [Sietsma et al., 1991, Holmstorm et al., 1992, An, 1996]. Addition of noise serves the purpose of enlarging the size of the training set and is akin to minimizing the true error function. Sietsma and Dow [1991] showed that training with Gaussian noise-added data improves the classification ability of multi-layer perceptron (MLP) networks. In another exhaustive study, An [1996] studied the effect of noise in the inputs and weights and demonstrated that noise-addition is helpful in improving the generalization performance of an ANN model. A significant difficulty in this approach is determining the strength of the noise to be added to the individual elements of the example set. This difficulty arises from the fact that addition of very small noise amount results in changes of no consequence whereas high amount of noise clearly distorts the intrinsic relationship among the

input-output variables. Thus, it is absolutely essential to devise a formalism that fixes the amount of noise to be added while creating a noise-superimposed enlarged example set. Accordingly, the method presented in the following section utilizes a novel approach for improving the prediction accuracy and generalization performance of an ANN model when the example data contain instrumental noise and measurement errors. To be specific, the approach generates an enlarged example data set wherein multiple patterns are synthesized by adding an optimal amount of Gaussian noise to the elements of each input-output example pattern. The important issue of determining the optimal magnitude of the superimposed noise has been addressed by using the genetic algorithm-based nonlinear optimization strategy. The said method describedbriefly in section 3.2.6 is elabroted in the following section. The efficacy of the optimal noise superimposition technique has been validated by conducting two case studies, namely: (i) ANN-based steady-state modeling of a non-isothermal continuous stirred tank reactor (CSTR) wherein a consecutive $A \rightarrow B \rightarrow C$ reaction occurs, and (ii) ANN based modeling of benzene isopropylation reaction on Hbeta catalyst.

3.3.2 GA-Based Generation of Enlarged Noise Superimposed Data

The enlarged noise superimposed input-output data set to be used in the ANN modeling possesses following characteristics: (i) it is created by generating multiple noise-superimposed samples from each input-output pattern of the example input-output set, (ii) the superimposed noise is normally distributed and the magnitude of noise is specific to individual input-output variables of the example set data, (iii) the optimal magnitude of the variable-specific noise is determined by the genetic algorithm (GA) method, and (iv) the enlarged data set after appropriate partitioning into the training and validation sets yields an ANN model possessing improved prediction and generalization performance.

Consider an example data set, D, comprising input-output vector pairs, (x_1, y_1), (x_2, y_2),…,(x_p, y_p),…,(x_P, y_P) such that $x_p \in \Re^N$; p = 1,2,…, P is a vector of model's input variables and $y_p \in \Re^K$ is a vector of the corresponding model output variables. The relationships between input vectors, x_p; p = 1, 2,…, P, and the corresponding output vector, y_p, are governed by a K-dimensional nonlinear

function vector, *f*. A feed-forward neural network, such as the most widely employed three-layered multilayer perceptron (MLP), approximates the nonlinear relationships between x_p and y_p as given by:

$$y_p = f(x_p, W^H, W^O) \tag{3.1}$$

where, matrices W^H and W^O, represent the weights on connections between MLP's input and hidden layer nodes and, hidden and output layer nodes, respectively. For improving the prediction accuracy and generalization performance of the ANN model, we create noise-superimposed enlarged version, $\hat{D}$, of the data matrix D according to the method proposed by Kulkarni et al. [2002]. In a system where the relationship between its inputs and outputs is nonlinear, the dependent (output) variables exhibit varying extents of sensitivity to the changes in the causal (input) variables. Thus it is advisable to add varying extent of noise to each individual input-output variable rather than keeping the noise magnitude same for all the stated variables. In the present study, Gaussian noise has been considered for generating the enlarged data set. The amount of Gaussian noise to be superimposed is specific to an input-output variable and is characterized in terms of the tolerance percentage. The genetic algorithms formalism is used to search for the optimal input and output noise tolerance vectors, $\varepsilon^{I*} = [\varepsilon_1^{I*}, \varepsilon_2^{I*}, \ldots, \varepsilon_n^{I*}, \ldots, \varepsilon_N^{I*}]^T$ and $\varepsilon^{O*} = [\varepsilon_1^{O*}, \varepsilon_2^{O*}, \ldots, \varepsilon_k^{O*}, \ldots, \varepsilon_K^{O*}]^T$, where the input and output noise tolerances are defined in terms of the Gaussian probability density function (PDF) as given below in Eqs. (3.2) and (3.3), respectively.

$$\varepsilon_n^I = (3.09 \times 100) \times (\sigma_{pn}^I / x_{pn}), \qquad n = 1, 2, 3 \ldots, N \tag{3.2}$$

where, x_{pn} denotes *n*th element of the *p*th row of the input variable matrix, *X*: ε_n^I represents the noise tolerance for x_{pn} and σ_{pn}^I refers to the standard deviation of the Gaussian PDF.

$$\varepsilon_k^O = (3.09 \times 100) \times (\sigma_{kn}^O / y_{pk}), \qquad k = 1, 2, 3 \ldots, K \tag{3.3}$$

where, y_{pk} denotes the *k*th element of the *p*th row of output variable matrix, *Y*, ε_k^O represents the noise tolerance for the output element, y_{pk} and σ_{pk}^O refers to the standard deviation of the Gaussian PDF. The genetic algorithm steps involved

in searching the optimal values of the input-output noise tolerances ($\varepsilon^{I^*}, \varepsilon^{O^*}$) are as given below:

Step 1: Initialize a random population of noise tolerance vectors ($\varepsilon^{I^*}, \varepsilon^{O^*}$) representing candidate solutions to the nonlinear optimization problem wherein individual elements of a solution vector describe the extent of noise to be added to each input-output element of the example set.

Step 2: Utilize the input-output variable-specific noise tolerance values contained in a candidate solution vector to define the respective Gaussian probability distribution (PDF) and use the random numbers sampled from each PDF to generate multiple noise-superimposed sample input-output patterns corresponding to each input-output data pattern in the example set.

Step 3: Repeat step (2) with all candidate solution vectors and generate noise-superimposed enlarged sample input-output data sets equaling the number of candidate solutions.

Step 4: Develop ANN models using the noise-superimposed enlarged data sets created in step 3.

Step 5: Calculate fitness values of the candidate solutions describing noise tolerances and rank the solutions in the decreasing order of their fitness values.

Step 6: Perform GA operations, namely, *selection*, *crossover* and *mutation* on the ranked solution population to obtain a new population of candidate solutions representing noise tolerances.

Step 7: Repeat steps (2) to (6) till an optimal solution representing optimum values of the input-output variable specific noise tolerances leading to the ANN model with improved prediction and generalization performance is obtained.

3.3.3 Case Study – I: Steady-State Modeling of A CSTR

This case study considers an ANN-based steady-state modeling of a jacketed non-isothermal CSTR wherein two first order reactions in series, $A \rightarrow B \rightarrow C$, take place. The phenomenological model defining the CSTR dynamics represented in terms of three state variables, T (temperature), $\hat{C}_A$

(concentration of species, *A*) and $\hat{C}_B$ (concentration of species, *B*) is as given in Nandi et al., [2001]. The phenomenological model was used to generate the steady-state process data under varying values of six process parameters, viz., *V* (volume, m^3), *F* (inlet flow rate, m^3/min), *Q* (heat removal rate, KJ/min), C_A^0 (inlet concentration of species, *A*), C_B^0 (inlet concentration of species, *B*) and T^0 (inlet temperature, ^oC). The values of these parameters were respectively varied in the following ranges: [0.1-0.5], [0.5-1.0], [20000-200000], [600-4000], [6-550] and [295-305] to create an example data set of 50 patterns. These values form the six-dimensional input space of the ANN model, and the corresponding steady-state concentration of the desired product, *B* (C_B, mol/lit), forms the model output. To mimic real-world data comprising process noise, the Gaussian noise of strengths 5%, 10%, and 15% was added to each element of the simulated process data set. The three data sets thus formed are defined as D_1, D_2 and D_3, respectively. For comparing the performance of the optimal noise-superimposition strategy, ANN models were constructed using the data sets D_1, D_2 and D_3 directly (i.e., without employing optimal noise-superimposition strategy). The details of the network architecture and EBP-specific parameter values resulting in the optimal MLP models created using the three data sets are listed in Table 3.4. Also listed in the table are the values of the coefficient of correlation (CC) between the ANN model-predicted and target network outputs, average prediction error (%) and the RMSE corresponding to the training (80% data) and validation (20% data) sets.

In the next set of modeling simulations, the three data sets, D_1, D_2 and D_3, were enlarged ten times using the optimal noise superimposition formalism described earlier. The optimal noise tolerance values corresponding to the CSTR input-output variables obtained using the GA formalism, are listed in Table 3.5. The prediction and generalization performance of the three optimal MLP models constructed using as many noise-superimposed enlarged data sets (referred to as $\hat{D}_1$, $\hat{D}_2$ and $\hat{D}_3$, respectively) separately, was evaluated in terms of correlation coefficient, average error (%) and RMSE values and these are listed in Table 3.4. This table also gives architectural details of the optimal MLP models and the values of the EBP-specific parameters used in creating the models. It is clearly

observed from the comparison of the statistical values listed in Table 3.4 that the MLP models trained on the noise-superimposed CSTR data have consistently resulted in lower RMSE values for both the training and validation data. Using noise-superimposed training data sets, the training set RMSE (E_{trn}) values have decreased substantially i.e., by 86.1% to 96.6%. Similar reduction was also observed in the validation set RMSE values (E_{val}). Additionally the average error (%) values exhibit a significant reduction ranging between 78.5% and 97.7%, for the training sets and 94.4% to 97.9% for the validation sets. It is noticed that the correlation coefficient values pertaining to the output predictions made by the optimal MLP models using the noise-superimposed data are very high (≥ 0.999). It can thus be seen that the usage of noise-superimposed data for training an MLP model has led to smaller RMSE and average error (%) values, and higher CC values (indicating better prediction accuracy and generalization performance) when compared with the MLP models trained on the non-noise-superimposed data.

3.3.4 Case Study – II: Modeling of Benzene Isopropylation over Hbeta Catalyst

This case study considers ANN-based steady-state modeling of the pilot-plant scale reactor for the Hbeta catalyzed benzene isopropylation process. In this process, though the formation of cumene via isopropylation of benzene is the main reaction, a series of other components are also produced via side reactions.

Isopropylation of benzene is an important alkylation reaction in the petrochemical industry for the synthesis of cumene, which is the chief starting material in phenol production. In the last decade, several modifications of the zeolite beta were explored as potential catalysts in cumene synthesis [Perego, et al., 1994; Cavani, et al., 1997; Geatti, et al., 1997; Meima, 1998]. More recently, Sridevi et al. [2001] investigated isopropylation of benzene over Hbeta (protonic form of beta catalyst). Beta is a crystalline alumino-silicate catalyst with high silica content and its important characteristic is that it is the only large pore zeolite with chiral pore intersections. It consists of 12-membered rings interconnected by cages formed by intersecting channels. The linear channels have pore opening dimensions of $5.7\ \overset{0}{\text{A}} \times 7.5\ \overset{0}{\text{A}}$, whereas the tortuous channels with intersections of

two linear channels have approximate dimensions of 5.6 $\overset{0}{A} \times 6.5 \overset{0}{A}$. The catalyst has a pore volume of ≈ 0.2 cm^3/g. In the study by Sridevi et al. [2001], a phenomenological model for benzene isopropylation reaction was developed based on the isopropyl alcohol conversion in a continuous down-flow differential packed bed reactor taking into account the secondary reactions such as the dehydration of alcohol. This model however was restricted to the lower conversion (< 30%) of the limiting reactant, i.e., isopropyl alcohol, wherein heat and mass transfer resistances in the differential bed were assumed to be negligible. For maximizing yield and selectivity of cumene in the vapor phase alkylation of benzene with isopropyl alcohol over Hbeta catalyst, experiments were also conducted in a pilot plant scale reactor. Isopropylation of benzene involves a main reaction producing cumene and multiple side reactions as described below:

Main reaction :

Benzene + Isopropyl Alcohol → Cumene + Water (benzene alkylation)

Secondary reactions :

Cumene + Isopropyl Alcohol → p-Di-isopropyl Benzene + Water (cumene alkylation)

p-Di-isopropyl Benzene → m-Di-isopropyl Benzene (isomerization)

2 Isopropyl alcohol → Di-isopropyl ether + Water (alcohol dehydration)

Owing to the complex nonlinear nature of the benzene isopropylation process, an ANN was chosen for developing the steady-state process model. Accordingly, four reactor operating variables namely, reaction temperature (x_1), pressure (x_2), benzene to isopropyl alcohol mole ratio (x_3) and weight hourly space velocity (WHSV) (x_4), form the input space of the ANN-based model. The Cumene yield and selectivity defined as y_1 and y_2, respectively, are the model outputs and these are evaluated as:

$$y_1 = \frac{100 \times \text{weight of cumene formed per unit time}}{\text{weight of isopropyl alcohol fed per unit time}} \quad (3.4)$$

$$y_2 = \frac{100 \times \text{weight of cumene formed per unit time}}{\text{weight of total aromatics produced per unit time}} \quad (3.5)$$

The experimental data pertaining to the 42 pilot plant experiments [Nandi et al., 2004] were considered in the ANN modeling. The experiments studied the effect of varying values of the four operating variables (x_1 to x_4) on the cumene yield and selectivity. The experimental data sets comprising values of operating variables and the corresponding values of cumene yield and selectivity are designated as D_4, respectively. First, two optimal ANN models were developed using these data sets. The details of the network architectures, EBP-specific parameter values resulting in the optimal MLP models and the results in terms of coefficient of correlation (CC) between the model predicted and target network outputs, average prediction error (%) and the RMSE corresponding to the training and validation sets are listed in Table 3.4. Next, the data sets D_4 were enlarged using the optimal noise-superimposition strategy and the respective enlarged data sets ($\hat{D}_4$) were used to create two ANN models. The optimal noise tolerance values (ε^{I*} and ε^{O*}), corresponding to the four input and two output variables obtained using the GA formalism are listed in Table 3.6.

A comparison of the prediction and generalization performance of the yield and selectivity models trained using noise-superimposed and non-noise-superimposed data sets reveals that the usage of noise-superimposed enlarged training data sets has resulted in decreasing the training set RMSE (E_{trn}) values by 16.3% and 39.6%. A similar significant reduction was also observed in the validation set RMSE values (E_{val}). Also, the average error (%) values exhibit a significant reduction of nearly 55% for the training sets and 48.5% to 57.2% for the validation sets. Likewise case study-I, the smaller and comparable values of RMSE and average error (%) and high values of CC for both training and validation sets clearly indicate enhanced prediction accuracy and improved generalization performance of the MLP models trained on the optimal noise-superimposed data.

3.3.5 Concluding Remarks

The case studies described above present results of a generic methodology to develop artificial neural network based process models possessing excellent prediction accuracy and generalization performance. The method creates an enlarged noise-superimposed data set from the original noisy process data set, wherein the optimal magnitude of the input-output variable-specific superimposed noise is determined using a stochastic optimization strategy namely, genetic algorithm. Utilization of noise-superimposed data helps in fitting a smooth function leading to an improved prediction accuracy and generalization performance of the ANN model. The efficacy of the optimal noise-superimposition methodology has been demonstrated successfully by conducting ANN-based modeling of two chemical processes.

Table 3.4: Comparison of predictions and generalization performance of ANN-models using noise-superimposed and non-noise-superimposed data

Data Set*	MLP Parameters (N_{H1},N_{H2},η)#	Training Data Set			Validation Data Set		
		Correl. Coef.	RMSE (E_{trn}) (%improvement)	Average % Error (%improvement)	Correl. Coef.	RMSE (E_{val}) (%improvement)	Average % Error (%improvement)
D_1	6:4:0.25	0.999	10.93	3.93	0.999	11.02	10.58
$\hat{D}_1$	6:5:0.5	0.999	1.51 (86.1)	0.84 (78.5)	0.999	1.43 (87.0)	0.59 (94.4)
D_2	6:5:0.31	0.998	12.91	4.89	0.996	15.89	11.73
$\hat{D}_2$	6:6:0.7	0.999	1.22 (90.5)	0.36 (92.7)	0.999	1.738(89.1)	0.43 (96.3)
D_3	6:5:0.53	0.986	28.81	15.87	0.956	55.73	11.16
$\hat{D}_3$	6:6:0.7	0.999	0.98 (96.6)	0.36 (97.7)	0.999	1.40 (97.5)	0.23 (97.9)
D_4 (yield)	5:0:0.7	0.998	0.49	41.81	0.999	0.44	36.34
D_4 (select.)		0.974	4.64	4.63	0.976	4.68	4.67
$\hat{D}_4$ (yield)	5:4:0.7	0.999	0.41(16.3)	18.73(55.2)	0.999	0.31(29.5)	18.71(48.5)
$\hat{D}_4$ (select.)		0.999	2.80(39.6)	2.05(55.7)	0.999	2.79(40.4)	2.0(57.2)

N_{H1} = number of neurons in the first hidden layer; N_{H2} = number of neurons in the second hidden layer; η = learning rate.

* $\hat{D}_i$, i = 1, 2, 3 represents noise-superimposed enlarged data set.

Table 3.5: Optimal noise tolerance (%) values for CSTR variables

Data Set	**Optimal Noise Tolerances**						
	ε_1^{I*}	ε_2^{I*}	ε_3^{I*}	ε_4^{I*}	ε_5^{I*}	ε_6^{I*}	ε_1^{O*}
$\hat{D}_1$	0.679	0.592	0.324	0.263	0.489	0.109	0.678
$\hat{D}_2$	0.593	0.78	0.566	0.283	0.656	0.542	0.998
$\hat{D}_3$	0.607	0.395	0.169	0.001	0.947	0.615	0.458

Table 3.6: Optimal noise tolerance (%) values for benzene isopropylation process variables

Data Set	**Optimal Noise Tolerances**					
	ε_1^{I*}	ε_2^{I*}	ε_3^{I*}	ε_4^{I*}	ε_1^{O*}	ε_2^{O*}
$\hat{D}_4$ & $\hat{D}_5$	0.026	0.634	0.034	0.537	0.521 ($\hat{D}_4$)	0.982 ($\hat{D}_5$)

3.4 ESTIMATION OF GROSS CALORIFIC VALUE OF COALS USING ARTIFICIAL NEURAL NETWORKS*

The gross calorific value (GCV) is an important property defining the energy content and thereby efficiency of fuels, such as coals. There exists a number of correlations for estimating the GCV of a coal sample based upon its proximate and/or ultimate analyses. These correlations are mainly linear in character although there are indications that the relationship between the GCV and a few constituents of the proximate and ultimate analyses could be nonlinear. Accordingly, in this study a total of seven nonlinear models have been developed using the artificial neural networks methodology for the estimation of GCV with a special focus on Indian coals. The comprehensive ANN model developed here uses all the major constituents of the proximate and ultimate analyses as inputs while the remaining six sub-models use different combinations of the constituents of the stated analyses. It has been found that the GCV prediction accuracy of all the models is excellent with the comprehensive model being the most accurate GCV predictor. Also, the performance of the ANN models has been found to be consistently better than that of their linear counterparts. Additionally, a sensitivity analysis of the comprehensive ANN model has been performed to identify the important model inputs, which significantly affect the GCV. The ANN-based modeling approach illustrated in this study is sufficiently general and thus can be gainfully extended for estimating the GCV of a wide spectrum of solid, liquid and gaseous fuels.

3.4.1 Introduction

The abundance and versatility of coals makes them an important source of energy for the present and future. The chemical composition of coals is characterized

---

* Patel S. U., B. Jeevan Kumar, Y. P. Badhe, B. K. Sharma, S. Saha, S. Biswas, A. Chaudhury, S. S. Tambe and B. D. Kulkarni, *Fuel*, 86 (3), February 2007, 334–344.

in terms of their proximate and ultimate analyses. While the proximate analysis determines the contents of moisture, volatile matter, ash, and fixed carbon, the ultimate analysis measures the content of various elements, namely carbon, hydrogen, nitrogen, sulphur, and oxygen. An important property, which indicates the useful energy content of a coal and thereby its value as a fuel, is its *calorific value* (also known as *heat of combustion*), which is defined as the amount of heat evolved when a unit weight of the fuel is burnt completely and the combustion products cooled to a standard temperature of 298^0 K. It is usually expressed as the GCV (also termed *Higher Heating Value*, HHV). The magnitude of GCV varies significantly depending on the ash and moisture contents and the type of coal. It is determined using various basis, such as "dry mineral matter free (DMF)", "as received", and "dry" basis. Among these, the DMF basis is useful for scientific evaluation and classification of coals while in commercial applications calorific values are commonly determined using *as received* or *dry* basis. The GCV of a coal sample is measured experimentally using a Bomb calorimeter. Since GCV is a major indicator of the quality of coal, a number of linear correlations have been developed for its prediction on the basis of proximate and/or ultimate analyses. The advantages of the GCV correlations are [Parikh et al., 2005]: (i) they provide an easy and quick means for estimating the GCV thus saving the efforts involved in its experimental determination, (ii) application in the performance modeling exercise of combustion, gasification and pyrolysis processes involving coal, and (iii) facility of using the GCV as an algebraic expression in terms of fuel constituents, which in turn is useful in studying the influence of the proximate as well as ultimate analysis of a fuel on the process performance. Accordingly, this study first reviews various linear correlations proposed for the GCV estimation of coals and discusses the need of developing their nonlinear counterparts. Next, a number of nonlinear correlations (models) based on ANNs have been introduced in this study for the estimation of GCV with a special focus on Indian coals. The ANN-based GCV models presented here are found to possess an excellent GCV prediction accuracy and outperform the existing linear models.

3.4.2 Survey of GCV Correlations and Need for ANN-Based Models

There essentially exist two types of correlations for estimating the GCV of coals. The type–I correlations are meant exclusively for coals, while type-II correlations additionally cover a number of solid, liquid and gaseous fuels. Both types of correlations are based on the proximate and/or ultimate analyses. An important coal-specific correlation was proposed by Goutal et al., [1902] that correlated the calorific value (Q) with the volatiles and fixed carbon as given by:

$$Q = 82 \times F_C + k \times V_m \tag{3.6}$$

where, F_C and V_m denote the percentages of fixed carbon and volatiles, respectively (on the "dry ash free", DAF, basis), and k is a constant that depends on the V_m. Subsequently, a widely used GCV correlation as given below was proposed by Schuster et al. [1951],

$$Q = 8000 + V_m \times (70 - 1.65 \times V_m), \quad \text{(cal/gm)} \tag{3.7}$$

where, Q and V_m are determined on the DAF basis. Another correlation for the GCV, proposed by Spooner [1951], is given as:

$$Q = 8781 + 19 \times V_m - 144 \times C_O, \quad \text{(cal/gm)} \tag{3.8}$$

where, C_O refers to the oxygen content in coal. For estimating the GCV of Indian coals, Mazumdar [1954] proposed following expression,

$$Q = 9170 - 16 \times V_m - 60 \times C_M \times (1 - 0.001 \times C_M), \quad \text{(cal/gm)} \tag{3.9}$$

where, C_M denotes the percentage of moisture. More recently, Mazumdar [2000] proposed an expression for the determination of calorific value of coal (MJ/kg) as given by:

$$Q = 13.03 \times C_O + (100 - C_{MM}) \times \left(\frac{0.238 \times C_H}{C_C - 0.0065} \right) - 0.007 \times C_A, \ \text{(MJ/kg)} \tag{3.10}$$

where, C_{MM} is the mineral matter on dry basis and C_H, C_C and C_O respectively refer to the percentages of hydrogen and carbon, and the theoretical oxygen requirement for the complete coal combustion on the dry basis. The C_{MM} value is calculated using the Parr formula i.e., $C_{MM} = 1.08 \times C_A + 0.558 \times C_S$, where C_A and C_S refer to the percentages of ash and sulphur, respectively. For low sulfur coals, $C_{MM} = 1.1 \times C_A + 0.55 \times C_S$.

The energy content of an Indian coal (non-coking) is commonly expressed in terms of "Useful Heating Value (UHV)". In industrial applications, UHV (Q_U) is used for grading and pricing of coals and a simple empirical equation as given below was developed for its estimation by the Central Fuel Research Institute (CFRI), Dhanbad, India;

$$Q_U = 8900 - 138 \times (C_A + C_M), \qquad \text{(kcal/kg)} \tag{3.11}$$

where C_A and C_M refer to the percentages of ash and moisture at 60% relative humidity at 40^0 C. The usage of Eq. (3.11) began in 1979 and is still in vogue for grading non-coking Indian coals. An improvement in Eq. (3.11) has become necessary owing to its suboptimal performance in predicting the UHV values since the ash content in the coal being mined currently in India has increased significantly. Prior to 1979, approximately 85% of Indian coal was mined via underground mining operations and the average ash percentage in the coal was in the range of 25 % to 30 %. In contrast, a major portion (≈ 70 %) of the coal being mined presently (mainly via open-cast mining) has an average ash percentage of 45% or more. In Eq. (3.11), the same weightage of 138 is assigned to the moisture and ash contents and thus it is not valid for currently mined coals comprising high ash. For instance, according to Eq. (3.11), the UHV when ($C_A + C_M$) equals 64.5% is zero, while the actual GCV value is of the order of 2300 kcal/kg. In view of this fact, the applicability of Eq. (3.11) for the currently mined Indian coals has become questionable. It may also be noted that internationally, grading and pricing of coals is done in terms of their GCV and not

UHV. Thus, Chaudhury and Biswas [Choudhary and Biswas, 2002-03] proposed a correlation, which is a modified form of Eq. (3.11), for the estimation of GCV:

$$Q = a + \left(b \times Q_U\right), \qquad \text{(kcal/kg)} \tag{3.12}$$

where, a and b are constants whose magnitudes vary depending upon the geographical origin of the coal. The magnitudes of a and b for computing the GCV of coals from six prominent regions in India are: (i) MCL: 2043, 0.673, (ii) CCL: 2062, 0.6913, (iii) NCL: 2227, 0.6326, (iv) WCL: 2557, 0.55, (v) SECL: 2173, 0.69, and (vi) SCCL: 2290, 0.639, where MCL, CCL, NCL, WCL, SECL, and SCCL, respectively refer to Mahanadi, Central, Northern, Western, South-eastern, and Singareni coalfields.

Channiwala et al. [2002] have reviewed a large number of type-II correlations for estimating GCV of a number of solid, liquid and gaseous fuels and also proposed a unified correlation from the elemental analysis of fuels as given by:

$$\begin{aligned} Q = {} & 0.3491 \times C_C + 1.1783 \times C_H + 0.1055 \times C_S \\ & - 0.1034 \times C_O - 0.0151 \times C_N - 0.0211 \times C_A \end{aligned} \qquad \text{(MJ/kg)} \tag{3.13}$$

where C_C, C_H, C_N, and C_S denote the percentages of carbon, hydrogen, nitrogen and sulphur, respectively. The ranges of the mass percentage values (on dry basis) over which the correlation is valid are: $0\% \le C_C \le 92.29\%$, $0.43\% \le C_H \le 25.15\%$, $0\% \le C_O \le 50\%$, $0\% \le C_N \le 5.6\%$, $0\% \le C_S \le 94.08\%$, $0\% \le C_A \le 71.49\%$, and 4.475 MJ/kg $\le Q \le$ 55.345 MJ/kg. Equation (3.13) could predict the GCV of various types of fuels with an average absolute error of 1.45%. Although useful, a major difficulty with this correlation is that it requires values from the elemental analysis of coals that needs a costly equipment. Thus, Parikh et al. [2005] developed a proximate analysis based correlation for predicting the GCV of an entire spectrum of solid carbonaceous materials such as coals, lignite, all types of biomass materials, and char to residue-derived fuels. Their correlation is given as:

$$Q = 0.3536 \times F_C + 0.1559 \times V_m - 0.0078 \times C_A, \qquad \text{(MJ/kg)} \tag{3.14}$$

In addition to above-described type I and II correlations, relationships have also been proposed for computing the GCV exclusively of solid fuels [Kucukbayrak et al.,1991; Cordero et al., 2001; Fernandez et al., 1997; Demirbas, 1997; Jimenez, et al., 1991; Raveendran, 1996]. These correlations have restricted applicability since they are limited to a type or region of a fuel.

All the correlations discussed above assume linear relationships between the GCV and the constituents of the proximate and/or ultimate analyses. In order to verify the appropriateness of the linear relationships, we considered data from the proximate and ultimate analyses of a large number of coals (see Table 3.7) mined from different regions in India. Using these data, cross-plots were generated by plotting the individual constituents of the proximate and ultimate analyses against the corresponding GCVs. These plots are shown in Figure 3.4 wherein the eight panels (*a* to *h*) show the plots of GCV versus the percentages of ash, fixed carbon, hydrogen, carbon, oxygen, volatile matter, moisture and nitrogen, respectively. It is seen in these plots that there exists a clear linear dependence between the GCV and percentages of ash, fixed carbon, hydrogen, carbon, oxygen and volatile matter. However, a significant scatter is seen in the cross-plots of GCV versus percentages of sulphur, moisture and nitrogen. Thus, there exists a strong possibility that the GCV is nonlinearly correlated with C_S, C_M and C_N.

If such nonlinear relationships indeed exist, then these can be captured effectively by developing nonlinear models for the GCV estimation, which are likely to be more accurate than the linear correlations described earlier. Owing to the multiple causal factors (constituents of proximate and/or ultimate analyses), these nonlinear correlations would have a multiple input-single output (MISO) structure. There exist powerful nonlinear function optimization methods such as Marquardt's algorithm [Marquardt, 1963], to fit nonlinear relationships existing between MISO data. Given a nonlinear fitting function, the Marquardt's algorithm can efficiently estimate the parameters of a data-fitting function. However, a significant difficulty in this approach is choosing an appropriate MISO type nonlinear data-fitting function from an infinite number of functions that form the solution space of the MISO data-

fitting problem. Thus, an exhaustive trial-and-error procedure would be necessary to arrive at the correct nonlinear MISO model for the estimation of GCV. The artificial intelligence (AI) based nonlinear modeling formalism, namely ANNs overcomes the stated difficulty of choosing the correct form of the nonlinear data-fitting function. Specifically, in the ANN-based modeling it is not necessary to exclusively specify a system-specific nonlinear form of the data-fitting function and thus the exhaustive heuristic involved in choosing an appropriate data-fitting function is completely avoided. Accordingly, in this study, a number of ANN-based nonlinear models have been developed for the estimation of GCV of Indian coals. The main ANN-model in this study uses all the major constituents of the proximate and ultimate analyses as model inputs. In real practice, information on all the constituents of the said analyses may not be available and therefore a number of sub-models have also been developed using ANNs by considering various combinations of the constituents of the proximate and / or ultimate analysis as model inputs.

3.4.3 ANN-Based Models for GCV Estimation

The inputs used in developing the seven ANN-based models for estimating the GCV of Indian coals are listed in Table 3.8. Currently, there does not exist a comprehensive GCV model that uses information of all the major constituents of proximate and ultimate analyses. Thus, a comprehensive ANN model (I) has been developed that uses a total of ten inputs comprising major constituents of the proximate and ultimate analyses as also the He-density. Wherever feasible, the GCV estimation performance of an ANN model has been compared with its linear counterpart. Accordingly, prediction and generalization performance of ANN models II, VI and VII was compared with that of Eqs. (3.12), (3.13) and (3.14), respectively.

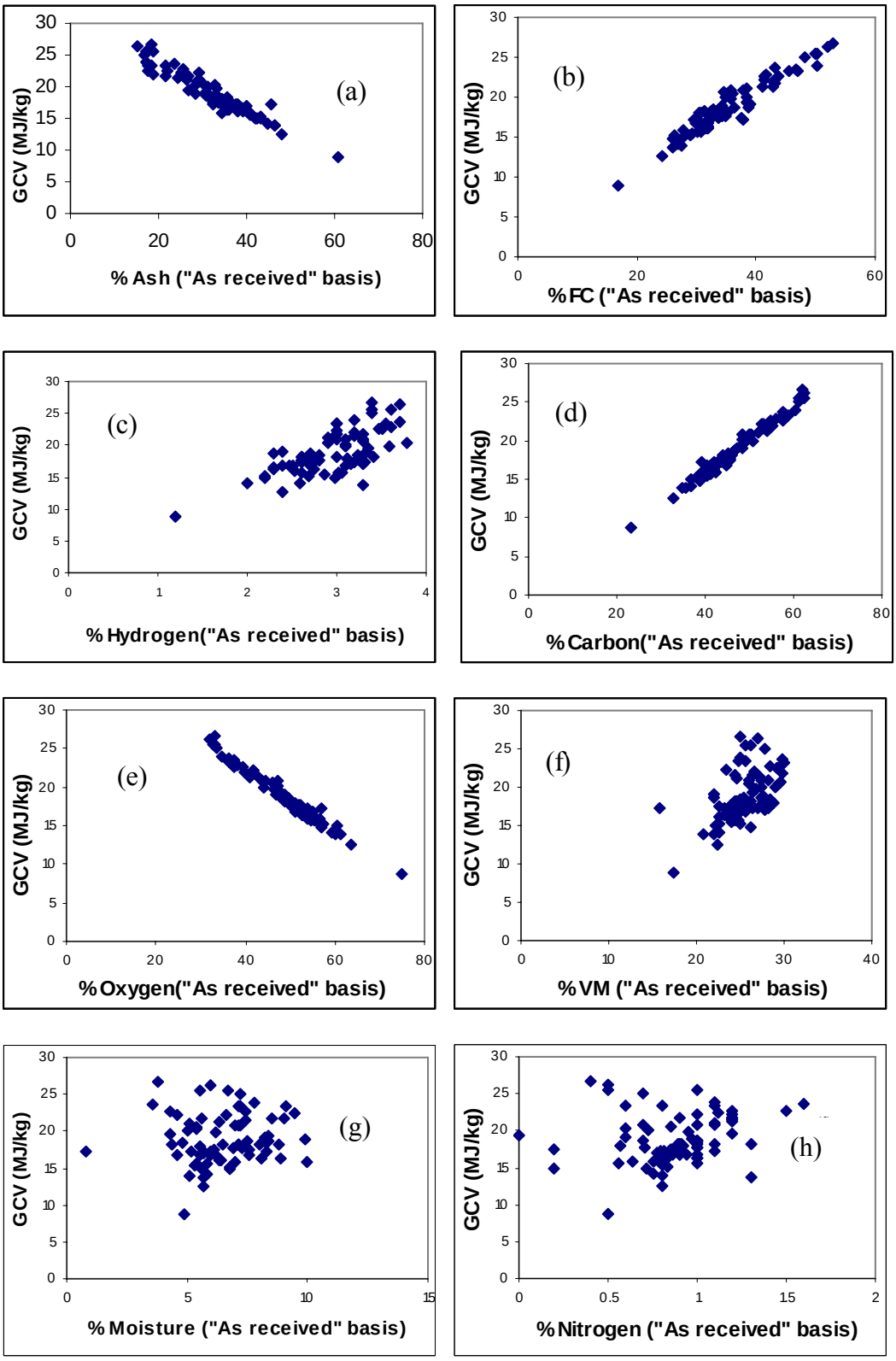

Figure 3.4: Cross-plots of GCV verses individual constituents of proximate and ultimate analyses

The justification for various combinations of inputs used in the six ANN-based sub-models is given below.

- Conducting proximate and ultimate analyses on "as received" basis is easier since it requires no preprocessing of the coal sample. Thus, a majority of ANN-based models employ data on *as received* basis.
- Since proximate analysis is easier to perform than the ultimate analysis, four models (II, III, IV and VII) have been developed using various constituents of the proximate analysis.
- Unavoidable presence of ash adversely affects the GCV of coals and, therefore, the ash percentage has been considered as an input in five out of the six sub-models.
- Similar to ash, presence of moisture acts as a diluent of the heating value of coals and therefore models II to IV use moisture as one of their inputs.
- Percentage of carbon has a direct effect on the GCV, i.e. an higher carbon content results in an higher heating value. Thus, models V and VI use carbon percentage in their input space.
- The organic and inorganic matter in the coal influences the Helium (He) density of coal. The density is an important indicator of coal's open pore structure that determines the extent to which reactants diffuse inside the coal's interiors thereby affecting its heating value. Thus, He-density has been considered as an input to model-III.
- The percentages of the proximate analysis constituents such as ash and moisture, are in turn dependent (albeit in a complex manner) upon the elemental composition of coals. Thus, the input space of models V and VI comprise constituents of the elemental analysis.

3.4.4 Collection of Data

The data set (see Table 3.7) comprising constituents of the proximate and ultimate analyses as also the corresponding experimentally determined GCVs (Kcal/kg) of 79

coal samples obtained from the mines located in the six prominent coal producing regions in India and supplied by the CFRI, Dhanbad, India, has been used for the purpose of developing ANN-based models. The data consists of values of ten constituents of the coal analysis namely, moisture (C_M), ash (C_A), volatile matter (V_m), fixed carbon (F_C), carbon (C_C), hydrogen (C_H), sulphur (C_S), nitrogen (C_N), oxygen (C_O) and He-density (ρ_{He}). These values were determined on "as received" basis, which can be converted to the "dry basis" by using the following expression.

$$Value(\text{"dry" basis}) = Value(\text{"as received" basis}) / (100 - C_M)) \times 100 \qquad (3.15)$$

Among the seven ANN-based models, the first five use data on "as received" basis while the remaining two (VI and VII) use data on the "dry basis".

In this study, all ANN models have been developed by considering two hidden layers in the MLP architecture and these were trained using the EBP algorithm. In order to ensure that the ANN models possess the much desired generalization ability, the data on coal analysis and the corresponding GCVs were partitioned into two sets namely *training* and *test* sets. While the training set was used in the EBP algorithm-based iterative minimization of RMSE, the test set was used after each training iteration for assessing the generalization ability of the MLP model. The network weights that resulted in the least RMSE for the test set (E_{tst}) were considered to be the optimal weights. Before partitioning the available data into the stated two sets, the values of individual inputs and the output (GCV) were scaled to lie between 0.05 and 0.95. All ANN models use the logistic sigmoid transfer function for computing the outputs of the hidden and output layer nodes. It may be noted that the nonlinear function approximation capability of ANNs stems from the usage of nonlinear transfer function such as the logistic sigmoid. For constructing an MLP model with optimal prediction accuracy and generalization performance, it is necessary to rigorously study the effect of MLP's structural parameters namely, the number of nodes in the first and second hidden layers (J, K) as also two EBP-specific parameters viz. η and μ. Accordingly, the values of J, K, η and μ were varied

systematically and those leading to the minimum test set error were considered optimal. In addition, the effect of random weight initialization in the EBP algorithm was studied by varying the seed values of the pseudo-random number generator to obtain an MLP model that corresponds to the global or the deepest local minimum on the model's nonlinear error surface. The details of the training and test sets along with the optimized values of the MLP's architecture and EBP parameters for the seven ANN models are given in Table 3.8.

3.4.5 Results and Discussion

The GCV prediction and generalization performance of ANN-based models is given in Table 3.9. Their performance is evaluated in terms of: (i) coefficient of correlation (CC) between the model predicted and the corresponding experimental (target) GCVs, (ii) RMSE, and (iii) average percent error (APE) in the GCV prediction. These quantities have been evaluated separately for the training and test sets. While the values of CC, RMSE and APE pertaining to the training set data are indicators of the GCV prediction accuracy of the models, the values in respect of the test set data indicate the generalization ability of the models. It is seen from the CC values in respect of the training set outputs of all the seven ANN models (see Table 3.9) that the respective magnitudes are high (>= 0.984). This indicates that the models possess excellent GCV prediction accuracy. The CC magnitudes in respect of the test set outputs are also high and comparable with those corresponding to the training sets thus indicating that the models possess excellent generalization ability as well.

Among the seven ANN-based models, the most comprehensive one (model-I) using all major constituents of the proximate and ultimate analyses as also the He-density as inputs has yielded the best overall GCV prediction accuracy and generalization performance. The corresponding values of the CC for both the training and test sets (0.996 and 0.997) are highest among the seven models. This result suggests that the most accurate estimation of GCV can be made from the major constituents of the proximate and ultimate analyses.

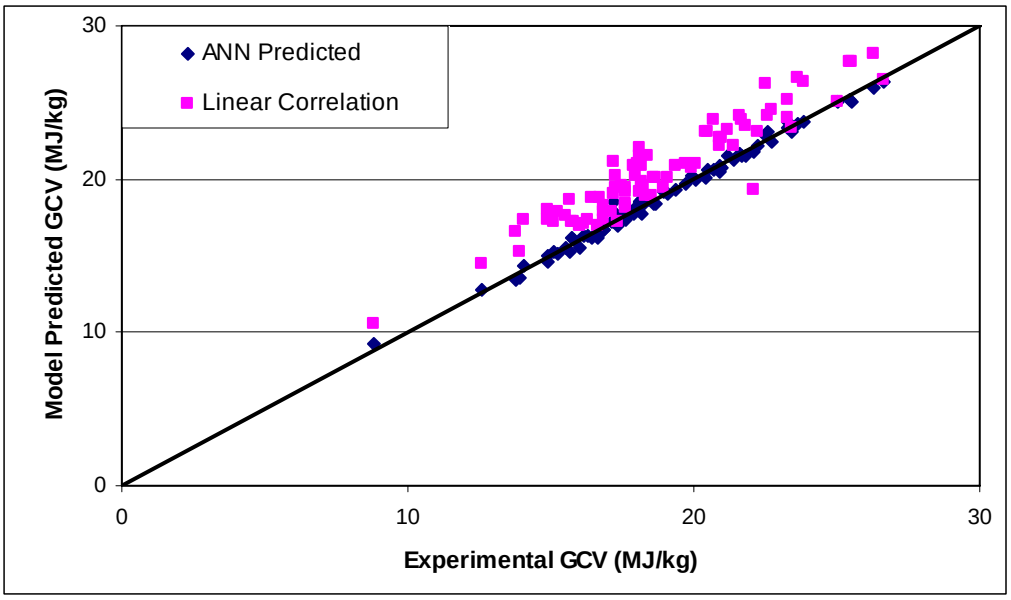

Figure 3.5: Graphical comparison of experimental GCVs with those estimated by ANN model-II and Eq. (3.13)

A comparison of the performance of the ANN model-II with its linear counterpart (Eq. 3.12) reveals that the ANN model possesses far superior prediction accuracy and generalization ability (i.e., higher CC and significantly lower RMSE and APE values). Figure 3.5 shows comparative plots of the GCVs determined experimentally and those estimated by the ANN model-II and Eq. (3.12). A close match between the experimental and model predicted GCVs indicates that the ANN model-II is capable of reasonably accurate GCV estimation from the knowledge of just two proximate analysis constituents, namely ash and moisture contents.

The ANN model-III considers He-density as an additional input to those considered by the model-II. The prediction and generalization performance of this model shows no significant improvement over the performance of the ANN model-II using ash and moisture as inputs. This suggests that inclusion of He-density does not provide any additional information useful in improving the model's GCV prediction accuracy over that provided by ash and moisture contents. The prediction results from ANN model-IV wherein He-density is replaced by the percentage of fixed carbon are very similar to those of models II and III. This can be judged by the close match

between the CC, RMSE and APE magnitudes in respect of the training and test sets of model-IV and models II and III.

The ANN model-V considers the elemental composition of coals as model inputs. Among the five prominant elements present in the coal namely carbon, hydrogen, sulphur, nitrogen and oxygen, only the first four are included in the input space since oxygen content is a derived quantity and can be calculated by subtracting the summation of the weight percentages of C_C, C_H, C_S, and C_N from 100. In this case too the ANN model could predict and generalize the GCV of Indian coals with excellent precision as indicated by the high CC magnitudes (≈ 0.984) for the training as well as test data sets.

The ANN model-VI is a coal-specific nonlinear counterpart of the linear correlation (Eq. 3.13) proposed by Channiwala et al. [2002]. Thus a comparison was made of the GCV predictions by model-VI and Eq. (3.13). This comparison reveals that the correlation coefficient values in respect of the GCV predictions made by both the models match very closely (i.e. CC ≈ 0.99). However, there exists a significant difference in the RMSE as well as APE values in respect of the predictions made by the ANN-based and linear models. Specifically, the ANN model could predict the GCVs with better accuracy and generalization performance ($RMSE_{trn}$ = 0.516, $RMSE_{tst}$ = 0.688, APE_{trn} = 2.215, APE_{tst} = 2.625) when compared to that of the linear Eq. 2.13 ($RMSE_{trn}$ = 4.352, $RMSE_{tst}$ = 4.515, APE_{trn} = 22.93, APE_{tst} = 21.484). Figure 3.6 shows comparative plots of the GCVs determined experimentally and those estimated by the ANN model-VI and Eq. (3.13). It is seen in this figure that the ANN model is able to predict the GCV with much higher accuracy when compared with Eq. (3.13).

The ANN-based model-VII uses percentages of fixed carbon, volatile matter and ash as model inputs. Parikh et al. [2005] earlier used these inputs in developing a correlation (Eq. 3.14) to overcome the experimental procedure involved in the determination of the elemental composition used in Eq. (3.13). A comparison of the GCV prediction and generalization performance of model-VII and Eq. (3.14) reveals (see Table 3.10) that similar to model-VI, the model-VII shows a better GCV

prediction accuracy and improved generalization performance when compared with that of Eq. (3.14). Figure 3.7 shows comparative plots of the GCVs determined experimentally and those estimated by model-VII and Eq. (3.14). As can be seen in this figure, there exist an excellent match between the ANN predicted and experimental GCVs.

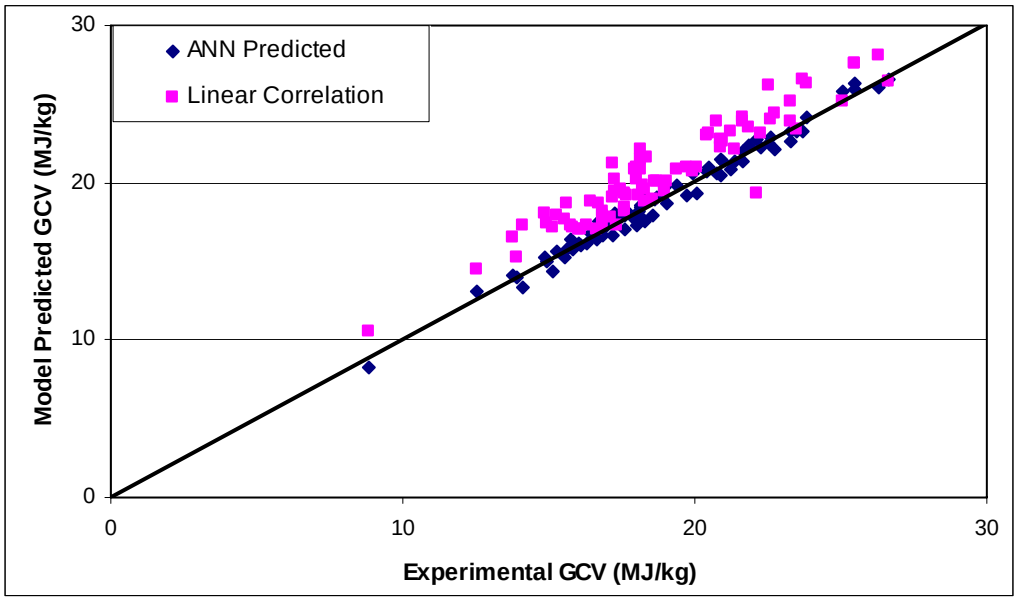

Figure 3.6: Graphical comparison of experimental GCVs with those estimated by ANN model-VI and Eq. (3.14)

The above-described results suggest that ANNs owing to their excellent nonlinear modeling ability are a better alternative to the linear models for the prediction of GCV of coals. The ANN weights can be used to predict the GCV magnitudes of new coal samples of Indian origin; the weight parameters of the ANN models can be obtained from the author.

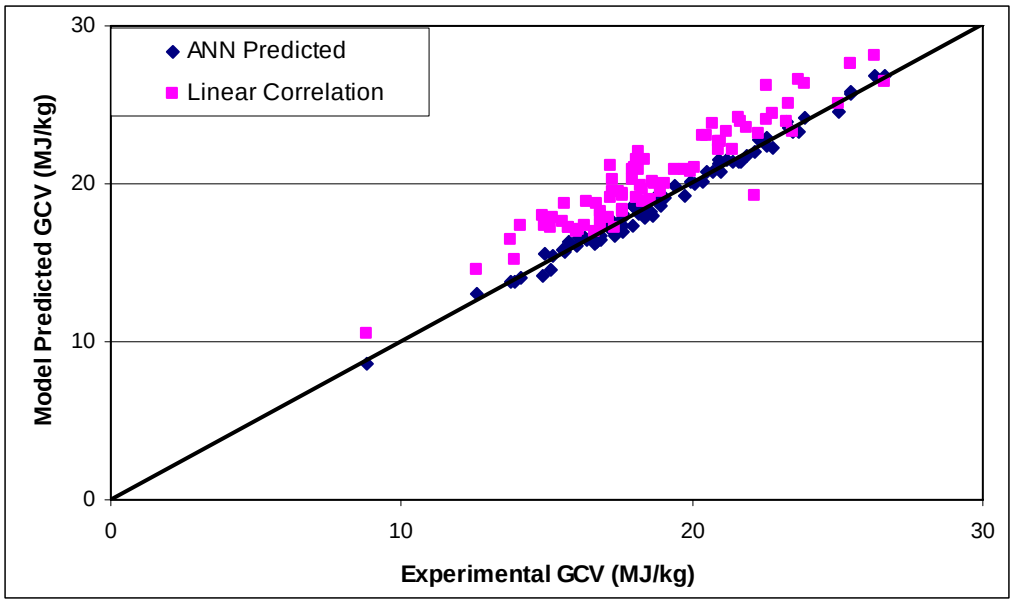

Figure 3.7: Graphical comparison of experimental GCVs with those estimated by and ANN model-VII and Eq. (3.15)

3.4.6 Identifying Important Inputs of ANN Models

There exist a number of methods for identifying important inputs of a nonlinear model and these have been reviewed by Sung [1998]. The important inputs are those, which even when perturbed by a small amount cause a relatively large change in the model output. In the case of ANN models, two methods namely sensitivity analysis (SA) and change of mean-square-error (MSE) have been proposed for identifying the hierarchy of model inputs in the order of their influence on the model output (see Section 3.4.7). In SA, the important inputs can be identified directly from an optimally trained ANN model whereas in the MSE methodology a number of ANN models need to be developed for identifying the significant inputs. An additional input identification method, known as "Fuzzy Curves" (see Section 2.5.6) has been proposed by Lin and Cunningham [1994, 1995]. In this method, a separate fuzzy curve model (FCM) is created for each input variable from the example input-output data. Next, the inputs are ranked depending upon how well a fuzzy curve model captures the relationship between an input and the output variable. In this study, we have chosen SA for identifying the important constituents of the

proximate and ultimate analysis that affect the GCV since the necessary optimal ANN models for the GCV prediction are already available.

The sensitivity analysis of an ANN model is performed by computing the sensitivity coefficient matrix (SCM). The numerical procedure for SCM for a single hidden layer MLP network was proposed by Englebrecht et al. [Engelbrecht, 1995] and Zurada et al. [1994]. This procedure was subsequently modified by Sung [1998] for computing the SCM of a two hidden layer MLP network. Both these procedures use the optimal weight parameters of an MLP model for identifying the hierarchy of significant network inputs. The MLP models that have been developed in this study use two hidden layers and, therefore, the SCM procedure of Sung [1998] has been used for conducting SA; this procedure has been appropriately modified as given below for computing the percentage of sensitivity $\left(\hat{S}\right)$ that an input variable exhibits towards an output variable.

3.4.7 Sensitivity Analysis (SA) of ANN Models

This section describes the procedure for computing the sensitivity coefficient matrix (SCM) and the percentage of sensitivity $\left(S_e\right)$ exhibited by an input variable towards an output of a two-hidden layer MLP model. Consider an MLP neural network (see Figure 2.1) housing *I*, *J*, *K* and *L* number of nodes in its input, hidden-I, hidden II and output layers, respectively. The nodes in the input, hidden-I, hidden-II and output layers are described by *X*, *V*, *Z* and *Y*, respectively, where $X = \left(x_1, x_2, ..., x_i, ..., x_I\right)$, $V = \left(v_1, v_2, ..., v_j ..., v_J\right)$, $Z = \left(z_1, z_2, ..., z_k, ..., z_K\right)$ and $Y = \left(y_1, y_2, ...y_l, ..., y_L\right)$.

For a training set pattern p, the sensitivity of the *l*th output (y_l) with respect to the input x_i is defined as:

$$S_{l,i}^{p} = \frac{\partial y_l}{\partial x_i} = y_l^{'} \sum_{k=1}^{K} \left(w_{kl}^{o} z_k^{'} \sum_{j=1}^{J} \left(w_{jk}^{H2} v_j^{'} w_{ij}^{H1} \right) \right) \tag{3.16}$$

where $y^{'}, z^{'}$ and $v^{'}$ are the respective derivative values of the activation function used in the computation of y, z and v; the weight on connection between *i*th and *j*th nodes is denoted as w_{ij}. Equation (3.16) can be recasted to define the sensitivity of any output with respect to any input pattern p in the training set as follows:

$$S^{p} = Y^{'} W_{KL} Z^{'} W_{JK} V^{'} W_{JK} \tag{3.17}$$

where, W_{KL}, W_{JK} and W_{IJ} are the weight matrices in respect of connections between output and hidden layer-II nodes, hidden layer-II and hidden layer-I nodes, and hidden layer-I and input layer nodes respectively; $Y^{'}(L \times L)$, $Z^{'}(K \times K)$ and $V^{'}(J \times J)$ are the diagonal matrices defined as:

$$Y^{'} = diag\left(y_1^{'}, ..., y_L^{'}\right) \tag{3.18}$$

$$Z^{'} = diag\left(z_1^{'}, ..., z_K^{'}\right) \tag{3.19}$$

$$V^{'} = diag\left(v_1^{'}, ..., v_J^{'}\right) \tag{3.20}$$

Equation 3.17 defines sensitivity with respect to a single input-output pattern. To calculate the sensitivity with respect to each input variable in the training set, following equation can be used [Zurada et al., 1994]:

$$\tilde{S}_{l,i} = \sum_{p=1}^{P} \frac{| S_{li}^{p} |}{P} \tag{3.21}$$

where $\tilde{S}_{l,i}$ denotes the sensitivity of *i*th input variable towards *l*th output. Finally, the sensitivity values can be normalized as given below to obtain the percent sensitivity value for an *l*th output with respect to an *i*th input.

$$\hat{S}_{l,i} = 100 \times \frac{\tilde{S}_{l,i}}{\sum_{i=1}^{I} \tilde{S}_{l,i}} \tag{3.22}$$

The results of the sensitivity analysis of the comprehensive ANN model-I are presented in Table 3.10. It can be observed from the tabulated values of $\hat{S}$ that the GCV exhibits highest sensitivity of 25 % towards oxygen content. The other significant inputs with sensitivity values higher than 10 % are, carbon ($\hat{S}$ = 20.4 %), ash ($\hat{S}$ = 15.9 %), fixed carbon ($\hat{S}$ = 12.3 %) and moisture ($\hat{S}$ = 10.4 %); in comparison, the GCV magnitude exhibits relatively lower sensitivity ($\hat{S}$ < 5 %) towards volatile matter, He-density, nitrogen, hydrogen and sulphur.

3.4.8 Conclusion

The existing models for estimating the GCV of coals from the constituents of the proximate and/or ultimate analyses are linear in character. It is observed that while the GCV exhibits a linear dependence on a number of constituents of the proximate and ultimate analyses, there exists a strong possibility that the GCV is nonlinearly dependent on some constituents of the stated analyses. Accordingly, seven nonlinear artificial neural network models of varying rigor have been developed in this study for the estimation of GCV with a special focus on Indian coals. The results of the GCV estimation clearly suggest that all the ANN models possess an excellent prediction accuracy and generalization performance with the comprehensive model (I) that uses all the major constituents of proximate and ultimate analyses as inputs, yielding the best prediction and generalization accuracy. More significantly, the ANN models are found to estimate the GCV magnitudes with better accuracy when compared to the three linear models using same inputs. Additionally, a sensitivity analysis of the ANN model-I has been performed to identify the hierarchy of model inputs in the order of their influence on the GCV magnitudes. The results of this analysis indicate that oxygen, carbon, ash, fixed carbon, and moisture have a stronger influence on the GCV than volatile matter, He-density, nitrogen, hydrogen and sulphur. This study clearly shows that ANNs are an attractive strategy for the estimation of GCV of coals. The modeling approach presented here can also be extended gainfully for an accurate GCV estimation of a wide spectrum of solid, liquid and gaseous fuels.

Table 3.7: Proximate and ultimate analysis data of Indian coals along with experimental GCV values

Sr. No.	Region	C_M	C_A	V_m	F_C	C_C	C_H	C_S	C_N	C_O	ρ_{He}	GCV (experimental) (MJ/kg)
1	MCL	6	38	26.3	29.7	41.9	3.29	0.26	0.89	53.7	1.75	17.17
2		5.8	41.1	23.9	29.2	40.2	2.87	0.18	0.56	56.2	1.81	15.53
3		7.5	34	26.2	32.3	45.8	3.26	0.29	0.71	50.0	1.71	17.63
4		7.6	32.4	22.5	37.5	45.4	2.81	0.26	0.85	50.7	1.75	17.48
5		8.3	32.5	27	32.2	43.5	3.19	0.87	0.81	51.7	1.66	17.24
6		5.8	44.7	22.5	27	37.0	2.59	0.24	0.75	59.4	1.85	14.09
7		5.7	47.8	22.3	24.2	32.9	2.40	0.40	0.80	63.5	1.92	12.56
8		6.8	41.8	25	26.4	39.2	2.68	0.35	0.83	56.9	1.76	15.13
9		4.4	35.6	27.6	32.4	45.5	3.25	0.62	0.91	49.7	1.7	18.17
10		4.6	39.7	25.5	30.2	40.2	3.09	0.62	0.94	55.2	1.74	16.74
11		5.5	35.5	28.7	30.3	45.4	3.12	0.54	0.92	50.0	1.6	18.02
12		4.8	35.5	28.4	31.3	46.1	3.23	0.48	0.99	49.2	1.6	18.31
13		7.1	34.4	27.7	30.8	46.1	3.42	0.62	1.10	48.8	1.71	18.09
14	NCL	9.5	17.6	29.1	43.8	57.7	3.47	0.29	1.12	37.5	1.47	22.55
15		7.2	17.4	30.0	45.4	58.8	3.54	0.37	1.10	36.2	1.54	23.29
16		5.2	36.7	28.2	29.9	43.3	3.31	0.51	0.84	52.0	1.69	17.27
17		6.7	24.8	26.8	41.7	51.9	3.00	0.40	0.79	43.9	1.77	19.95
18		6.2	28.1	27.4	38.3	51.0	3.58	0.39	0.95	44.1	1.63	18.28

Sr. No.	Region	C_M	C_A	V_m	F_C	C_C	C_H	C_S	C_N	C_O	ρ_{He}	GCV (experimental) (MJ/kg)
19		7.1	33.5	25.5	33.9	46.0	3.28	0.30	0.89	49.5	1.59	16.12
20		6.4	39.3	22.5	31.8	41.1	2.75	0.44	0.82	54.9	1.7	17.10
21		5.7	36.6	27.7	30.0	42.6	3.16	0.34	0.77	53.1	1.62	14.87
22		5.6	42.4	26.1	25.9	38.8	2.98	0.35	0.72	57.1	1.88	15.79
23		7.0	40.7	24.4	27.9	39.3	3.06	0.43	0.75	56.5	1.75	18.94
24		9.9	28.2	27.5	34.4	48.5	3.34	0.35	0.97	46.8	1.51	17.98
25		7.2	34.4	25.3	33.1	45.3	3.12	0.00	0.57	51.0	1.61	15.74
26		10	34.5	24.7	30.8	41.2	3.02	0.44	0.64	54.7	1.6	20.50
27		5.4	30.0	29.4	35.2	48.8	3.78	0.37	0.85	46.2	1.55	18.56
28	WCL	8.2	31.7	27.3	32.8	46.3	2.7	0.4	1.0	49.6	1.64	21.21
29		6.3	26.2	24.6	42.9	54.1	2.9	1.0	1.2	40.8	1.63	18.13
30		8.0	32.4	24.4	35.2	45.6	2.6	1.1	1.0	49.7	1.69	16.83
31		7.6	38.1	24.5	29.8	41.2	2.5	0.4	0.9	55.0	1.7	21.63
32		9.0	21.4	26.3	43.3	54.6	3.0	0.4	1.2	40.8	1.58	16.30
33		8.1	35.5	24.5	31.9	41.5	2.3	0.3	1.0	54.9	1.71	8.84
34		4.9	60.8	17.4	16.9	23.0	1.2	0.5	0.5	74.8	1.82	18.14
35		8.8	31.8	24.6	34.8	43.8	2.6	2.2	0.9	50.5	1.67	22.59
36		7.4	22.1	29.2	41.3	54.8	3.5	1.1	1.2	39.4	1.49	18.38
37		8.3	32.2	25.0	34.5	45.0	2.8	1.9	1.0	49.3	1.61	17.61
38		7.3	33.4	25.1	34.2	43.5	2.6	1.1	1.0	51.8	1.55	21.85
39		8.5	18.9	29.7	42.9	55.1	3.3	0.7	1.2	39.7	1.54	18.70

Sr. No.	Region	C_M	C_A	V_m	F_C	C_C	C_H	C_S	C_N	C_O	ρ_{He}	GCV (experimental) (MJ/kg)
40		7.5	30.6	25.4	36.5	46.6	2.7	2.0	1.0	47.7	1.55	17.64
41		6.9	34.3	24.0	34.8	44.1	2.6	1.4	1.0	50.9	1.57	15.24
42		5.3	43.2	22.5	29.0	37.8	2.2	2.0	0.8	57.2	1.62	17.93
43		7.3	33.1	25.8	33.8	44.9	2.7	0.7	0.9	50.8	1.46	16.01
44		6.3	37.9	24.1	31.7	42.5	2.52	0.46	0.79	53.7	1.78	19.40
45		8.4	26.8	26.4	38.4	48.4	3.35	0.52	0.00	47.7	1.58	18.21
46		6.5	33.4	26.2	33.9	45.7	3.00	0.50	1.30	49.5	1.67	22.76
47		4.3	25.5	28.4	41.8	56.0	3.60	0.60	1.50	38.3	1.58	23.67
48		3.6	23.5	29.8	43.1	57.7	3.70	0.60	1.60	36.4	1.54	20.98
49		5.1	29.3	27.3	38.3	51.8	3.30	0.60	1.10	43.2	1.62	19.72
50		4.3	33.3	26.7	35.7	48.6	3.10	0.60	1.20	46.5	1.65	13.78
51		5.7	46.3	22.0	26.0	34.8	3.30	0.60	1.30	60.0	1.6	15.65
52		5.4	40.7	23.9	30.0	40.4	2.60	0.40	1.00	55.6	1.79	16.67
53		6.0	38.8	23.9	31.3	41.6	2.70	0.50	0.90	54.3	1.69	16.84
54		6.0	42.8	24	27.2	37.8	2.50	0.40	1.10	58.2	1.74	17.22
55		0.8	45.7	15.7	37.8	39.1	2.75	0.24	0.79	57.1	1.78	20.08
56		5.0	31.3	28.9	34.8	48.4	3.09	0.34	0.73	47.4	1.67	22.26
57	SECL	4.6	29.0	23.3	43.1	53.3	3.0	1.0	1.0	41.7	1.53	25.47
58		5.5	18.7	25.5	50.3	61.1	3.6	1.2	1.0	33.1	1.45	18.63
59		8.4	31.1	21.9	38.6	46.5	2.3	1.0	0.7	49.5	1.57	19.07
60		8.2	30.8	22.0	39.0	47.3	2.4	1.0	0.6	48.7	1.56	21.66

Sr. No.	Region	C_M	C_A	V_m	F_C	C_C	C_H	C_S	C_N	C_O	ρ_{He}	GCV (experimental) (MJ/kg)
61		5.6	26.9	24.3	43.2	53.0	3.0	1.1	0.9	42.0	1.52	22.13
62		6.6	25.3	26.6	41.5	52.7	3.2	1.1	1.2	41.8	1.54	26.29
63		6.0	15.0	26.9	52.1	62.5	3.7	1.3	0.5	32.0	1.46	20.90
64		7.2	28.9	26.0	37.9	48.2	3.0	1.0	0.7	47.1	1.52	23.30
65		9.1	18.2	25.6	47.1	57.5	3.0	1.1	0.8	37.6	1.49	20.40
66		5.4	32.6	25.9	36.1	49.1	2.9	0.9	0.6	46.5	1.55	23.46
67		7.1	21.5	24.7	46.7	57.8	3.0	1.2	0.6	37.4	1.46	26.65
68		3.8	18.4	25.0	52.8	61.9	3.4	1.2	0.4	33.1	1.48	25.05
69		7.2	16.7	27.8	48.3	61.2	3.4	1.1	0.7	33.6	1.43	25.49
70		6.7	17.1	26.2	50.0	62.2	3.4	1.2	0.5	32.7	1.45	17.60
71		6.1	36.4	24.9	32.6	43.9	2.6	0.9	0.2	52.4	1.68	14.93
72		6.8	43.2	22.2	27.8	36.6	2.2	0.7	0.2	60.3	1.77	23.86
73	SCFL	7.8	17.1	24.9	50.2	60.5	3.2	0.3	1.1	34.9	1.48	21.40
74		7.4	24.3	27.1	41.2	53.6	3.2	1.1	1.2	40.9	1.54	20.73
75		7.1	29.0	28.1	35.8	50.4	3.1	1.5	1.1	43.9	1.52	20.90
76		5.7	37.5	23.2	33.6	43.1	2.6	1.0	1.1	52.2	1.59	17.33
77		8.9	35.9	23.2	32.0	42.0	2.3	2.2	1.0	52.5	1.64	16.41
78		5.5	40.1	24.2	30.2	42.0	2.4	0.7	1.0	53.9	1.68	16.85
79		5.1	46.5	20.8	27.6	35.6	2.0	0.5	0.8	61.1	1.78	13.91

Table 3.8: Details of ANN-based GCV models*

Model No.	Basis	Model Inputs	Training set size	Test set size	*I*	*J*	*K*	Learning rate (η)	Momentum coeff. (μ)
I	As received	Moisture, ash, volatile matter, fixed carbon, carbon, hydrogen, sulphur, nitrogen, oxygen and He-density	63	16	10	5	6	0.2	0.1
II	As received	Ash, moisture	64	15	2	5	6	0.2	0.1
III	As received	Ash, moisture, He-density	64	15	3	5	6	0.2	0.1
IV	As received	Ash, moisture, fixed carbon	64	15	3	5	10	0.2	0.1
V	As received	Carbon, hydrogen, sulphur, nitrogen	64	15	6	3	6	0.2	0.1
VI	Dry	Carbon, hydrogen, sulphur, nitrogen, oxygen, ash	64	15	3	5	6	0.2	0.1
VII	Dry	Fixed carbon, volatile matter, ash	64	15	3	5	6	0.2	0.1

* *I* = No. of input nodes; *J* = No. of nodes in the first hidden layer; *K* = No. of nodes in the second hidden layer.

Table 3.9: Statistical analysis of GCV prediction and generalization performance of ANN-based and linear models*

Models	Performance of ANN models						Performance of linear models*					
	Training set			Test set			Training set			Test set		
	CC	RMSE	APE	CC	RMSE	APE	CC	RMSE	APE	CC	RMSE	APE
I	0.996	0.281	1.233	0.997	0.514	2.067	-	-	-	-	-	-
II	0.986	0.537	2.433	0.996	0.787	3.062	0.565	3.191	13.957	0.623	3.931	17.412
III	0.987	0.528	2.443	0.994	0.618	2.205	-	-	-	-	-	-
IV	0.988	0.479	2.155	0.995	0.725	2.663	-	-	-	-	-	-
V	0.984	0.590	2.701	0.988	0.630	2.405	-	-	-	-	-	-
VI	0.989	0.516	2.215	0.989	0.688	2.625	0.983	4.352	22.930	0.991	4.515	21.484
VII	0.984	0.612	2.574	0.989	0.696	2.485	0.981	2.699	12.893	0.985	3.280	14.146

* The GCV prediction and generalization performance of ANN-models II, VI, and VII have been compared with the linear models described by Eqs. 3.12, 3.13 and 3.14, respectively.

Table 3.10: Results of sensitivity analysis (SA) of ANN model-I

Sr. No.	SA-based hierarchy of inputs	% sensitivity $(\hat{S})$
1	Oxygen	25.0757
2	Carbon	20.4558
3	Ash	15.8912
4	Fixed carbon	12.2942
5	Moisture	10.4246
6	Volatile matter	4.3730
7	He density	4.1563
8	Nitrogen	3.6144
9	Hydrogen	2.7548
10	Sulphur	0.9599

3.5 SOFT-SENSOR DEVELOPMENT FOR FED BATCH BIOREACTORS USING SUPPORT VECTOR REGRESSION*

Industrial fermentation processes involving a fed-batch operation are extensively used for the production of antibiotics, amino acids, microbial cells, enzymes and organic acids. Efficient monitoring and control of these systems is greatly facilitated if sensor measurements of the process variables or at least their reasonably accurate estimates are available continuously in real-time. Often, reliable biosensors or robust on-line measurement techniques for the process variables such as the concentrations of the active biomass and product species are not readily available. To overcome this difficulty, usage of software based sensors (known as 'soft-sensors' or 'virtual sensors') is recommended. Softsensors allow an online estimation of the unmeasured or "difficult-to-measure" process variables from the values of easily and frequently measured variables. In recent years, artificial neural networks owing to their significant ability of nonlinear function approximation are widely prescribed for the development of softsensors. However, the ANN approach suffers from the drawbacks such as extensive numerical effort required to find a globally optimum solution and "black-box" nature of the resultant model. In this section, therefore, a state-of-the-art statistical/machine learning based formalism known as "support vector regression (SVR)" possessing some novel characteristics, has been presented for the soft-sensor development in fed-batch processes. The efficacy of the SVR formalism has been demonstrated by considering two simulated bio-processes namely, invertase and streptokinase. Also, the performance of the SVR based soft-sensors is compared with those developed using a standard ANN technique. In the case of invertase process, the differential utilization rate of ethanol and glucose is used as the basis for developing a soft-sensor. For the streptokinase process, soft-sensors

---

* Desai K. M., Y. P. Badhe, S. S. Tambe and B. D. Kulkarni, *Biochemical Engineering Journal*, 27, 2006, 225–239.

for the active biomass and streptokinase concentrations are developed using the characteristic that the wild type and its recombinant give rise to different growth and substrate utilization profiles. The results presented here clearly indicate that the SVR is an attractive alternative to ANNs for the softsensor development.

3.5.1 Introduction

Industrial applications of enzymes have increased rapidly in the past few years. Most enzymes are produced by a submerged, aerobic fermentation involving a batch operation. A fed-batch culture is widely used for the production of enzymes from a microbial source that suffers catabolic repression. The fed-batch operation is found to be superior to batch and continuous operations in situations of a parallel formation of the desired and undesired products [Ohno et al., 1976; Staniskis, 1984].

The traditional methods of improving the efficiency of a bio-process comprise strain modification and development of the media by empirical means. Invariably, these approaches are found to be inadequate owing mainly to the process variability and operational difficulties. Thus, in recent years a significant attention is being paid to monitoring, control and optimization of bio-processes with a view to bring about improvements in the process productivity and economics. The ability of controlling a bio-process at its optimal state accurately and robustly is of immense importance from the view point of reducing the production cost, increasing product yield, and maintaining quality of metabolic products. An efficient feed-back or the model based control is thus necessary to achieve the stated objectives.

The task of controlling and monitoring a bioprocess efficiently and robustly is faced with major difficulties such as the existence of significant uncertainties emanating from the complex non-linear dynamics typically exhibited by the bioprocesses, and the lack of–in most cases–reliable hardware and/or bio-sensors for measuring values of the process and/or product quality variables [Shimizu, 1996]. The first of these difficulties can be overcome by constructing an appropriate phenomenological or an empirical process model capable of describing the non-linear process dynamics accurately. The second significant difficulty, that is the lack of reliable hardware and/or biosensors, can be addressed

by developing softsensors that can accurately estimate values of the process and product quality variables in real-time. Softsensors are software based sophisticated monitoring systems, which can relate less accessible and infrequently measured process variables with those measured easily and frequently. In essence, softsensors correlate the unmeasured process and product quality variables with the frequently measured process variables and thus assist in making real-time predictions of the unmeasured variables. Softsensors are useful in the control and monitoring of fermentation processes wherein owing to the unavailability of appropriate hardware sensors and/or bio-sensors, the values of important process and product quality variables are not continuously available. The predictive performance of softsensors depends upon the reliable measurements of easily accessible process variables and also on the mathematical and/or statistical techniques used in the interpretation and correlation of the process data. A number of authors have described different approaches for the soft-sensor development in the batch, fed-batch and continuous processes [Bastin 1990; Acha et al., 1999; Albert et al., 2001; Linko et al., 1999; James et al., 2002 ; Sachez et al., 1999; Adilson et al., 2000; Montague et al., 1986; Pons et al., 1988].

In the last decade, *artificial neural networks*, (refer Section 2.2.1) owing to their attractive functional approximation properties, have become a powerful formalism not only for constructing exclusively data-driven nonlinear process models but also for developing soft-sensors [Eerikäinen et al., 1993; Zhu et al., 1996; Karim et al., 1992]. More recently, support vector regression (refer Section 2.2.2), which shares many features with ANNs, but has some additional novel characteristics, is gaining widespread acceptance for exclusively data-driven nonlinear modeling applications [Nandi et al., 2004]. Despite endowed with a number of attractive features, the SVR being a new formalism, is yet be explored widely in the biochemical/biotechnology applications. Therefore, in the present work, the SVR formalism has been introduced for developing soft-sensors for bioprocesses. Specifically, two simulated fed-batch processes namely invertase and streptokinase, have been considered for the SVR-based softsensor development. Also, the performance of the SVR-based softsensors has been rigorously compared with those developed using a standard ANN formalism. The

results of the two case studies presented here clearly indicate that the SVR is an attractive strategy for developing soft-sensors for bioprocesses.

3.5.2 Invertase Production Model

The phenomenological model for biphasic growth of *Saccharomyces carlsbergensis and* invertase production is given as [Pyun et al., 1989]

$$\frac{d}{dt}(x_t) = (\mu_G + \mu_A)x_t \tag{3.23}$$

$$\frac{d}{dt}(s) = -\frac{\psi s x_t}{v} + \frac{s_F F}{v} \tag{3.24}$$

$$\frac{d}{dt}(e) = (\pi_A - \pi_C)x_t \tag{3.25}$$

$$\frac{d}{dt}(v) = F \tag{3.26}$$

$$\frac{d}{dt}(c_{\text{inv}}) = (\eta_S \mu_G + \eta_A \mu_A)x_t \tag{3.27}$$

where

x_t , s, and e: concentrations of cells, glucose, and ethanol, respectively

μ_G, μ_A, ψ: specific growth rates on glucose and ethanol and, specific rate of glucose consumption, respectively,

π_A, and π_C: specific rates of ethanol production and ethanol consumption, respectively,

η_S and η_A: ratios of the specific invertase synthesis rate to the specific growth rate on glucose and on ethanol, respectively,

c_{inv}: invertase activity,

q: volumetric feed rate of glucose,

s_F: glucose feed concentration,

v: fermenter volume.

The following nonlinear feed rate profile that maximizes the streptokinase production [Toda, et al., 1980] is used for the data generation.

q = 0.2 l/h for ($0 \le t \le 0.58$); q = 0 l/h for ($0.58 \le t \le 2.28$); $q = q_c$ l/h for ($2.28 \le t \le 12.4$); q = 0 l/h for ($12.4 \le t \le 13$)

where,

$$q_c = \frac{\psi x v}{s_F - s} + x_t v(-1.4892)(s)^{(1.2013)}(c_{inv})^{(-0.1046)} \qquad (3.28)$$

The details of above rate expression and kinetic model can be found in [Pyun, et al., 1989]. The following values of model parameters and operating conditions are used in SVR simulations:

μ_G^{max} = 0.39 h^{-1}, μ_A^{max} = 0.11 h^{-1}, k_s = 0.021 g/l, k_p = 0.014 g/l, $Y_{x/s}^R$ = 0.52 g/g, $Y_{x/s}^F$ = 0.15 g/g, $Y_{x/p}^R$ = 0.67, $Y_{p/s}^R$ = 0.33 g/g, e_0 = 0 g, $(c_{inv})_0$ = 0 KU/g, v_0 = 0.6 l, v_{max} = 1.5 l,

$s_F = 0.5/[v_{max} - v_0]$ g/l.

3.5.3 Softsensor for Invertase Process

Saccharomyces carlsbergensis shows a biphasic nature of growth, i.e., it can utilize glucose as well as ethanol in the event of a glucose scarcity [Dedem et al., 1975; Beejherk et al., 1977]. This feature is advantageous since among the two substrates namely glucose and ethanol, the latter is cheaper. Hence, the fermentation is conducted by maintaining the glucose concentration at a level such that the yeast is forced to use ethanol. *Saccharomyces carlsbergensis* and some other yeasts exhibit the diauxic type of growth whereby in the first growth phase, the glucose is utilized via an aerobic fermentation with carbon dioxide and ethanol as the main reaction products. When glucose is completely exhausted, the ethanol produced earlier serves as a substrate for the further growth. In a fed-batch culture, which is used extensively to suppress the catabolic repression, it is necessary to control the concentration of glucose at an optimal level for

maximizing the cellular yield and maintaining a high growth rate. When yeasts exhibiting the diauxic growth are used for the production of enzymes, the extent of cellular growth and enzyme production depends upon the balance between the metabolic states of the aerobic fermentation and respiratory growth. Accordingly, an estimation of the current metabolic state can be made from the changes in the concentrations of glucose and ethanol in the broth.

The major advantage of a fed-batch bioreactor is during fermentation, the feed composition and feed flow rate can be manipulated to maximize the product formation. Thus, manipulation of the feed rate is an important aspect of the fed-batch operation from the view point of process control and optimization. The optimal feeding policy depends significantly on the initial feed concentration and flow rate and its manipulation is highly sensitive to the changes in the kinetic parameters. Modak et al. [1986] have reported that for certain kinetics, glucose-stat (maintaining the glucose concentration at a constant level) is optimal. However, very often the substrate concentration is maintained by regulating the feed rate in an optimal manner. Pyun et al. [1989] have reported the results of a detailed study on the optimization of the biphasic growth of Saccharomyces carlsbergensis in a fed-batch culture. Recently, optimization of the same system has been carried out by Sarkar and Modak [2003] using *Genetic Algorithms* (see Section 2.4.3); the optimal feed profiles obtained thereby exhibit an excellent match with those obtained by Pyun et al. [1989]. Both these studies have reported four different profiles for as many combinations of high and low initial concentrations of the substrate and biomass. Since all these feed profiles were nearly identical, only one feed profile has been considered as a reference feed profile in this study.

To generate process data for developing a softsensor for the invertase from *Saccharomyces carlsbergensis* process, the phenomenological model proposed by Toda et al. [1980] (also see Pyun et al. [1989] has been used. It may however be noted that the model is used only to simulate the process and thereby generating the process data. In real practice, data collected by running the process physically, are to be used for developing the softsensor. As can be noted from the invertase model (see Section 3.5.2), The fed-batch invertase process is described in terms of five operating variables namely, glucose concentration (s, g/l), ethanol

concentration (e, g/l), bioreactor volume (v, l), biomass concentration (x_t, g/l) and invertase concentration (c_{inv}, KU/g). This case study aims at developing an SVR-based softsensor for the prediction of c_{inv}. To generate process data, the set of five ordinary differential equations (see Eqs. 3.23 to 3.28) representing the process dynamics was simulated under varying initial conditions; the ranges in which the initial values were varied are: ($0.3 < v_0 < 0.6$, $0.02 < s_0 < 0.1$, and $0.03 < x_{t0} < 0.06$). A total of 33 batches with varying initial conditions were simulated over the fermentation duration of 13 hrs. The values of four operating variables, i.e. s, v, x_t and e, and that of the single product quality variable namely, invertase concentration, c_{inv}, computed at 30 minute intervals formed the process data set. The real-world process data always contain some instrumental and measurement noise and, therefore, 5% Gaussian noise was introduced in each variable of the process data for mimicking the real-world process scenario. This set can be viewed as a three-way array of size 33 (number of batches) × 5 (process variables) × 27 (measurement intervals). Since the concentrations of glucose, biomass and ethanol as also the reactor volume significantly influence the activity of the invertase, it is necessary to consider the history of these variables along with their current values for developing the softsensor model. Thus, the current and lagged values of variables s, e, x_t and v, were used as inputs to the SVR-based model predicting the current value of the invertase concentration (activity). While the concentrations of glucose and ethanol can be estimated on-line using biosensors [Lui et al., 1998; Folly et al., 1996; Rank et al., 1995], the biomass concentration can be estimated from the optical density of the broth.

An SVR-implementation known as "ε-SVR" in the LIBSVM software library [Chang et al., 2002], was used to develop the softsensor model. The LIBSVM library utilizes a fast and efficient implementation of the widely used method known as *sequential minimal optimization* (SMO) [Joachims, 1998; Platt, 1998] for solving large quadratic programming problems and thereby estimating parameters, $\boldsymbol{\alpha}, \boldsymbol{\alpha}^*$ and b, of the SVR's fitting function (see Eq. 2.41). In this study, the RBF kernel function was used to avoid the dot product calculations in the feature space, $\boldsymbol{\Phi}$. To develop an SVR-based softsensor model possessing good prediction and generalization ability, it is necessary to judiciously select the number of lagged values of variables s, e, x_t and v. Accordingly, multiple

softsensor models with varying number of lagged values of the stated variables were constructed. The generalization ability of these models was evaluated by using a test set that comprised 20% of the available process data; the remaining 80% data were used as the training set for building the SVR models. The model that yielded the least RMSE magnitude for the test set was chosen as the optimal model; the RMSE was computed as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{p}\left(c_{inv}^{i}(t) - \hat{c}_{inv}^{i}(t)\right)^{2}}{p}} \tag{3.29}$$

where, i is the pattern index; t denotes the discrete time (Δt = 30 min) and, $c_{inv}^{i}(t)$ and $\hat{c}_{inv}^{i}(t)$ are the desired and SVR-predicted invertase concentrations corresponding to the ith pattern, respectively. The optimal softsensor model obtained by following the above-described procedure has ten inputs defining the current and lagged values of variables s, e, x_t and v, and the model is defined as:

$$\hat{c}_{inv}(t) = f\left(s(t), s(t-1), e(t), e(t-2), x_t(t), x_t(t-2), v(t), v(t-1), v(t-2), v(t-3)\right) \tag{3.30}$$

where, $x(t\text{-}k)$ denotes the value of a variable x lagged by k number of discrete time intervals and $\hat{c}_{inv}(t)$ is the SVR-model predicted invertase activity at time, t. It can thus be seen that the optimal SVR-based softsensor model has used 1, 1, 1 and 3 lagged values of the model input variables, s, x_t, e and v, respectively. The optimal number of support vectors (SVs) used by the SVR algorithm for fitting the invertase activity model was 228. The optimal values of the four ε-SVR algorithm specific parameters that minimized the RMSE with respect to the test set (E_{tst}) are: width of the RBF kernel (σ) = 1, regularization constant (C) = 2, loss function parameter (ε_{loss}) = 0.00001 and tolerance for the termination criterion (ε_{tol}) = 0.00001.

The performance of the SVR based optimal softsensor model in predicting the invertase activity was compared with that of the standard MLP-based model. Here, a two hidden layer MLP network trained using the EBP algorithm was used; the training and test sets used for developing the MLP based softsensor were the same as used in the development of the SVR-based softsensor. To construct an optimal MLP model, the effects of network's structural parameters (number of

hidden layers and the number of nodes in each hidden layer) as also two EBP-specific parameters, namely, the learning rate and momentum coefficient, were studied rigorously. Additionally, the effect of random weight initialization was examined to obtain a model that corresponds to the global or the deepest local optimum on the nonlinear error surface [Nandi, et al., 2001]. The MLP's architectural details and the EBP-specific parameter values that yielded an optimal softsensor model are: number of input nodes = 10, number of neurons in the hidden layer-I = 4, number of neurons in the hidden layer-II = 2, number of neurons in the output layer = 1, learning rate = 0.6 and momentum coefficient = 0.05.

The values of correlation coefficient (CC), RMSEs and the average error (%) pertaining to the invertase activity predictions made by the SVR and ANN based softsensors are listed in Table 3.11. As can be seen from the tabulated values that the CC magnitudes corresponding to the SVR model predictions are very close to unity indicating an excellent match between the desired and model predicted invertase activity values. Also, the average error (%) and RMSE values in respect of both the training set (E_{trn}) and the test set (E_{tst}) are sufficiently small. A close match between the stated statistical quantities for both the training and test sets indicates that the SVR based softsensor has excellent generalization ability as well. A comparison of CC, RMSE and average percent error values corresponding to the SVR and ANN predictions reveals that the SVR based softsensor has consistently outperformed the ANN-based softsensor model. These results suggest that the SVR-based softsensor can be effectively used for near-accurate online estimation of the invertase activity. An illustrative comparison of the SVR (see Figure 3.8) and ANN (see Figure 3.9) model predicted values of the invertase activity at batch times of 2, 7 and 13 hrs.

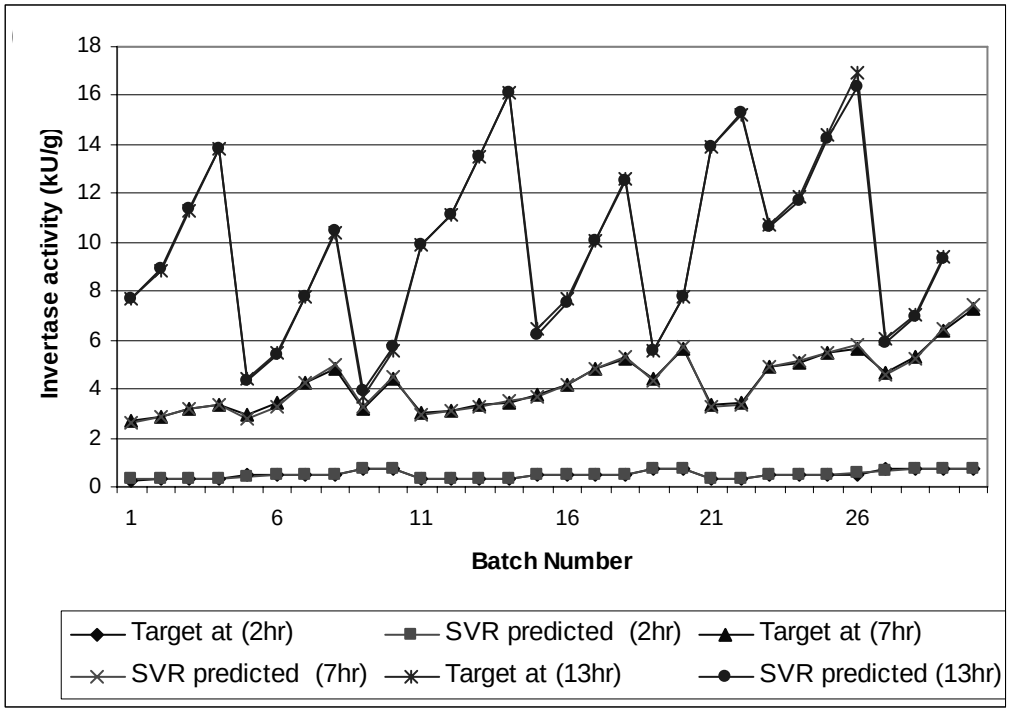

Figure 3.8: Invertase activity at two, seven and 13 hour time duration as predicted by the SVR-based softsensor

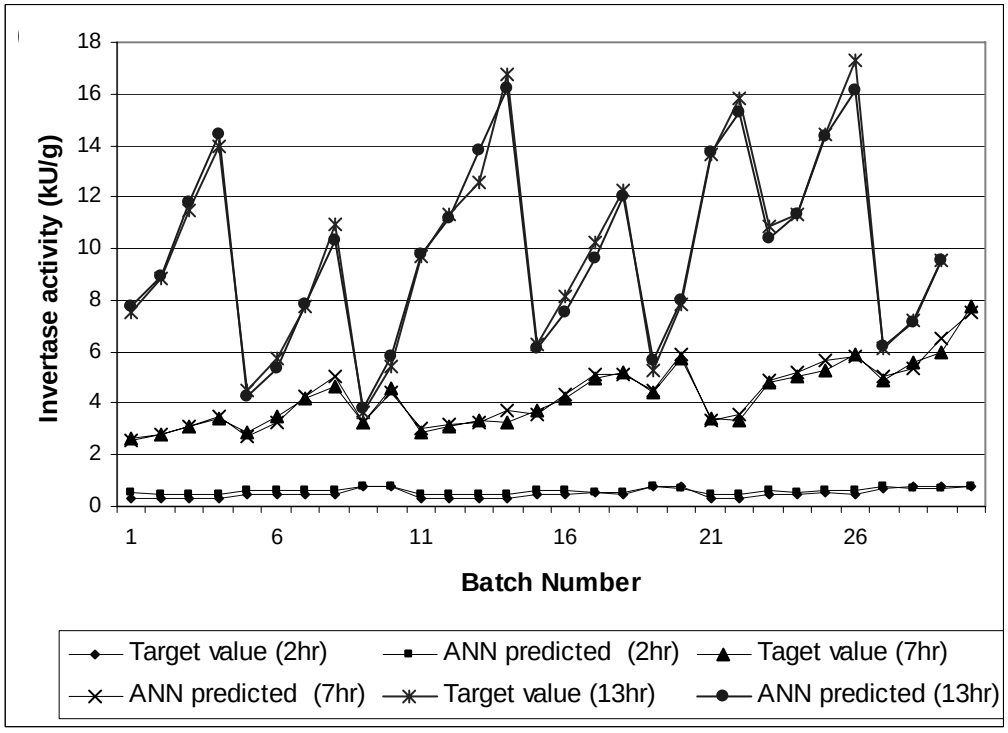

Figure 3.9: Invertase activity at two, seven and 13 hour time duration as predicted by the ANN-based softsensor

Table 3.11: Comparison of invertase activity prediction performance of SVR and ANN-based softsensor models

Model	Data	Correlation Coefficient	Average error (%)	RMSE
SVR	Training set	0.999	3.306	0.084
	Test set	0.999	3.764	0.115
ANN	Training set	0.997	12.068	0.266
	Test set	0.998	5.116	0.270

3.5.4 Softsensor for Streptokinase Process

The phenomenological model of streptokinase process using *Streptoccus sp.* in batch fermentation [Patnaik, 1999] is given as:

$$\frac{dx_t}{dt} = \mu x_a - \frac{q}{v} x_t \tag{3.31}$$

$$\frac{dx_a}{dt} = \left(\mu - k_d\right) x_a - \frac{q}{v} x_a \tag{3.32}$$

$$\frac{ds}{dt} = \frac{-\mu x_a}{Y_X} + \frac{q}{v}\left(s_{in} - s\right) \tag{3.33}$$

$$\frac{dl_a}{dt} = Y_M \mu x_a - \frac{q}{v} l_a \tag{3.34}$$

$$\frac{ds_t}{dt} = Y_P \left(\mu - k_d\right) x_a - k_P s_t - \frac{q}{v} s_t \tag{3.35}$$

$$\frac{dv}{dt} = q\left(t\right) \tag{3.36}$$

$$\mu = \mu_m \left(\frac{s}{K_S + s}\right)\left(\frac{K_I^b}{K_I^b + l_a^{\,b}}\right) \tag{3.37}$$

where,

x_t, x_a : concentrations of total and active biomass, respectively,

s, s_t, l_a : concentrations of substrate, streptokinase and lactic acid, respectively,

v : volume of reactor,

The following nonlinear feed rate profile that maximizes the streptokinase production is used for the data generation [Patnaik, 1995].

$$q(t) = a_0 + a_1(t/T) + a_2(t/T)^2 + a_3(t/T)^3 \tag{3.38}$$

The parameter values used to simulate the model are:

a_0 = 0.9959 (l/h), a_1 = -0.3037 (l/h), a_2 = -1.3418 (l/h), a_3 = 0.6499 (l/h), b = 2.39, k_d = 0.020 (l/h), k_p = 0.0005 (l/h), K_I = 12.66 (g/l), K_S = 13.14 (g/l), S_{in} = 70.0 (g/l), T =12.0 (h), Y_M = 4.80 (g/g), Y_P = 0.44 (g/g), Y_X = 0.15 (g/g), μ_m = 0.74 (l/h)

Initial conditions used in the model simulations are:

l_a (0) = 0 g/l, s_t(0) = 0 g/l, s_0 = 70 g/l, x_t(0) = 0.7 g/l, x_a(0) = 0.7 g/l, v(0) = 51.

3.5.5 Details of Softsensor Development

Genetically engineered microorganisms have become important vehicles for the production of valuable bio-molecules. Owing to the "gene dosage effect," each cell can possess multiple copies of a plasmid up to a certain threshold level. This feature results in the increased production of the recombinant product. The two major factors affecting the expression of a plasmid bearing gene are cultivation conditions [Ryan et al., 1989] and the bioreactor operation mode [Georgiou et al., 1985]. Though genetic engineering can also be used in the production of metabolites, its main application is in the production of high value proteins.

In the recombinant technology, the host microorganism is infected with multiple copies of plasmids. Each plasmid carries the gene responsible for the production of the desired product. Thus, during fermentation cells are forced to accumulate a very high concentration (in excess of 10% of the cell's dry weight) of the recombinant protein. Such a protein overproduction does not serve the microorganism in any useful way. Instead it burdens the cell and hence cells

always try to get rid of this burden [Ryan et al., 1991]. Several studies have reported a decrease in the specific growth rate of plasmid-bearing cells with an increase in the plasmid copy number [Lee et al.,1984; Seo et al., 1985]. Owing to their higher growth rate, the plasmid-free cells, once formed, outgrow the plasmid-bearing cells and eventually take over the entire population. Also, most cells while undergoing a cell division lack the mechanism for the proper partitioning of the plasmid copies in an offspring. These factors eventually lead to the plasmid instability, owing to which there always remains a fraction of plasmid-free cells, which utilizes the fermentation resources without yielding any desired product. Thus, it is necessary to ensure that the batch mode fermentation does not produce an excess of plasmid-free cells. This can be achieved by strict monitoring and control of the plasmid-free cells. Though an estimation of the biomass of frequently used host cells—such as the E-coli—is possible by measuring the optical density of the broth, online measurement of the plasmid-free cell fraction is in general difficult. The stated difficulty, however, can be overcome by developing softsensors for the estimation of concentrations of the active cell mass and the recombinant protein. It is well-known that there exists a substantial difference in the growth rates of the plasmid-free and plasmid-bearing cells. As a result, these cells give rise to two distinct biomass and substrate profiles [Ryan, et al., 1991]. Accordingly, the substrate and total biomass concentrations can be used to develop softsensors (as illustrated using the SVR and ANN formalisms) for predicting concentration values of the recombinant protein and active cell mass.

Similar to the invertase process, the streptokinase fed-batch process data were generated using the phenomenological model [Patnaik, 1995; 1999]. This model comprises six ordinary differential equations (ODEs) (Eqs. 3.31 to 3.37) representing the dynamics of six process variables namely, active biomass concentration (x_a, g/l), total biomass concentration (x_t, g/l), substrate concentration (s, g/l), reactor volume (v, l), lactic acid concentration (l_a, g/l) and streptokinase concentration (s_t, g/l). The data for multiple fed batches were generated by integrating the set of six ODEs using the Gear's algorithm and by varying the process initial conditions in the following ranges: $60 \leq s_0 \leq 80$ and $0.5 \leq x_{a0} \leq 0.9$. A total of 45 batches over 12 hr duration were simulated. The values of six

process variables viz., v, x_t, s, l_a, x_a and s_t computed at 30 minute intervals formed the process data set. Next, 5% Gaussian noise was introduced in each variable of the process data to mimic the real-life process scenario. This data formed a three-way array of size, 45 (number of batches) × 6 (process variables) × 25 (time intervals). Changes in the reactor volume and concentrations of the biomass and substrate (i.e. glucose) are the major indicators of a change in the streptokinase and active biomass concentrations. Accordingly, the current as also the lagged values of variables v, x_t, and s were considered as inputs to the two softsensor models predicting the current concentrations of streptokinase and the active biomass.

In this case study too, the ε-SVR algorithm from the LIBSVM software library was used to develop the two softsensor models. Although not considered here, a softsensor for predicting the lactic acid concentration also can be developed in a manner similar to the models for the streptokinase and active biomass concentrations. The number of lagged values of variables v, s and x_t, and also the ε-SVR specific parameters were chosen via an heuristic optimization such that the RMSE with respect to the test set is minimized. The optimal ε-SVR softsensor models that minimized the test set RMSE (E_{tst}) have eight inputs defining the current and lagged values of the three operating variables, v, s and x_t, and the form of the models is as given below.

$$\hat{s}_t(t) = f_1\left(v(t), v(t-1), v(t-2), v(t-3), s(t), s(t-1), x_t(t), x_t(t-2)\right) \qquad (3.39)$$

$$\hat{x}_a(t) = f_2\left(v(t), v(t-1), v(t-2), v(t-3), s(t), s(t-1), x_t(t), x_t(t-2)\right) \qquad (3.40)$$

where, $\hat{s}_t(t)$ and $\hat{x}_a(t)$ refer to the concentrations of streptokinase and active biomass at a discrete time, t, respectively, and $x(t-k)$, k = 1, 2, 3, denote the lagged values of a variable, x. The number of lagged values of each of the process variables namely, v, s and x_t, in Eqs. 3.39 and 3.40 were determined by varying the number systematically and choosing the optimal number that minimized the test set RMSE. During the development of two softsensors, a training set of 308 patterns was used to estimate the SVR model parameters $\boldsymbol{\alpha}$, $\boldsymbol{\alpha}^*$ and b, and a test set comprising 70 patterns was used to evaluate the generalization performance of the SVR models. These data patterns comprising the current and lagged values of

process variables were generated by appropriately arranging the process data from 45 batches. The optimal number of support vectors (SVs) used by the SVR algorithm for fitting the streptokinase and active biomass concentration models were 297 and 301, respectively. The values of four ε-SVR algorithm specific parameters that minimized the test set RMSE pertaining to the streptokinase softsensor are: width of the RBF kernel (σ) = 1, regularization constant (C) = 5, loss function parameter (ε_{loss}) = 0.00065 and tolerance for termination criterion (ε_{tol})= 0.0051. The respective parameter values for the active biomass concentration softsensor are: σ = 2.24, C = 11, ε_{loss} = 0.00001 and ε_{tol} = 0.00001.

The prediction and generalization performance of the two SVR-based softsensors was compared with that of the respective ANN-based softsensors. Here, a single hidden layer MLP architecture trained using the EBP algorithm was used to construct a multiple input - two output softsensor model. This single MLP model predicts the concentration values of both streptokinase and the active biomass. The training and test sets used in developing the MLP based softsensors were same as used in the development of SVR-based softsensors. The architectural details and the EBP specific parameter values that yielded an optimal MLP-based softsensor model are: number of input nodes = 8, number of neurons in the single hidden layer = 4, number of neurons in the output layer = 2, learning rate (η) = 0.5 and momentum coefficient = 0.05. An illustrative graphical comparison of the SVR and ANN model predicted values of streptokinase at the batch times of 2, 5, 7 and 12 hrs is depicted in Figure 3.10 and Figure 3.11, espectively. Similar comparison for the active biomass concentration is depicted in Figure 3.12and Figure 3.13, respectively. A comparison of the correlation coefficient, RMSE and average error (%) values pertaining to the predictions made by the SVR and ANN based softsensor models is given in Figure 3.12.

It is seen from the tabulated values of correlation coefficient, RMSE and average error that the respective magnitudes for the SVR and ANN-based streptokinase and active biomass softsensors match very closely with the SVR faring slightly better than the ANN. Also, the SVR-based softsensors are able to predict the streptokinase and active biomass concentrations with high prediction accuracy as indicated by the high CC magnitudes ($\geq$ 0.998) and low (1.4% to 1.6%) average error values. Moreover, a close match between the CC, average

error and RMSE values for both training and test sets indicates an excellent generalization performance by the SVR based softsensors. Thus, the results of case study-II also indicate that the SVR is an attractive alternative to ANNs for the softsensor development.

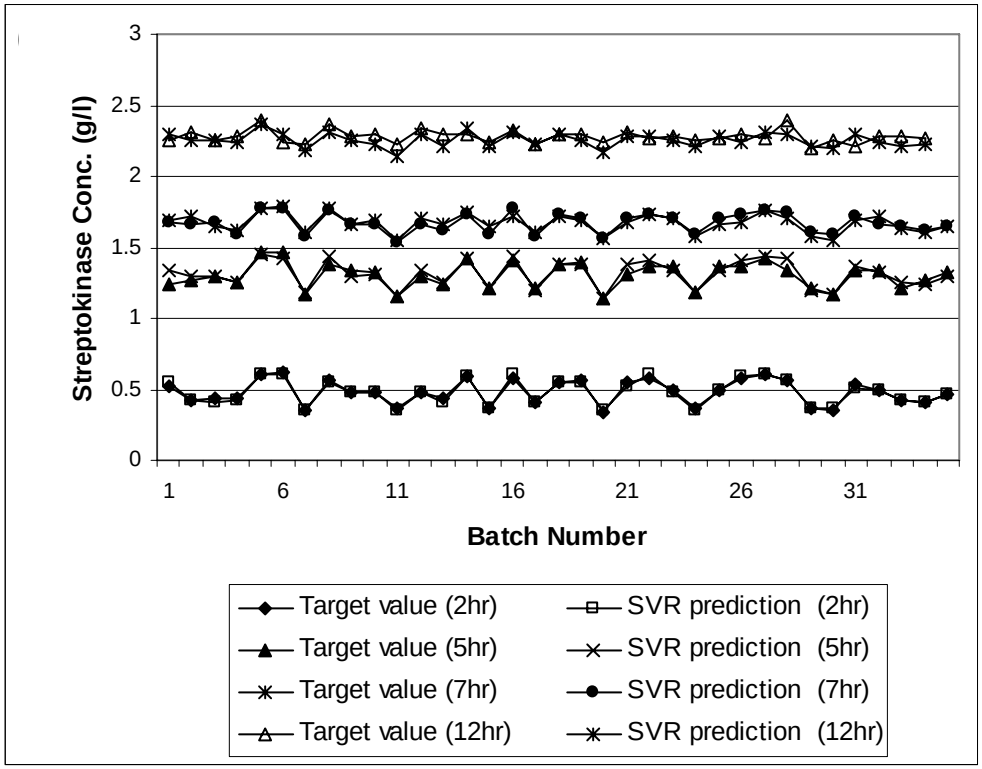

Figure 3.10: Streptokinase concentration at two, five, seven and twelve hour time duration as predicted by the SVR-based softsensor

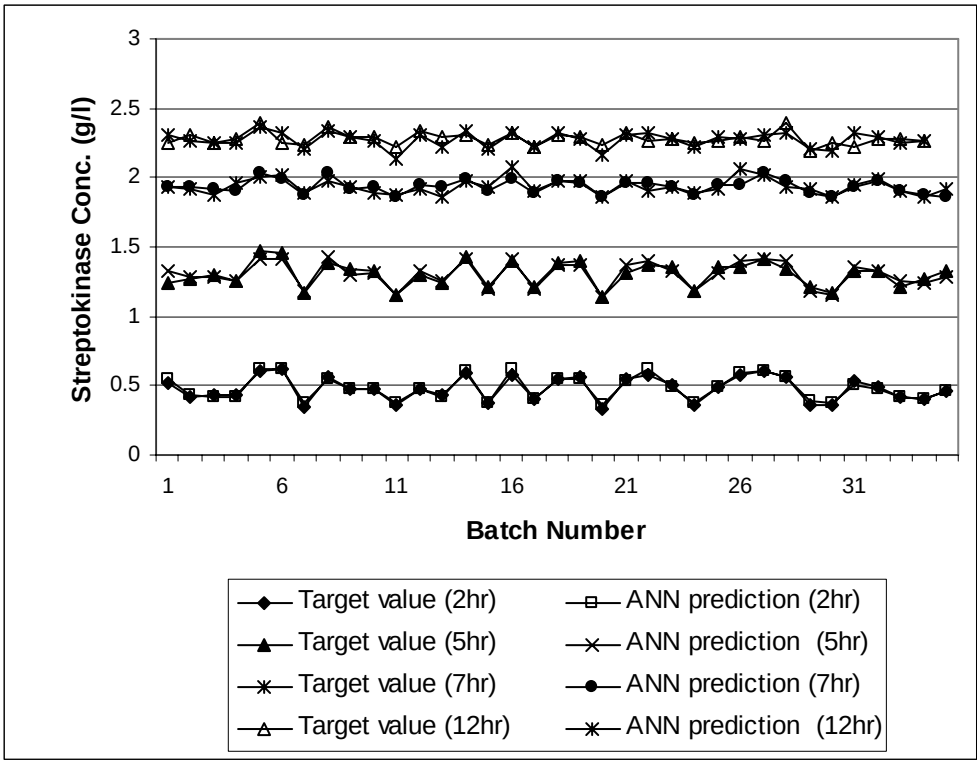

Figure 3.11: Streptokinase concentration at two, five, seven and twelve hour time duration as predicted by the ANN-based softsensor

Table 3.12: Comparison of prediction performance of SVR and ANN based softsensors

Softsensor	Method	Data	Correlation Coefficient (CC)	Average error (%)	RMSE
Streptokinase	SVR	Training set	0.999	1.592	0.0329
		Test set	0.998	1.625	0.0401
	ANN	Training set	0.998	1.659	0.0312
		Test set	0.997	1.726	0.0315
Active biomass	SVR	Training set	0.999	1.385	0.0743
		Test set	0.999	1.538	0.0719
	ANN	Training set	0.998	1.537	0.0743
		Test set	0.997	1.887	0.0779

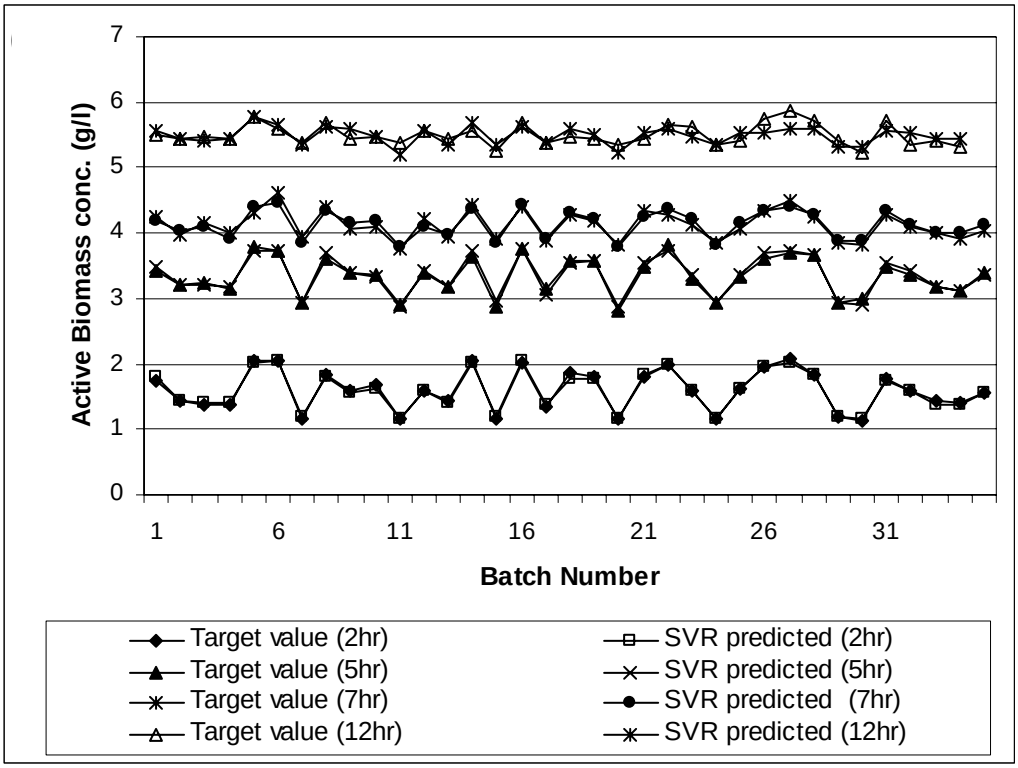

Figure 3.12: Active biomass concentration at two, five, seven and twelve hr. time duration as predicted by the SVR-based softsensor

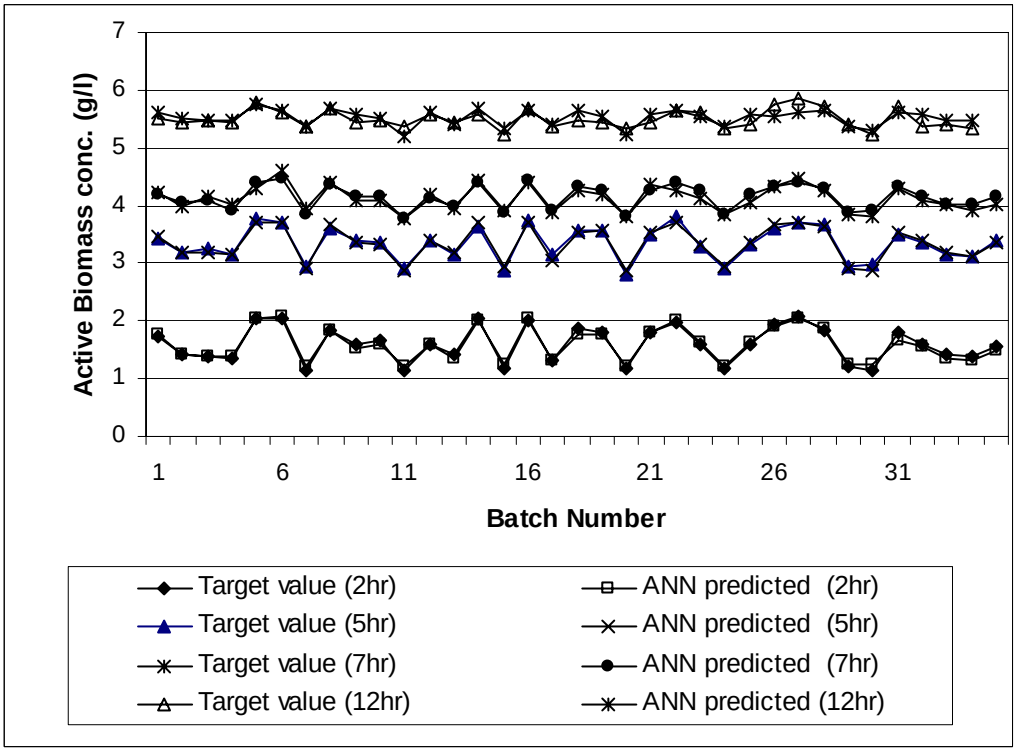

Figure 3.13: Active biomass concentration at two, five, seven and twelve hr. time duration as predicted by the ANN-based softsensor

3.5.6 Conclusion

The microbial fermentation is used extensively in the production of antibiotics, proteins, polysaccharides, amino acids, etc. These fermentation systems often exhibit a complex non-linear dynamical behavior, which poses significant difficulties for process monitoring and control. An additional problem in the efficient monitoring and control of fermentation processes is in most situations reliable hardware sensors or biosensors for the product species are not available. In such cases, the product concentration/quality is determined using time-consuming and tedious instrumental and/or chemical analyses. The usage of softsensors overcomes this difficulty. Softsensors are software based sophisticated monitoring systems, which can correlate the "difficult-to-measure" or unmeasured process variables with those measured easily and frequently. Artificial neural networks owing to their significant ability of approximating nonlinear functions have been the primary candidates for the development of softsensors. In recent years, a novel statistical/machine learning theory based formalism known as "support vector regression" has been introduced for performing non-linear function approximation. The SVR formalism has many attractive features such as: robustness of solution, sparseness of regression, good generalization capability and automatic control of solution complexity. Hence, in the present section the effectiveness of the SVR formalism for softsensor applications was evaluated by considering two simulated fed-batch processes namely, invertase and streptokinase.

The characteristic feature of the diauxic yeasts that the rate of glucose and ethanol assimilation changes with variations in the metabolic state, has been exploited for developing the SVR based soft-sensor estimating the invertase activity. This softsensor could estimate the invertase activity with accuracies of 96.7% and 96.3% for the training and test sets, respectively. In the case of genetically modified microorganisms, the feature that the plasmid-bearing and plasmid-free cells follow different substrate and biomass profiles, has been utilized to develop the SVR-based softsensors estimating concentrations of the protein (streptokinase) and active biomass. Here again the estimation accuracy of the SVR based streptokinase and active biomass softsensors was found to be

excellent ranging approximately between 98.4% and 98.6%. In both the case studies, the generalization performance of the SVR-based softsensors also was excellent. Additionally, the performance of the SVR-based softsensors was compared with the ANN-based ones and the results obtained thereby clearly indicate that the SVR formalism is an attractive alternative to ANNs for softsensor applications. In the SVR-based modeling, the search for the globally optimum solution is avoided since—unlike ANNs—the SVR solves a quadratic programming problem possessing a single minimum. This feature considerably reduces the numerical effort involved in developing an SVR-based softsensor. Also, SVR-based models are amenable to interpretation in contrast to the "black box" ANN models. Although in this study, the SVR formalism has been explored only for softsensor applications, it is a generic nonlinear modeling formalism and therefore can also be utilized efficiently for developing steady-state and dynamic models of bioprocesses.

3.6 SUPPORT VECTOR REGRESSION FOR BIOPROCESS IDENTIFICATION*

3.6.1 Introduction

Majority of biochemical processes are nonlinear in nature. Modeling and identifying these processes is one of the important tasks in the industrial practice. These tasks are required to build better model–predictive controls (MPC), and prediction and optimization of the process behavior for which phenomenological as well as empirical techniques are available. However, many times developing a phenomenological model for a given nonlinear biochemical process becomes tedious and costly because of the process complexity, lack of sufficient phenomenological knowledge of the system and huge amount of resources required for acquiring the knowledge. Owing to these difficuilties, empirical modeling and process identification techniques such as ANNs, nonlinear autoregressive moving average and genetic programming need to be used.

Recently developed nonlinear modeling technique known as "Support Vector Regression" (refer Section 2.2.2) provides a promising tool for process identification of nonlinear processes. Unlike other data-based modeling methods such as ANNs, which approximate nonlinear input-output relationships, the SVR method first projects inputs to a high dimensional feature space and then correlate them linearly with the target space. Advantages of the SVR technique are: strong statistical background, globally optimum solution (via quadratic function optimization) and good generalization performance.

In the present study, SVR formalism has been used to identify the process involving biological treatment of polluted water by a mixed continuous culture. A culture involving *Colipidium campylum* (protozoa) and *Alcaligenes faecalis* (bacteria) is used for the biological treatment of the polluted water containing Asparagine. A software library, LIBSVM [Chang et al., 2002] which includes

---

* Badhe Y. P., J. Singh Cheema, M. Potdar, S. S. Tambe and B. D. Kulkarni, *BIOHORIZON*, held at IIT, New Delhi, 2003.

sequential minimal optimization (SMO) algorithm [Platt, 1998] for solving the quadratic optimization problem is used for developing the SVR-based model. The process input-output data for the SVR-based identification were generated by simulating the phenomenological process model and identification was performed using both noise-free and noisy process data. The results obtained here show that the SVR is a promising technique for process identification in the presence and absence of noise in the process data.

3.6.2 Biological Treatment of Polluted Waters by Mixed Continuous Culture

The activated sludge used in the biological treatment of polluted water is a typical ecosystem composed of bacteria, protozoa, fungi and metazoa. Interactions such as competition, commensalism, mutualism (symbiosis), synergism and predation occur amongst the microorganisms in the ecosystem and as a consequence, BOD, COD or other nutrients are removed from polluted waters [Sudo, 1984].

Mixed continuous culture involves *Colipidium campylum* (protozoa) and *Alcaligenes faecalis* (bacteria) to remove the pollutants in polluted waters that contained Asparagine. The population densities of *Colpidium campylum* (*P*), *Alcaligenes faecalis* (*X*), and the concentration (*S*) of residual asparagine in the effluent can be ynamically monitored and represented in the following differential equations:

$$\frac{dS}{dt} = D\,(S_0 - S) - \frac{F\,(S,X)}{Y} \tag{3.41}$$

$$\frac{dX}{dt} = F\,(S,X) - D\,X - \frac{G\,(X,P)}{W} \tag{3.42}$$

$$\frac{dP}{dt} = G\,(X,P) - D\,P \tag{3.43}$$

$$\text{where, } F\,(S,X) = \mu_m \frac{S}{K+S}\ X \tag{3.44}$$

$$G\,(X,P) = \mu_{p,max} \frac{X}{K_x + X}\ P \tag{3.45}$$

where t refers to the time and S_0 , D, Y and W are the initial asparagine concentration, dilution rate, yield factor of bacterial growth defined by $\Delta P/ (-\Delta X)$

and the yield factor of protozoan growth defined by $\Delta P/(-\Delta X)$, respectively. Parameters, $\mu_m, \mu_{p,\max}, K_x$ and K refer to the maximum value of μ, maximum value of μ_p where μ_p is the specific growth rate of protozoa, saturation constant for protozoa and saturation constant for bacteria, respectively [Sudo, 1984]. The steady state values of the variables and parameters appearing in Eqs. (3.41) to (3.45) are given in Table 3.13.

Table 3.13: Steady-state values of variables and parameters

Variable/ Parameter	Unit	Value
X	mg/ l	10
P	mg/ l	5
S	mg/ l	5
K_x	mg/ l	5
K	mg/ l	11
D	h^{-1}	0.064
μ_m	h^{-1}	0.1
$\mu_{p,max}$	h^{-1}	0.1
Y	---	0.5
W	---	0.15
h	---	0.1

For identification, the process at steady-state was perturbed randomly. Specifically the manipulated variables representing the dilution rate, D, was perturbed as shown in Figure 3.14 and its response on the control variable, S, was monitored (see Figure 3.15). The data was generated by simulating (integrating) model Eqs. 3.41 – 3.45, using fourth order Runge- Kutta method using the step size (h = 0.1) upto the time of 700 hours. After scaling the D and S data between upper and lower limits of 0.95 and 0.05 respectively, they were divided into *training*, *test* and *validation* sets. The training data were subjected to SVR based identification and the optimal model obtained thereby was used to predict the *test* and *validation* set outputs. The same procedure was repeated for the noise-induced data.

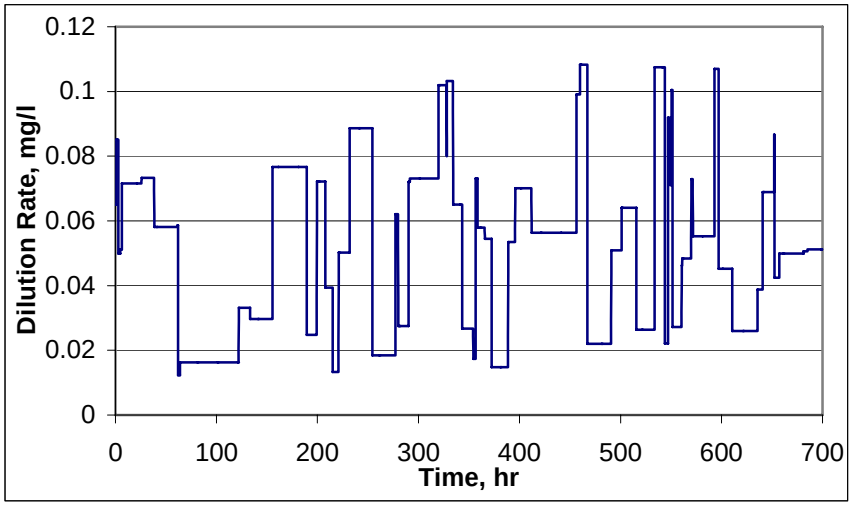

Figure 3.14: Random variations in the manipulated variable, *D*

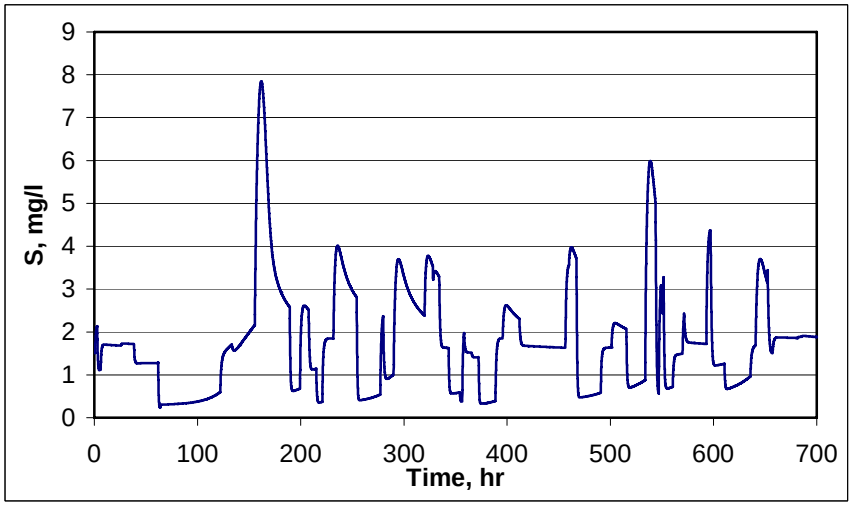

Figure 3.15: Response of S to random variations in *D*

3.6.3 Results and Discussion

The ε-SVR formalism was used to develop a "one-step ahead" predictor model for the controlled variable, *S*, using variable time lags for the manipulated and the controlled variables. Clean data as well as white-noise induced data were used to test the performance of the SVR-based models. The noise-induced data contained 5% of white noise in both *D* and *S* data. The one step ahead prediction results for the training, test and validation data sets pertaining to the noise-free (clean) and noisy process data are presented in the graphical form in Figure 3.16 and Figure 3.17, respectively.

From these figures it can be seen that the SVR methodology fits the noise-free and noisy one step ahead control variable data with an excellent precision. The *correlation coefficient* (CC) and the RMSE (*Root Mean Square Error*) values for the predicted results are given in Table 3.14. As can be seen the CC values for the desired and model-fitted one-step-ahead concentration of the control variable are very close to unity signifying an excellent fit. Also, the RMSE values are very low further supporting the excellent fit by the SVR model.

Table 3.14: Prediction results from ε- SVR based models

Data Sets	Clean Data		Noisy Data	
	RMSE	**CC**	**RMSE**	**CC**
Training data	0.015	0.997	0.063	0.993
Test data	0.016	0.994	0.059	0.993
Validation data	0.022	0.992	0.076	0.991

The SVR parameters used in simulations are: Cost (upper bound on Lagrange multipliers) = 10, ε (tolerence for the ε insensitive loss function) = $1*10^{-4}$, gamma (radial basis function's width factor) = 0.1 and error tolerance criteria = $1*10^{-4}$.

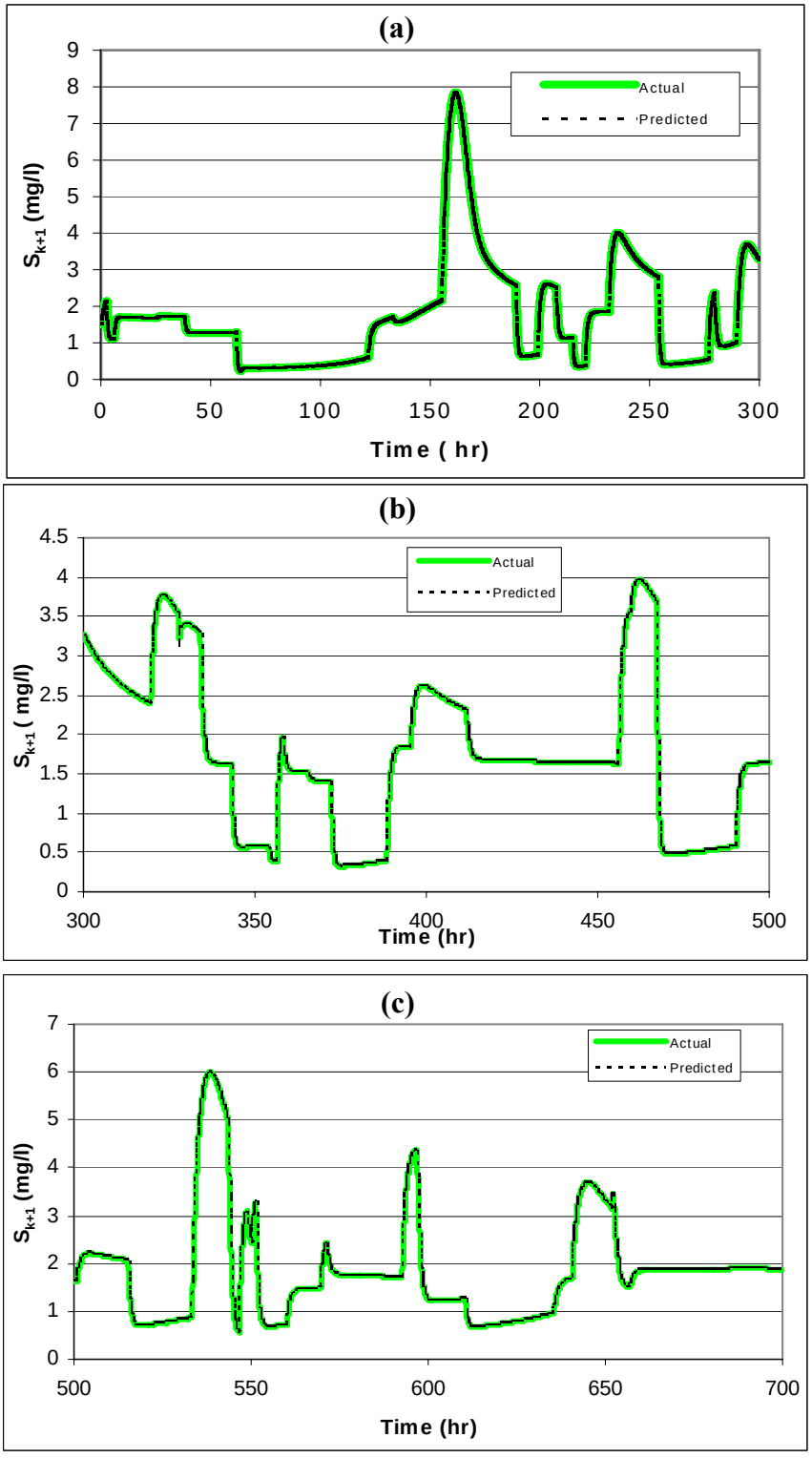

Figure 3.16: SVR predicted and desired S_{k+1} values for (a) training, (b) test and (c) validation data sets

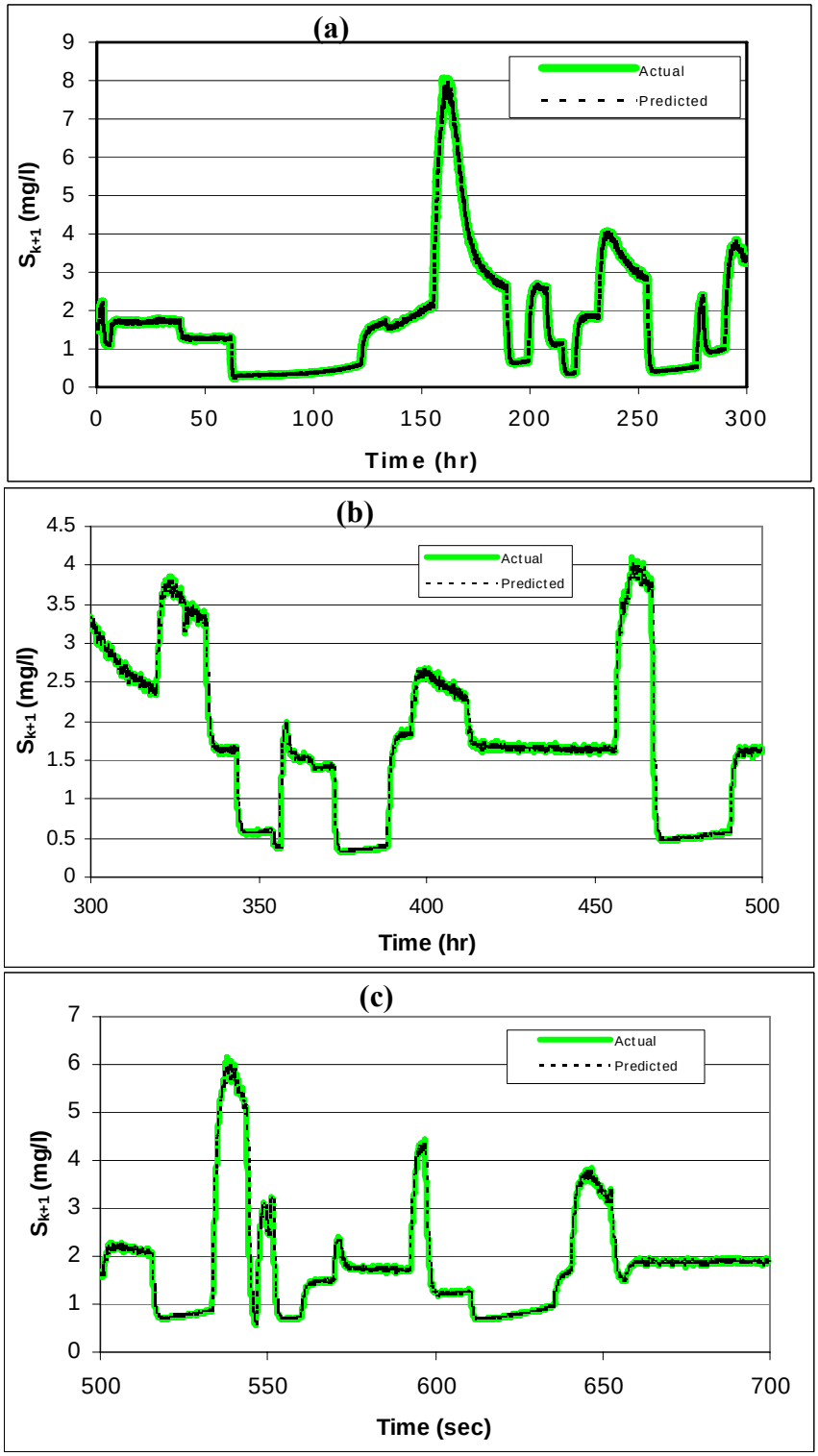

Figure 3.17: SVR predicted and actual S_{k+1} values for noisy (a) training, (b) test & (c) validation data set

3.6.4 Conclusion

In this study, Support Vector Regression has been successfully employed for the process identification of biological treatment of polluted waters using mixed continuous culture. The results obtained here show that the SVR methodology is a promising tool for nonlinear process identification. The SVR formalism has been found to perform excellently for clean as well as noisy process data.

3.7 GENETIC PROGRAMMING FOR DATA-DRIVEN MODELING OF NON-LINEAR CHEMICAL PROCESSES*

Genetic Programming, described in Section 2.2.3, is an emerging branch of artificial intelligence. In recent years, the GP formalism has found a novel application; that is, development of data-driven models. Specifically, the technique is capable of automatically obtaining the mathematical equation that fits a given set of process input-output data. The major advantage of GP is that it does not require specification of an exact form of the data-fitting function, as searches and optimizes the exact form of the best-fitting functional form and its parameters. The present work illustrates the case study involving GP-based modeling of benzene isopropylation over Hbeta catalyst process.

3.7.1 Modeling of Benzene Isopropylation Over Hbeta Catalyst Process

Isopropylation of benzene is an important alkylation reaction in the petrochemical industry for the synthesis of cumene, which is the chief starting material in phenol production. In the last decade, several modifications of the zeolite beta were explored as potential catalysts in cumene synthesis. More recently, steady-state modeling of isopropylation of benzene over Hbeta (protonic form of beta catalyst) is presented in Nandi et al. [2002]. The details of benzene isopropylation over Hbeta catalyst process are elaborated in Section 3.3.4.

3.7.2 Results and Discussion

The data from 42 experiments comprising selectivity and percentage yield (see Table 3.15) was randomly split into two sets namely, *training* and *test* data sets in 80:20 ratio, respectively. The training data were used to build the GP model and the test data were used to evaluate the generalization ability of the

---

* Phulwale U. S., Badhe Y. P., Mandge D. P., Tambe S. S., Kulkarni B. D., Poster Presented at NCL Day, NCL, Pune, 2005.

model. The GP software used in this study has been developed in-house at NCL and utilized to developed two models predicting cumene yield and selectivity, respectively. The two GP-based non-linear models (expression trees) for the prediction of cumene yield and selectivity are as given below.

$$y_1 = \left[\sin\left(\sin\left(\sin\left(x_3\right)\right) + \sin\left(\sin\left(x_1\right)\right)\right)\right]^{\left(0.064*\sin\left(x_1\right)\right)^{x2} - 0.3} \tag{3.46}$$

$$y_2 = \left(\sin\left(\left(\left(x_3 + x_2\right) + e^{x_4}\right) * \left(x_1 - 0.05\right)\right)\right)^{\cos\left(\sin\left(x_2 + 0.21\right)\right)} \tag{3.47}$$

where, y_1 and y_2 refer to selectivity and yield of cumene, respectively and x_1, x_2, x_3 and x_4 refer to temperature (^oC), pressure (atm), mole ratio of benzene to isopropyl alcohol (mole ratio) and weight hourly space velocity (hr^{-1}), respectively.

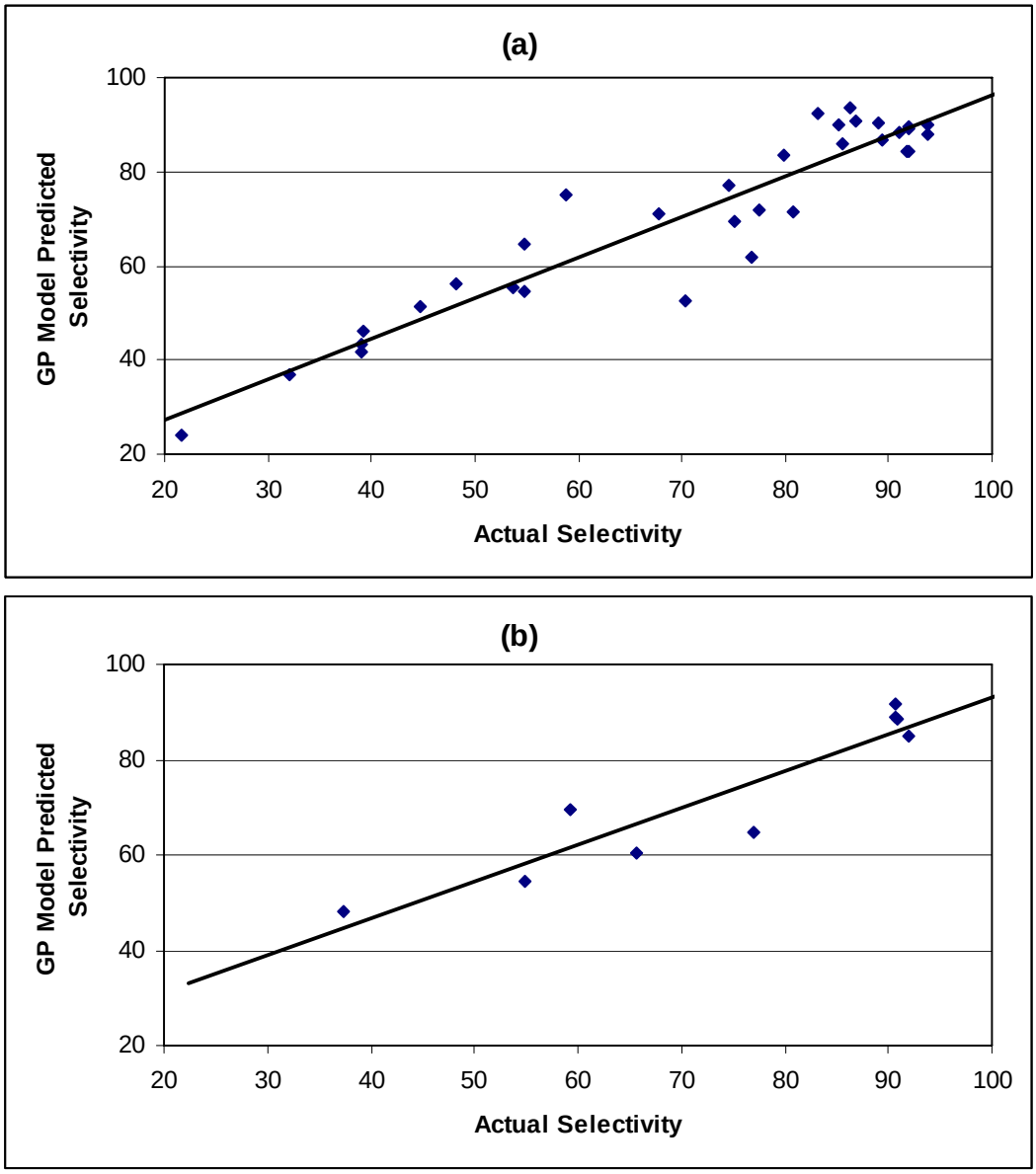

Figure 3.18: GP model predictions of cumene selectivity: (a) Plot of training data (b) Plot of test data

Table 3.15: Benzene isopropylation over Hbeta catalyst process data

Expt. No.	Temperature (^{0}C)	Pressure (atm.)	Benz/IPA (mole ratio)	WHSV (hr^{-1})	Yield (wt %)	Selectivity (wt %)
1	110	1	8	3.3	0.07	77.03
2	145	1	8	3.3	11.6	58.75
3	180	1	8	3.3	15.78	79.93
4	210	1	8	3.3	17.365	90.72
5	215	1	8	3.3	16.09	91.95
6	150	4	8	3.3	12.2	65.74
7	135	4	8	3.3	12.99	74.58
8	110	4	8	3.3	0.71	80.82
9	100	4	8	3.3	0.19	75.02
10	110	1	10	3.3	0.55	67.74
11	110	1	8	3.3	0.24	54.85
12	110	1	6	3.3	0.37	53.63
13	110	1	3	3.3	0.2	32.13
14	110	1	1	3.3	0.14	21.62
15	110	1	8	6.8	0.24	54.85
16	110	1	8	8	0.15	44.64
17	110	1	8	9.5	0.13	37.38
18	110	1	8	10.5	0.08	39.3
19	110	1	8	12	0.09	39.13
20	110	1	8	13	0.07	39.1
21	105	1	8	6.8	0.3	70.38
22	110	1	8	6.8	0.24	54.85
23	115	1	8	6.8	0.35	48.25
24	130	1	8	6.8	4.61	76.68
25	185	1	8	6.8	9.2	59.23
26	210	1	6.5	3.3	20.04	91.8
27	155	1	6.5	3.3	16.93	77.4
28	180	1	6.5	3.3	20.27	90.9
29	210	1	6.5	3.3	19.86	91.9
30	225	1	6.5	3.3	19.1	89.3
31	250	1	6.5	3.3	17.89	85.2
32	275	1	6.5	3.3	17.29	83.1
33	230	1	6.5	2.5	20.33	91.1
34	215	1	7	5	19.86	91.9
35	215	10	7	5	19.54	92
36	215	18	7	5	18.68	89.1
37	215	25	7	5	17.74	86.8
38	195	25	6	5	18.92	85.6
39	210	25	6	5	22.1	93.7
40	230	25	6	5	22.02	93.8
41	250	25	6	5	21.35	90.7
42	280	25	6	5	20.48	86.2

Table 3.16 shows the performance of the two GP models (Eqs. 3.46 and 3.47) in predicting the output in the training and test data sets in terms of the correlation coefficient and the root mean squared error (RMSE). Also, the plots of the training and test set output predictions are shown in Figure 3.17 and Figure 3.18, respectively. The GP model for the cumene yield has achieved CC values of 0.98 and higher which shows that the model has good prediction accuracy as also generalization ability. The GP model for the cumene selectivity possesses correlation coefficient values of 0.94 and 0.92 for the training and test data sets, respectively. These values though not excellent, are reasonably good suggesting average prediction accuracy and generalization performance by the model. These results are similar to the earlier ANN modeling work conducted by Nandi et al., [2002]. The origin of sub-optimally performing selectivity model could be measurement errors in the experimental data.

Table 3.16: Perfromance of the GP-based models

Models		**Correlation Coefficient**	**RMSE**
Selectivity (y_1)	Training data	0.94	7.06
	Test data	0.92	7.19
Yield (y_2)	Training data	0.99	1.42
	Test data	0.98	2.61

3.7.3 Conclusion

This case study explores the GP's potential in performing exclusively data-based non-linear process modeling. While the GP has produced an accurate model for the cumene yield, the model predicting cumene selectivity is an average one probably due to measurement errors in the experimental data. As can be seen, GP posses an excellent capacity to provide nonlinear models exclusively from the process data. The major advantage of GP formalism is that it does not require guessing in advance the form of the data-fitting function. The method is capable of searching and optimizing both the fitting function as also its parameters automatically.

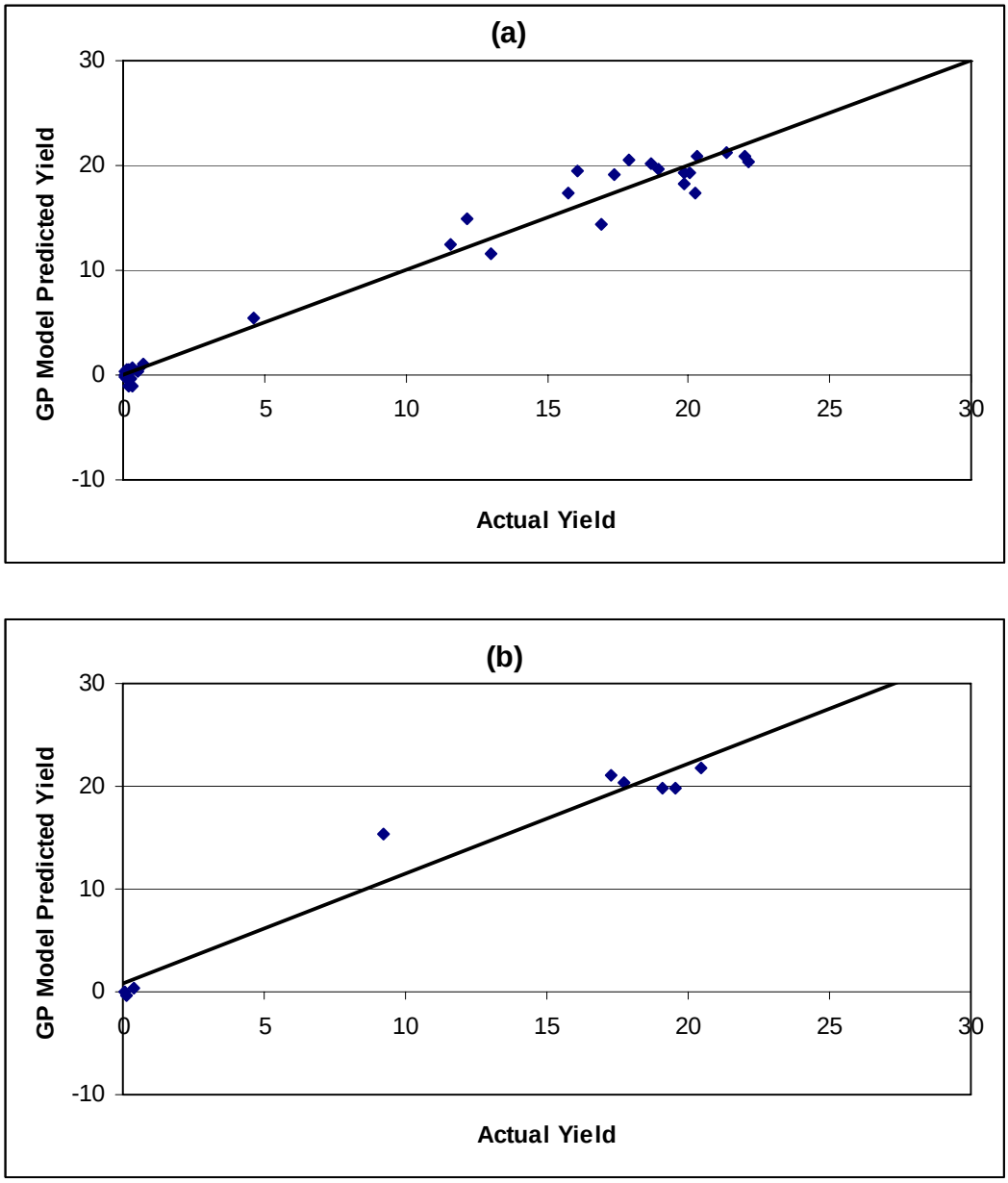

Figure 3.19: GP model predictions for cumene yield. (a) Plot of training data (b) Plot of test data

3.8 REFERENCES

1. Acha, V., M. Meurens, H. Naveau, D. Dochain, G. Bastin and S. N. Agathos, Model-based estimation of a reductive dechlorination process via an Fourier transform infrared sensor, *Wat. Sci. Tech.* 8 (1999) 33–40.
2. Adilson, J. D. A., M. Rubens, Soft-sensors development for on-line bioreactor state estimation. *Comp. Chem. Eng.* 24 (2000) 1099–1103.
3. Agarwal, M. (1997), A Systematic Classification of Neural-Network-Based Control, *IEEE Control Sys.*, 17(2), 75–93.
4. Albert, S. and R. D. Kinley, Multivariate statistical monitoring of batch processes: an industrial case study of fermentation supervision. *Trends Biotech.*, 2 (2001) 53–62.
5. An, G. (1996), The Effects of Adding Noise during Backpropagation Training on a Generalization Performance, *Neural Computation*, 8, Issue 3, 643–674,.
6. Bastin, G. and D. Dochain (1990), *On-line estimation and Adaptive control of Bioreactors*. Elsevier, Amsterdam.
7. Beejherk, H. E., R. J. A Hall (1977), mechanistic model of aerobic growth of Saccharomyces cerevisiae. *Biotechnol. Bioeng.*, 19, 267–296
8. Bhat, N., and T. McAvoy (1990), Use of Neural Nets for Modeling and Control of Chemical Process Systems, *Comp. Chem. Eng.*, 14, 573.
9. Bishop, C. M. (1994), Neural Networks and Their Applications, *Rev. of Sci. Instru.*, 65, 1803.
10. Cavani, F., G. Girotti, V. Arrigoni and G. Terzoni (1997), Alkylation catalyst for aromatic compounds for lower olefins, *US patent* 5650547 A, July 22.
11. Channiwala S. A., Parikh P. P. (2002), A unified correlation for estimating HHV of solid, liquid and gaseous fuels. *Fuel*, 81, 1051–1063.
12. Chih-Chung Chang, Chih-Jen Lin (2002), LIBSVM: a Library for Support Vector Machines. http://www.csie.ntu.edu.tw/~cjlin
13. Choudhary A., Biswas S. (2002–03), CFRI internal report titled Development of equivalency chart between UHV and GCV, Report No. TR/CFRI/3.02/2002–03.

14. Cordero T., Marquez F., Rodriquez-Mirasol J., Rodriguez J. J. (2001), Predicting heating values of lignocellulosic and carbonaceous materials from proximate analysis. Fuel, 80, 1567–1571.
15. Davis L. (Ed.) (1987), *Genetic Algorithms and Simulated Annealing*, London: Pitman.
16. Dedem, G. V., M. A. Moo-Young (1975), Model for diauxic growth. *Biotechnol. Bioeng.*, 17, 1301–1312.
17. Demirbas A. Calculation of higher heating values of biomass fuels. *Fuel* 1997; 76(5):431–34.
18. Desai, K., Y. P. Badhe, S. S. Tambe, B. D. Kulkarni (2006), Soft-sensor Development for Bioreactors Using Support Vector Regression, *Biochem. Engg. J.*, 27(3), 225–239.
19. Eerikäinen, T., P. Linko, S. Linko, T. Siimes, Y. H. Zhu, Fuzzy logic and neural network applications in food science and technology. *Trends Food Sci. Technol.*, 4 (1993) 237–242.
20. Engelbrecht, A. P., Cloete, I., and Zurada, J. M. (1995), Determining the significance of input parameters using sensitivity analysis, From natural to artificial neural computation: *Proceedings of International Workshop on Artificial Neural Networks*. Malaga-Torremolinos, Spain: Springer, 382–388.
21. Fernandez P, Diaz RM, Xiberta J. (1997), Correlations of properties of Spanish coals with their natural radionuclides contents Fuel, 76(10), 951–955.
22. Folly, R. O. M., B. Valdman, F. Valero, C. Solá (1996), Potentiometric sensor for online glucose determination. *Biotechnol. Techniques*, 11, 867–870.
23. Freeman, J. A., and D. M. Skapura (1991), *Neural Networks : Algorithms, Applications, and Programming Techniques*, Addison-Wesley, Reading, MA.
24. Geatti, A., M. Lenarda, L. Storaro, R. Ganzerla and M. Perissinotto (1997), Solid acid catalysts from clays: cumene synthesis by benzene alkylation with propene catalyzed by cation exchanged aluminium pillared clays, *J. Mol. Catal A: Chem.*, 121, 111–118.
25. Georgiou, G., J. J. Chalmers, M. Shuler, D. B. Wilson (1985), Continous immobilized recombinant protein production from E. coli capable of selective protein excretion: A feasibility study. *Biotechnol. Prog.*, 1, 75–79.
26. Goldberg, D. E. (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison–Wesley, New York.

27. Goutal, M. (1902), *Acad. Sci.* Paris, 135: 477–479;
28. Hernandez, E. and Y. Arkun (1992), Study of the Control-Relevant Properties of Backpropagation Neural Network Models of Nonlinear Dynamical Systems, *Comp. Chem. Eng.*, 16**,** Issue 4, 227–240.
29. Holland, J. H. (1975), *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI: Univ. Mich. Press.
30. Holmstorm, L. and Koistimen, P. (1992), Using additive noise in back-propagation training, *IEEE transactions on neural networks*, 3(1), 24–38.
31. Hunt, K., D. Sbarbaro, R. Zbikowski, and P. Gawthrop (1992), Neural Networks for Control Systems – A Survey, *Automatica*, 28, 1083.
32. James, S., R. Legge, H. Buddman (2002), Comparitive study of black box and hybrid estimation methods in fed batch fermentation. *J. Process Control*, 12, 113–121.
33. Jimenez L, Gonzalez F. (1991), Study of the physical and chemical properties of lignocellulosic residues with a view to the production of fuels. *Fuel*; 70: 947–50.
34. Joachims, T. (1998), Making large-scale SVM learning practical, in: B. Schölkopf, C.J.C. Burges, A.J. Smola (Eds.), *Advances in Kernel Methods—Support Vector Learning*, MIT Press, Cambridge, MA.
35. Karagoz, O., J. Versteeg, M. Mercer and P. Turner (2004), Advanced Control Methods Improve Polymers' Business Cycle, *Hydrocarbon Processing*, 83 (4), 45–49.
36. Karim, M. N., S. L. Rivera (1992), Comparison of feed-forward and recurrent neural networks for bioprocess state estimation. *Comp. Chem. Eng.*, 16, 369–377.
37. Kucukbayrak S, Durus B, Mericboyu AE, Kadioglu E. (1991), Estimation of calorific values of Turkish lignites. *Fuel*, 70, 979–981.
38. Kulkarni, B. D., S. S. Tambe, J. B. Lonari, N. K. Valecha, S. V. Deshmukh, B. S. Shenoy, and S. Ravichandran (2002), Performance of artificial neural network models in the presence of instrumental noise and measurement error, *U. S. Patent* (USPA 20030191728) filed on March 27, 2002.
39. Kulkarni, B. D., S. S. Tambe, R. K. Dahule, and R. K. Yadavalli (Jul 1999), National Chemical Laboratory, Pune, India. *Hydrocarbon Processing*, 89–97.
40. Lee, S. B., J. E. Baily (1984), Analysis of growth rate effects on productivity

of recombinant E. coli populations using molecular mechanisms model. *Biotechnol. Bioeng.*, 26, 66.

41. Lin Y. and G. Cunningham (1995), A new approach to fuzzy-neural system modeling, *IEEE Trans. Fuzzy Systems* 3, 190–198.
42. Lin Y. and G. Cunningham (1994), Building a fuzzy system from input-output data, *J. Intelligent Fuzzy Systems* 2, 243–250.
43. Linko, S., Yi-Hong Zhu, P. Linko (1999), Applying neural networks as software sensors for enzyme engineering. *Tibtech*, 17, 155–162.
44. Lui, H., H. Li, T. Ying, K. Sun, Y. Qin, D. Qi (1998), Amperometric biosensor sensitive to glucose and lactose based on co-immobilization of ferrocene, glucose oxidase, β-galctosidase and mutarotase in β-cyclodextrin polymer, *Annal. Chim. Acta,* 358, 137–144.
45. Marquardt, D.W. An algorithm for least squares estimation of nonlinear parameters. *J. Soc. Ind. Appl. Math.* 1963; 11: 431–441.
46. Mazumdar B. K. (1954), Coal systematics: Deductions from Proximate Analysis of Coal Part I. *Journal of Scientific & Industrial Research*, 13B(12): 8, 57–63.
47. Mazumder B. K. (2000), Theoretical oxygen requirement for coal combustion: relationship with its calorific value. *Fuel*, 79(14), 13–19.
48. Meima, G. R. (1998), Advances in cumene production, *CATTECH*, June, 5–12.
49. Modak, J. M., H. C. Lim, Y. J. Tayeb (1986), Genral characters of optimal feed rate profiles for various fed-batch fermentative process, *Biotechnol. Bioeng.*, 28, 1396–1407.
50. Montague, G. A., A. J. Morris, A. R. Wright, M. Aynsley, A. C. Ward (1986), Online estimation and adaptive control of penicillin fermentation. *IEE Proceedings*, 5, 240–246.
51. Nahas, E. P., M. A. Henson, and D. E. Seborg (1992), Nonlinear Internal Model Control Strategy for Neural Network Models, *Comp. Chem. Eng.*, 16, Issue 12, 1039–1057.
52. Nandi, S., S. Ghosh, S. S. Tambe, B. D. Kulkarni (2001), ANN-Assisted Stochastic Process Optimization Strategies, *AIChE J.*, 47, 126–135.

53. Nandi, S., Y. Badhe, J. Lonari, U. Shridevi, B. S. Rao, S. S. Tambe, B. D. Kulkarni (2004), Hybrid process modeling and optimization strategies integrating neural networks/support vector regression and genetic algorithms: study of benzene isopropylation on Hbeta catalyst. *Chem. Eng. J.*, 97, 115–129.
54. Narendra, K., and K. Parthasarathy (1990), Identification and Control of Dynamical Systems Using Neural Networks, *IEEE Trans. Neural Networks,* 1, 4.
55. Ohno, H., E. Nakanishini, and T. Takamastsu (1976), Optimal control of semibatch fermentation, *Biotechnol. Bioeng.* 17, 847–864.
56. Parikh J, Channiwala S. A, Ghosal G. K. (2005), A correlation for calculating HHV from proximate analysis of solid fuels. *Fuel*, 84(4), 87–94.
57. Patel, S. U., B. Jeevan Kumar, Y. P. Badhe, B. K. Sharma, S. Saha, S.Biswas, A. Chaudhury, S. S. Tambe and B. D. Kulkarni (2007), Estimation of gross calorific value of coals using artificial neural networks, *Fuel*, 86(3), 334–344.
58. Patnaik, P. R. (1995), A heuristic approach to fed-batch optimization streptokinase fermentation. *Bioprocess Eng*, 13, 109–12.
59. Patnaik, P. R. (1999), Improvement of the microbial production of streptokinase by ccontrolled filtering of process noise. *Process Biochemistry*, 35, 309–315.
60. Perego, C., G. Pazzuconi, G. Girotti and G. Terzoni (1994), Process for the preparation of cumene, *Eur. Pat.* Appl. EP629599 A1, Dec. 21.
61. Platt, J. C. (1998), Fast training of support vector machines using sequential minimal optimization, in: *Advances in Kernel Methods—Support Vector Learning*, B. Schölkopf, C.J.C. Burges, A.J. Smola (Eds.), MIT Press, Cambridge, MA.
62. Pons, M. N., A. Rajab, J. M. Flaus, J. M. Engasser (1988), A. Cheruy, Comparison of estimation methods for biotechnological processes. *Chem. Eng. Sci.*, 8, 1909–1914.
63. Pyun, Y. R., J. M. Modak, Y. K. Chang, H. C. Lim (1989), Optimization of biphasic growth of Saccharomyces carsbergensis in fed-batch culture, *Biotechnol. Bioeng.*, 33, 1–10.

64. Ramasamy, S., S. S. Tambe, B. D. Kulkarni, and P. B. Deshpande (1995), Robust Nonlinear Control with Neural Networks, *Proc. R. Soc. Lond. A.*, 449, 65.
65. Rank, M. J. Gram, K. S. Nielsen, B. Danielsson (1995), On-line monitoring of ethanol, accetaldehyde, and glycerol during industrial fermentations with Saccharomyces cerevisiea. *Appl. Microbiol. Biotechnol.*, 42, 813–817.
66. Raveendran K, Ganesh A. (1996), Heating value of biomass and biomass pyrolysis products. *Fuel*, 75(15), 1715–1720.
67. Rumelhart, D., G. Hinton, and R. Williams (1986), Learning Representations by Backpropagating Errors, *Nature*, 323, 533–536.
68. Ryan, W., S. J. Parulekar (1991), Recombinant protein synthesis and plasmid instability in continuous cultures of Escherichia coli JM103 harboring high copy number plasmid. *Biotechnol. Bioeng.*, 37, 415 –429.
69. Ryan, W., S. J. Parulekar, B. C. Stark (1989), Expression of *β*-lactamase by recombinant E-coli strains containing plasmid of different sizes- effect of pH, phosphate and dissolved oxygen. *Biotech. Bioeng.*, 34, 309–319.
70. Sachez, J. L., W. C. Robinson, H. Budman (1999), Developmental studies of an adaptive on-line softosensor for biological wastewater treatment. *Canadian J. Chem. Eng*., 77, 707–717.
71. Sarkar, D., J. M. Modak (2003), Optimization of fed-batch bioreactor using genetic algorithm. *Chem. Eng. Sci.*, 58, 2283–2296.
72. Schuster V. F. (1951), Über die Berechnung des Heizwertes von Kohlen aus der Immediatzusammensetzung. *Brennstoff – Chemie*, 32, 19–20.
73. Seo, J. H., J. E. Baily (1985), Effects of recombinant plasmid contains on growth properties and cloned gene product formation in E-coli. *Biotechnol. Bioeng.*, 27, 1668.
74. Shimizu, K. (1996), A tutorial review on bioprocess systems engineering. *Comp. Chem. Eng.*, 6/7, 915–941.
75. Sietsma, J., R. J. Dow (1991), Creating artificial neural networks that generalize, *Neural Networks*, 4, 67–79.
76. Spooner, C.E. 1951, Swelling power of coal. *Fuel*, 30:193–202.
77. Sridevi, U., B. K. B. Rao, N. C. Pradhan, S. S. Tambe, C. V. Satyanarayana and B. S. Rao (2001), Kinetics of isopropylation of benzene over Hbeta catalyst, *Ind. Engg. Chem. Res.*, 40, 3133–3138.

78. Staniskis, J., D. Levisauskas (1984), An adaptive control algorithm for fed batch culture. *Biotechnol. Bioeng.*, 26, 419–425.
79. Sudo, R., S. Aiba (1984), Role and Function of Protozoa in the Biological Treatment of Polluted Waters in *Advances in Biochemical Engineering/ Biotechnology* Volume 29' A. Flechter (Ed.), Springer-Verlag, pg. 117–141.
80. Sung A. H. (1998), Ranking importance of input parameters of neural networks. *Expert Systems with Applications*, 15, 405–411.
81. Tambe, S. S., B. D. Kulkarni, and P. B. Deshpande (1996), *Elements of Artificial Neural Networks with Selected Applications in Chemical Engineering, and Chemical & Biological Sciences*, Simulation & Advanced Controls Inc., Louisville, USA.
82. Tendulkar, S. B., S. S. Tambe, I. Chandra, P. V. Rao, R. V. Naik, and B. D. Kulkarni (1998), Hydroxylation of Phenol to Dihydroxybenzenes: Development of Artificial Neural–Network-Based Process Identification and Model Predictive Control Strategies for a Pilot Plant Scale Reactor, *Ind. Eng. Chem. Res.*, 37, 2081.
83. Toda, K., I. Yabe and T. Yamagata (1980), Kinetics of biphasic growth of yeast in continuous and fed batch culture. *Biotechnol. Bioeng.*, 22, 1805.
84. Zhu, Y. H., T. Rajalahti, S. Linko (1996), Application of Neural network to lysine production. *Biochem. Eng. J.*, 62, 207–214.
85. Zurada, J. M., Malinowski, A., and Cloete, I. (1994), Sensitivity analysis for minimization of input data dimension for feed forward neural network, *Proceedings of IEEE International Symposium on Circuits and Systems*, London: IEEE Press, 4, 47–50.

CHAPTER 4

APPLICATIONS OF AI-BASED CLASSIFICATION/CLUSTER ANALYSIS

4.1 INTRODUCTION

Cluster analysis is an exploratory data analysis method for solving classification problems. Its object is to sort cases (people, things, events, etc.) into groups or clusters, so that the degree of association is strong between members of the same cluster and weak between members of different clusters. Each cluster thus describes, in terms of the data collected, the class to which its members belong; and this description may be abstracted through use from the particular to the general class or type.

Cluster analysis is thus a tool of discovery. It may reveal associations and structure in a given data set, which though not previously evident, nevertheless are sensible and useful once, found. The results of a cluster analysis may contribute to the definition of a formal classification scheme with which to describe populations or indicate rules for assigning new cases to classes for identification and diagnostic purposes. It can also provide measures of definition, size and change in what previously were only broad concepts or find exemplars to represent classes. Clustering can be performed in two modes, namely "supervised" and "un-supervised" modes. In supervised clustering the classification of the data is known a priori and the task of a supervised clustering algorithm is to learn this classification and predict the class of a new data. In unsupervised clustering the data classes are unknown. Here the task of an unsupervised algorithm is to categorize the data into appropriate number of classes as also to correctly identify the class of a new data entity. As can be seen, unsupervised clustering is relatively difficult task when compared with supervised clustering.

This chapter deals with the cluster analysis of the different faults that can occur in a batch fermentation process involving protein synthesis and citric acid production. Also, this chapter illustrates application of an artificial intelligence based clustering (classification) method, namely, "self organizing map (SOM)" for the classification of biochemical batch process data. The SOM [Kohonen, 1990, Abonyi, et al., 2003] is an ANN that undergoes unsupervised learning and it is particularly useful in visualizing high dimensional data onto a two-dimensional surface. The present study aims at demonstrating the efficacy of SOM for

classification applications involving nonlinear projection of a high dimensional input space onto a low, i.e., two dimensional (2-D) projected space to diagnose faulty batches. Here, a case study involving the biosynthesis of protein has been conducted to illustrate SOM's efficacy in process monitoring and fault detection and diagnosis applications.

4.2 MONITORING AND FAULT DETECTION OF A BATCH FERMENTATION PROCESS USING SELF ORGANIZING MAPS*

Batch processes are characterized by flexible, unsteady, and finite-duration operation. Also, most of these processes exhibit nonlinear dynamical behavior wherein the product is analyzed after the batch completion. The product quality measurements, if feasible and economical, are also done at finite time intervals as the batch run progresses. In order to obtain high quality products consistently, it is necessary that the process operating variables precisely follow their specified trajectories. However, often the malfunctions such as deviations in the specified trajectories, errors in charging the reactor with materials, and variations in impurities, lead to batch-to-batch variations. These affect the product quality adversely. Thus it becomes critical to detect and diagnose process faults and monitor the process continuously, since the faults can lead to reduced conversion, higher operating cost and sometimes even catastrophic failures and accidents. Accordingly, developing methodologies for the timely detection and diagnosing of faults has been an active area of research in recent years. The present study illustrates a Self-Organizing Map (SOM) based method for identifying process faults when the input conditions to a batch process are faulty. The proposed method can be easily extended to other types of above-stated malfunctions. The SOM is ANN-based nonlinear pattern recognition (classification), dimensionality reduction, and data projection and visualization formalism and it is described in detail in Chapter 2 (Section 2.3.2). The conventional methods for classification include, for example, K-means clustering [MacQueen, 1967] and fuzzy C-means clustering [Dunn, 1973] which cluster a multivariable data into clusters, without losing significantly the information content in the data. A major drawback of these conventional methods is that they are supervised clustering techniques and thus forcefully classify the data in a pre-specified number of clusters. Thus, when the

---

* Badhe Y. P., V. Wadekar, K. M. Desai, S. S. Tambe and B. D. Kulkarni, Poster Presented on NCL Day at NCL Pune, Feb 28, 2004.

number of classes in the data is unknown, these methods are not effective. In such situations SOM can be seen as a very useful classification method.

4.2.1 Case Study-I: Fed-Batch Fermenter for Protein Synthesis

This case study considers the fed-batch fermenter system for protein synthesis. The input data vectors for the SOM based classification consisted of three batch-process variables namely; (i) culture cell density (x_1), (ii) substrate concentration (x_2), and (iii) hold-up volume (x_3) [Lim et al., 1977]. The aim of this case study is to identify whether a batch has developed a fault at an early stage of the batch run. Here, we consider abnormal variations (faults) in the input variables (at the start of the batch) as a source of the fault in the batch fermentation process. It is assumed that at any given time, only one input variable is outside its normal range of its operation (termed *single fault*).

A. Simulation of the phenomenological model

To generate the process data for the fed-batch fermenter system for protein synthesis, the phenomenological model proposed in Kulkarni et al. [2003] has been used (see Eqs. 4.1 to 4.8). Here, the model is used only to simulate the process and generation of the process data under normal and faulty process operation. The historic experimental data can also be used for the SOM-based fault detection and diagnosis. The data set for a total of 39 batches, comprising values of the three predictor variables (x_1, x_2 and x_3) measured at one hour time intervals, and values of two output variables (y_1, y_2) measured at the end (15^{th} hr) of the batch, was generated by solving the set of five ordinary differential equations (Eqs. 4.1 to 4.8) given in section (4.2.1B) below. This set consisted data from 27 normal batches with the initial conditions in the "normal" ranges, ($0.95 < x_1 < 1.5$, $4.5 < x_2 < 5.5$, and $0.95 < x_3 < 1.5$) and 12 faulty batches (numbered 28 to 39) with initial conditions outside the stated ranges of a normal operation. The faulty batches are those wherein the initial conditions of an operating variable has deviated by 5% and 10% from its above-stated normal range. The real-life process data always contain some measurement noise and therefore 3% Gaussian noise was added to all the elements of the simulated data set. This dataset can be viewed

as a three-way array of size 39 (number of batches) × 3 (process variables) × 16 (time intervals). Next, the data set was partitioned on time basis, i.e. the data at the same time in different batches were taken together and they were subjected to the SOM based classification to find out how early faulty batches can be identified from the normal ones.

B. Phenomenological model for protein synthesis

$$\frac{dy_1}{dt} = g_1(y_2 - y_1) - \frac{u}{x_3} y_1 \tag{4.1}$$

$$\frac{dy_2}{dt} = g_2 x_1 - \frac{u}{x_3} y_2 \tag{4.2}$$

$$\frac{dx_1}{dt} = g_3 x_1 - \frac{u}{x_3} x_1 \tag{4.3}$$

$$\frac{dx_2}{dt} = -7.3 g_3 x_1 - \frac{u}{x_3}(20 - x_2) \tag{4.4}$$

$$\frac{dy_2}{dt} = u \tag{4.5}$$

$$g_1 = \frac{4.75 g_3}{0.12 + g_3} \tag{4.6}$$

$$g_2 = \frac{x_2}{0.1 + x_2} \exp(-0.5 x_2) \tag{4.7}$$

$$g_3 = \frac{21.87 x_2}{(x_2 + 0.4)(x_2 + 62.5)} \tag{4.8}$$

where t, x_1, x_2 and x_3 refer to time (min), culture cell density ($g\ l^{-1}$), substrate concentration ($g\ l^{-1}$), and hold up volume (l), respectively; y_1 and y_2 are the concentrations ($g\ l^{-1}$) of the secreted protein and the total protein, respectively, and u refers to the nutrient (glucose) feed rate ($l\ h^{-1}$). The fed-batch culture is fed at an exponentially increasing nutrient feed rate, $u = u_0 e^{0.219t}$ [Abonyi et al., 2003], with the constraint, $0 \le u \le 2.5$, where u_0 is a constant (= 0.0926 $l\ h^{-1}$).

C. Results and Discussion

The optimum grid size of the 2-D SOM was selected by running the SOM algorithm using different grid sizes. The optimum grid size obtained was [15 × 15] neurons and the algorithm was run for 2000 training epochs (1000 in rough training phase and 1000 in fine training phase) using the SOM Tool box [Vasanto et al., 2000].

The results of classification are portrayed on two dimensional SOM grids. These can be interpreted using the U-matrix plots displaying the relative distance between the points using a gray scale, wherein the whiter space between any two points indicates that the points are closer than the points having less white space between them (see Figure 4.1 – Figure 4.3). The actual distance between the points can be obtained from the scale bar shown in the right hand side of the figures. In Figure 4.1 to Figure 4.3 the unit lables indicate the fault index (i.e., which of the three variables is behaving abnormally). The best matching units without any index indicate normal batches. From the SOM grid plots obtained at different time intervals (3rd hr, 7th hr, and 12th hr)(see Figure 4.1 – Figure 4.3), it is observed that the best matching units corresponding to faulty batches shown on the plot move away from the normal batches. These essentially attain the character of outliers. The faulty batches in which (x_1) or (x_3) is outside the normal operating range move away from the points representing the normal batches as compared to the batches for which (x_2) is in abnormal operation range. From this observation we can interpret that the system is tolerant in case of deviations in the initial substrate concentration (x_2) with 5% and 10% variations from the normal operating range. Applying SOM, the faulty batches are identified at the 3rd hr itself, which is an early stage of the batch run and identification of a faulty batch at such an early stage is very useful to avoid further adverse effects.

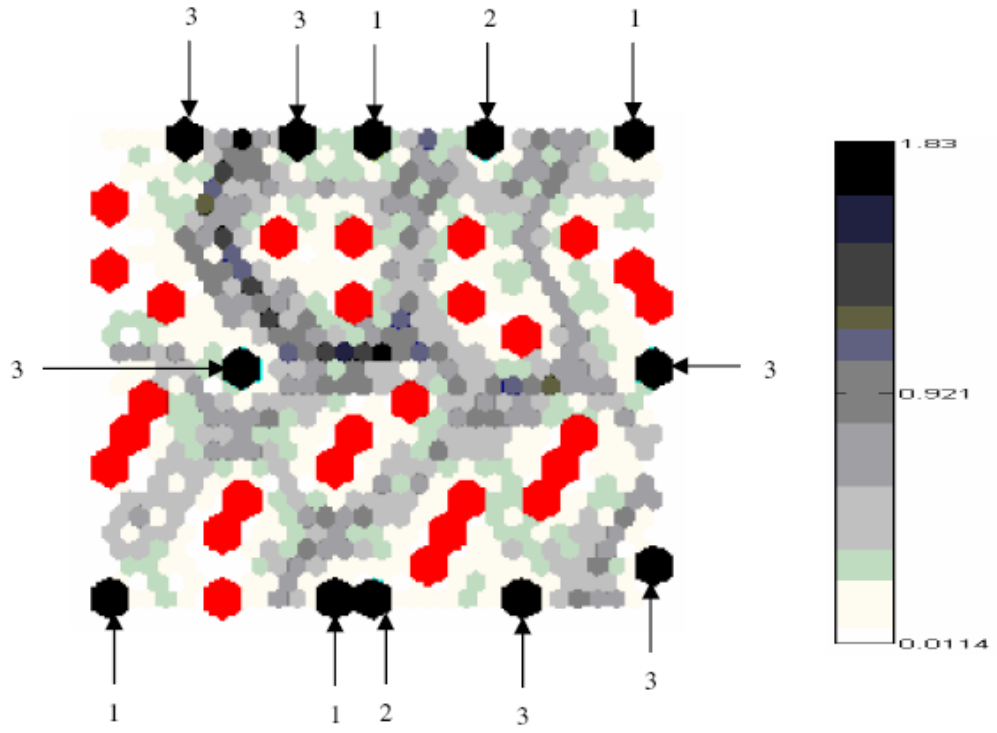

Figure 4.1: U-matrix visualization of self-organizing maps at 3rd hr of the process operation showing the faulty as well as normal batch

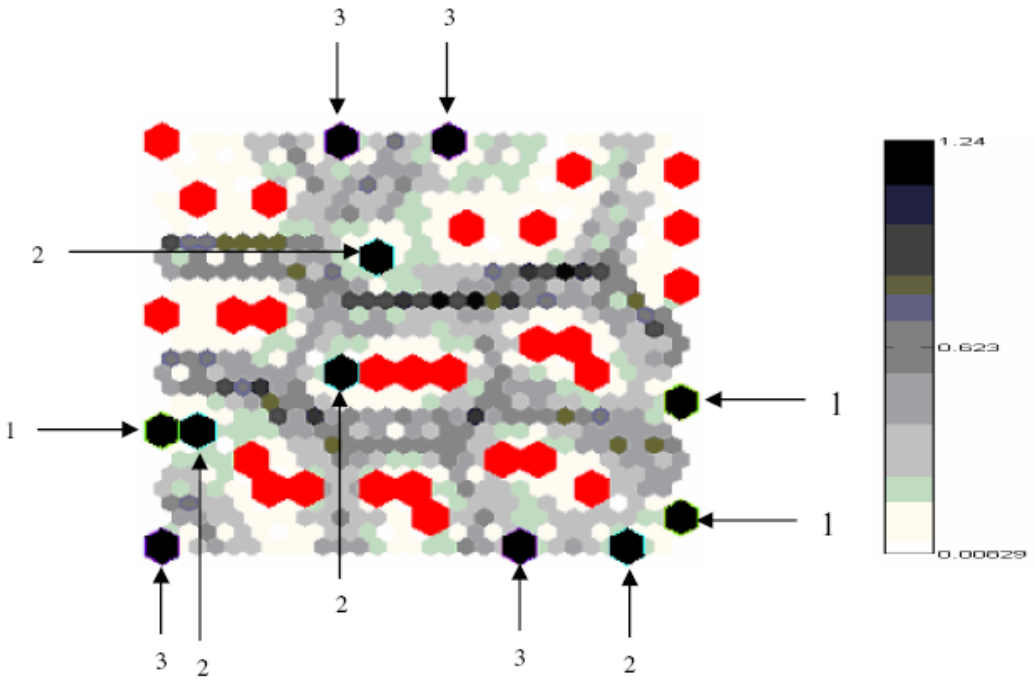

Figure 4.2: U-matrix visualization of SOM at 7th hr of the process operation showing the faulty as well as normal batches

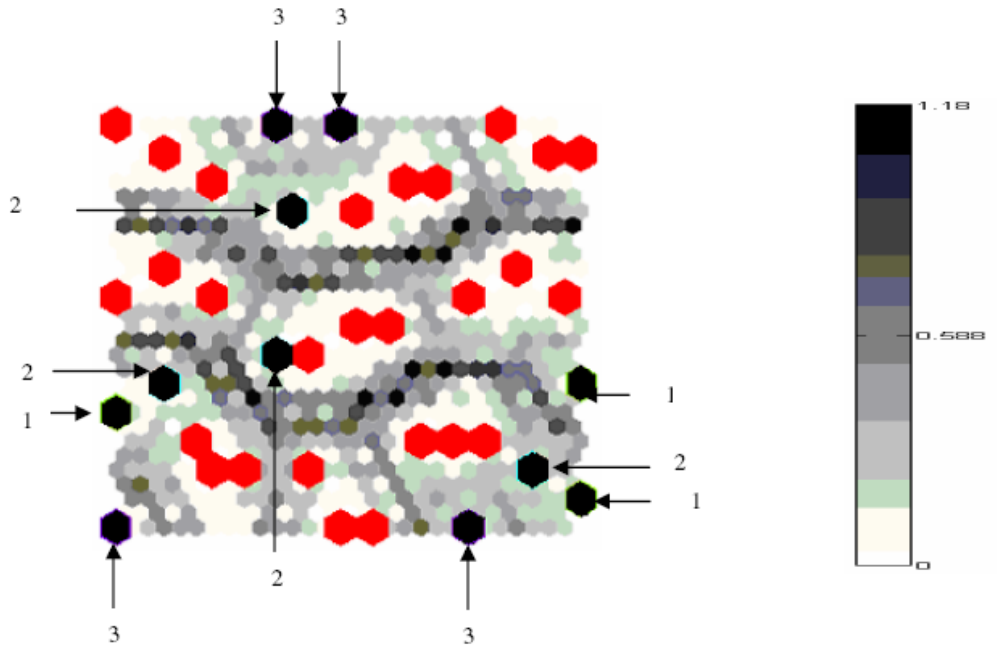

Figure 4.3: U-matrix visualization of self-organizing maps at 12th hr of process operation showing the faulty as well as normal batches

4.2.2 Case Study-II: Fault Detection/Diagnosis of Batch Fermentation Process of Citric Acid Production

Despite advanced computer based control systems, process variables (reactant concentrations, temperature, pressure, etc.) do fluctuate owing to equipment and sensor malfunctions. The faults arising from such fluctuations can lead to reduced conversion and selectivity higher operating cost and sometimes even catastrophic accidents. Thus it becomes important to detect and diagnose process faults in a timely manner so that corrective measures can be taken swiftly. Detection and diagnosing of fault requires continuous process monitoring.

When the number of process variables to be monitored is large, it poses significant difficulties in using the conventional "trends and scatter" plot method of monitoring. For overcoming these difficulties, linear and nonlinear techniques, which reduce the dimensionality of the process data and thereby allow monitoring of fewer variables, are often used. In this study, SOM has been illustrated for the purpose of monitoring and fault detection in fermentation processes. For illustrating SOM-based monitoring and fault detection/diagnosis, a batch fermentation process involving biomass growth and citric acid production by Aspergillus niger [Bizukojc et al., 2003] is considered. The progress of the

fermentation process as it transverses through a number of stages, can be effectively visualized with the help of SOM on a two-dimensional map.

A. Batch process of biomass growth and citric acid production

A morphologically structured model for the batch process of biomass growth and citric acid accumulation by Aspergillus niger [Bizukojc et al., 2003] is used in this work. The model consists of ten ordinary differential equations, which balance biomass and four physiological zones, and includes the most important medium components, such as carbon sources, nitrogen source and citric acid. The phenomenological model equations are as follows.

$$\frac{dSUC}{dt} = -r_H \tag{4.9}$$

$$\frac{dFRU}{dt} = \upsilon_S r_H - r_{FRU} X \tag{4.10}$$

$$\frac{dGLU}{dt} = \upsilon_S r_H - r_{GLU} X \tag{4.11}$$

$$\frac{dAMON}{dt} = -r_{AMON} X \tag{4.12}$$

$$\frac{dCIT}{dt} = r_{CIT} X \tag{4.13}$$

$$\frac{dZ_A}{dt} = -u_1 - u_2 + Z_A(\mu_A - \mu) \tag{4.14}$$

$$\frac{dZ_B}{dt} = u_1 - u_3 - Z_B \mu \tag{4.15}$$

$$\frac{dZ_C}{dt} = u_2 + u_3 - u_4 - Z_C \mu \tag{4.16}$$

$$\frac{dZ_D}{dt} = u_4 - Z_D \mu \tag{4.17}$$

$$\frac{dX}{dt} = \mu X \tag{4.18}$$

where,

AMON: concentration of ammonium ions (g l^{-1})

B: sucrose hydrolysis rate parameter at linear term (h^{-1})

CIT: concentration of citric acid (g l^{-1})

FRU: concentration of fructose (g l^{-1})

GLU: concentration of glucose (g l^{-1})

$r_{CIT/F}$: citric acid production rate from fructose (g *CIT* l^{-1} h^{-1})

$r_{CIT/G}$: citric acid production rate from glucose (g *CIT* l^{-1} h^{-1})

r_{FRU}: fructose utilization rate (g *FRU* l^{-1} h^{-1})

r_{GLU}: glucose utilization rate (g *GLU* l^{-1} h^{-1})

r_H: rate of sucrose hydrolysis (g *SUC* l^{-1} h^{-1})

SUC: concentration of sucrose (g l^{-1})

t: time (hr)

u_1: metamorphosis reaction zone A into B rate (g A g X^{-1} h^{-1})

u_2: metamorphosis reaction zone A into C rate (g A g X^{-1} h^{-1})

u_3: metamorphosis reaction zone B into C rate (g B g X^{-1} h^{-1})

u_4: metamorphosis reaction zone C into D rate (g C g X^{-1} h^{-1})

X: concentration of biomass (g l^{-1})

Z_A: zone A fraction (g A g X^{-1})

Z_B: zone B fraction (g B g X^{-1})

Z_C: zone C fraction (g C g X^{-1})

Z_D: zone D fraction (g D g X^{-1})

μ_a: specific growth rate for zone A (h^{-1})

v_S: stoichiometric coefficient for hydrolysis of sucrose (g *GLU* g SUC^{-1}) or (g *FRU* g SUC^{-1})

To generate process data pertaining to normal and faulty batches, the above-stated 10 ordinary differential equations (Eqs 4.9 to 4.18) were solved by the standard *Gear's* algorithm. In total, 36 batches were simulated. Batches 1 to 32 are the normal batches wherein the concentration of citric acid at the end (170 hrs) of the batch is more than 29 g/l. Batches 33 to 34 are abnormal batches due to low initial concentration of NH_4^+ ions. Batches 35 to 36 are also abnormal owing to low concentration of the biomass. Table 4.1 represents the initial concentrations of the variables/parameters used in the simulation of all 36 batches along with the labels used to represent these batches on the result plots from SOM.

Table 4.1: Initial concentrations and labels for all batches

No.	**Parameters**	**Batch No. 1 to 11**	**Batch No. 12 to 22**	**Batch No. 23 to 32**	**Batch No. 33 & 34**	**Batch No. 35 & 36**
1	Sucrose conc. g/l	85-93.95	94.189	94.189	94.189	94.189
2	Glucose conc. g/l	0	0	0	0	0
3	Fructose conc. g/l	0	0	0	0	0
4	Ammonium ions conc. g/l	0.534	0.50-0.56	0.534	0.4-0.45	0.534
5	Citric acid conc. g/l	0	0	0	0	0
6	Biomass conc. g/l	0.024	0.024	0.023-0.025	0.024	0.008-0.01
7	Number of batches	11	11	10	2	2
8	Labels	1S to 11S	1A to 11A	1xx to 10xx	aA and aAA	ax and axx

B. Application to classification of fermentation process into trophophase and idiophase phase

To identify two different phases in the reaction, SOM was applied and optimized to get the results presented in the form of the U-matrix and the component planes in Figure 4.4. The U-matrix denotes the distances between the nodes of the SOM grid. The color of hexagon between nodes indicates the

distance of the nodes from its neighboring nodes [Rantanen et al., 2001]. Dark red color represents the area where the elastic SOM nodes are stretched between the two data clusters; the dark blue color represents the closest distance. From U-matrix in the Figure 4.4, it can be seen that six clusters are formed: the top two clusters on the left and right sides correspond to Trophophase and remaining four clusters to Idiophase. The component planes, Figure 4.4, depict how the SOM visualizes the data (inputs) space. Let us first look at the top-left corner cluster in Figure 4.4; it has 94.1 g l^{-1} concentration level (see panel "suc"), a zero value for glucose concentration (see panel "glu") and fructose (see panel "fru"), 0.534 g l^{-1} of concentration for ammonium ions (see panel "amon"), zero citric acid concentration (see panel "cit") and 0.0638 g l^{-1} concentration for biomass. Component planes or sections for the four zones (z_a, z_b, z_c and z_d) represent the stretch of each zone individually.

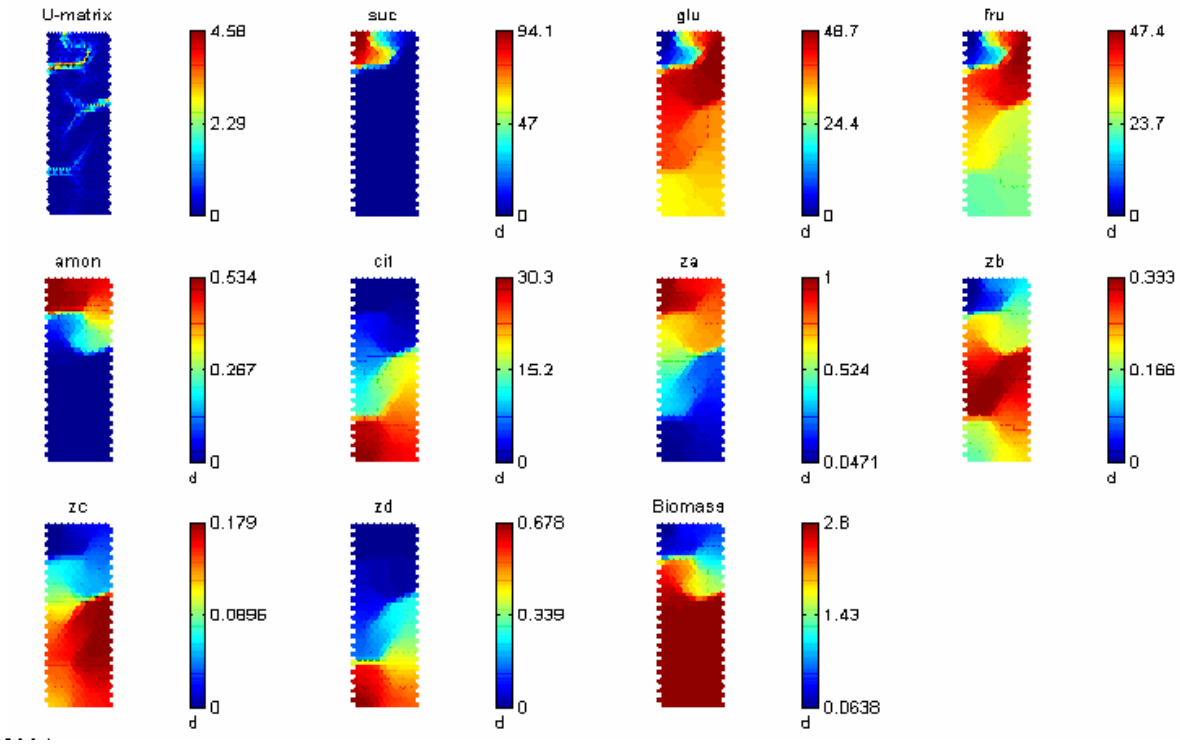

Figure 4.4: Visualization of SOM for citric acid production for a batch with initial conditions given in **Table 4.1** Batch No. 17

C. Application to fault detection and diagnosis (by cluster method)

The simulated data for all 36 batches can be used for classification of the normal and abnormal batches. Note that the total batch time is 170 hrs and thus the SOM was applied at three times (30th hr, 100th hr and the last at 170th hr). The resultant U-matrix and their labels are shown in Figures (Figure 4.5 – Figure 4.7). Labels listed in Table 4.1 can be used in the figures (Figure 4.5 – Figure 4.7) to identify the batches. In Figure 4.5, abnormal batches (denoted by aA, aAA, ax and axx on the left side) are not well separated from some of the normal batches as this is a very early phase in the reaction. In other figures (Figure 4.6 and Figure 4.7), the abnormal batches are well separated from the normal batches (two batches on extreme left and another two on the extreme right). Thus, SOM has successfully separated the normal and abnormal batches after half the batch time is completed.

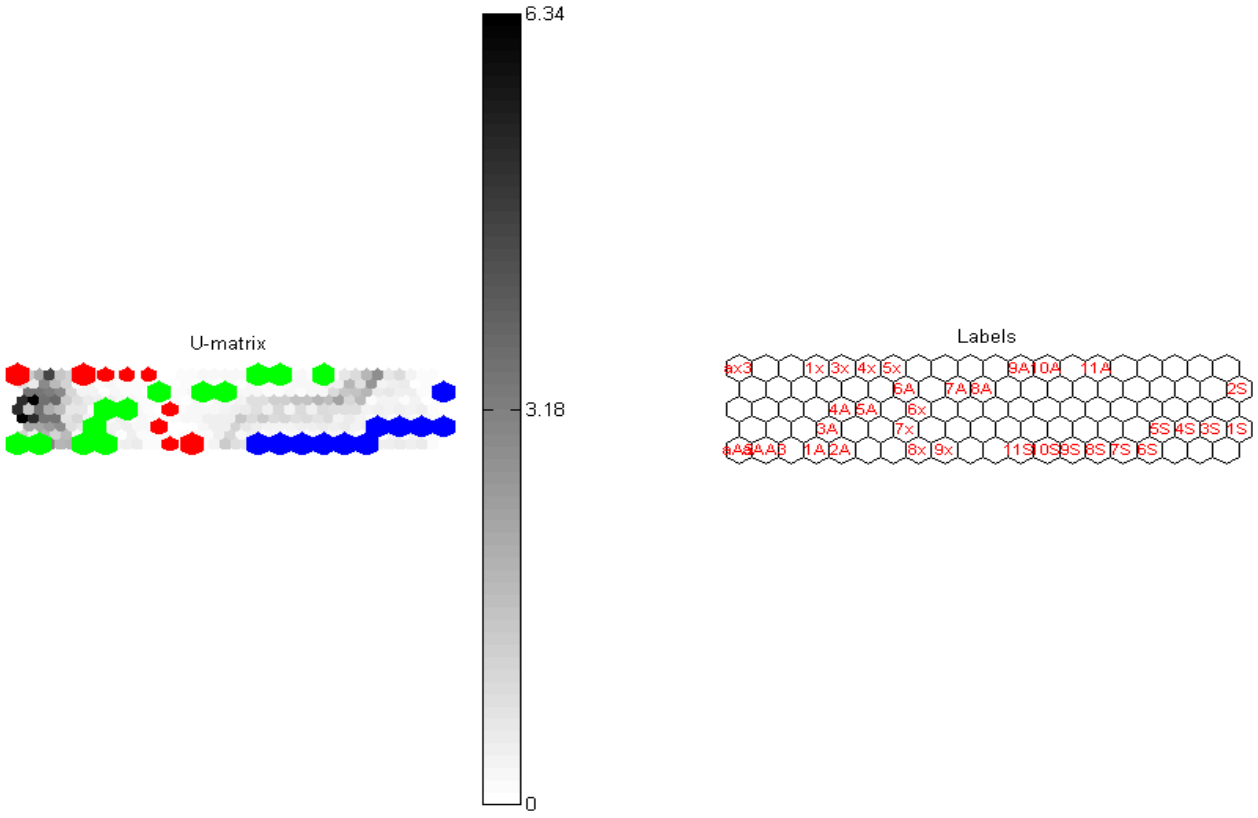

Figure 4.5: U-matrix with the distribution of the batches, along with labels(at 30th Hrs)

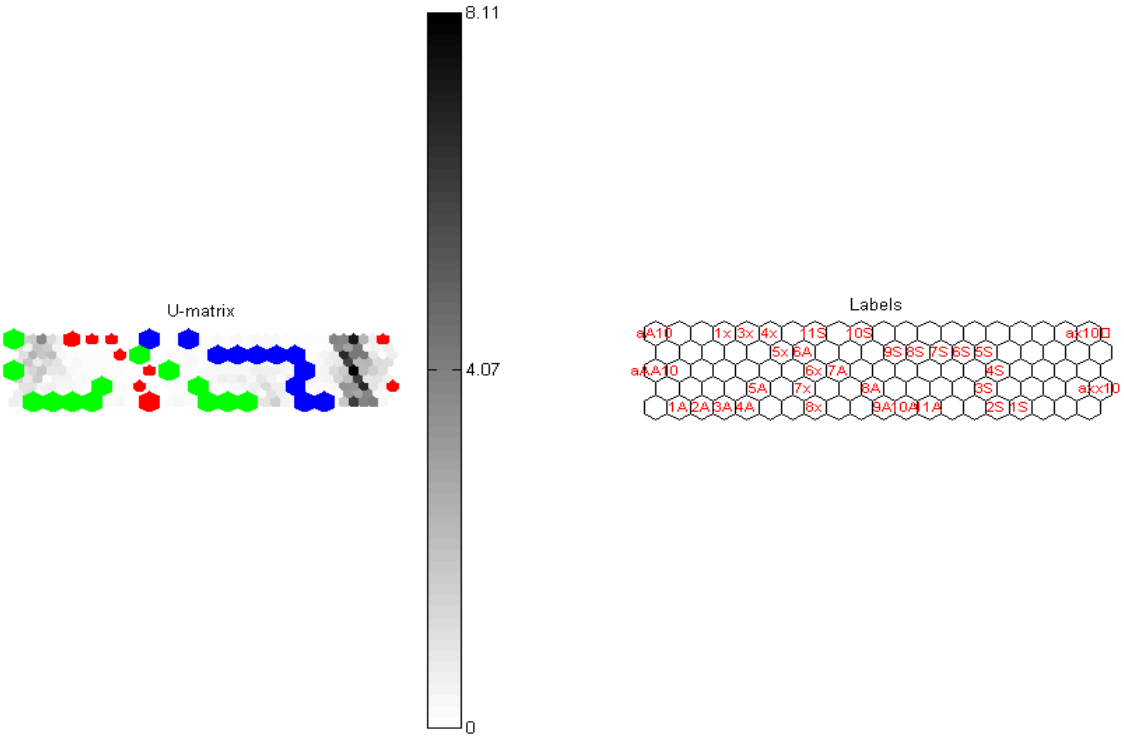

Figure 4.6: U-matrix with the distribution of the batches, along with labels (at 100 Hrs)

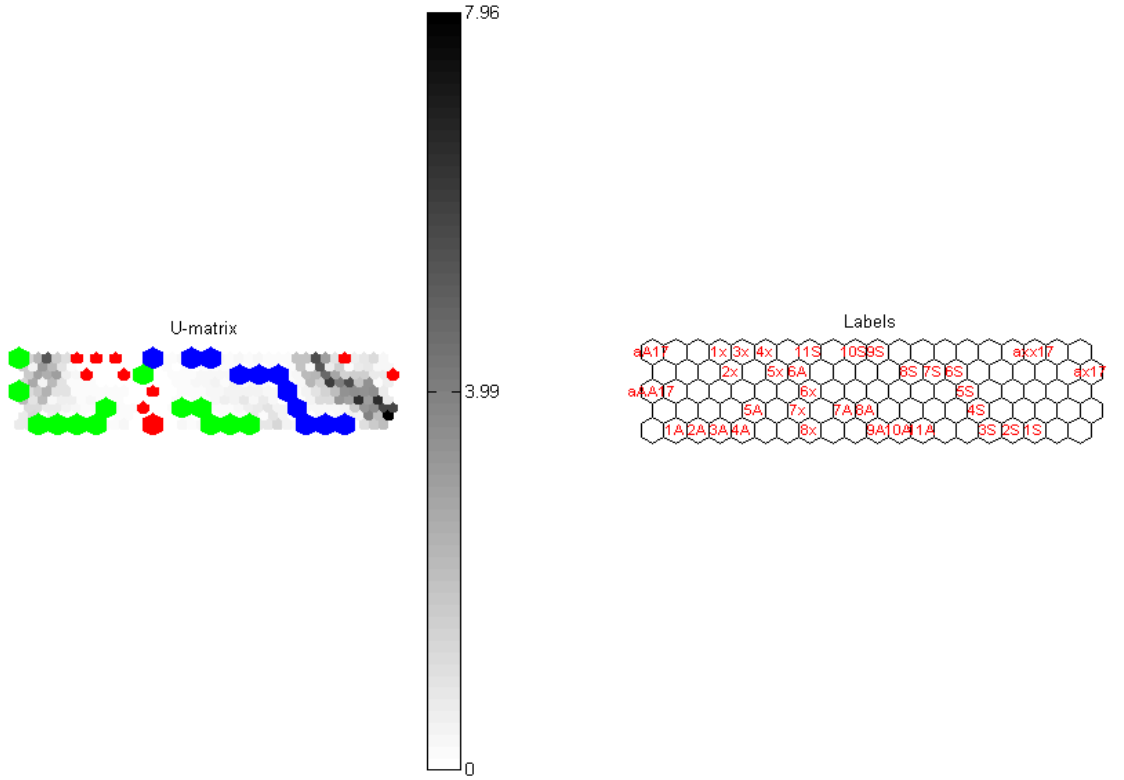

Figure 4.7: U-matrix with the distribution of the batches, along with labels (at 170 Hrs)

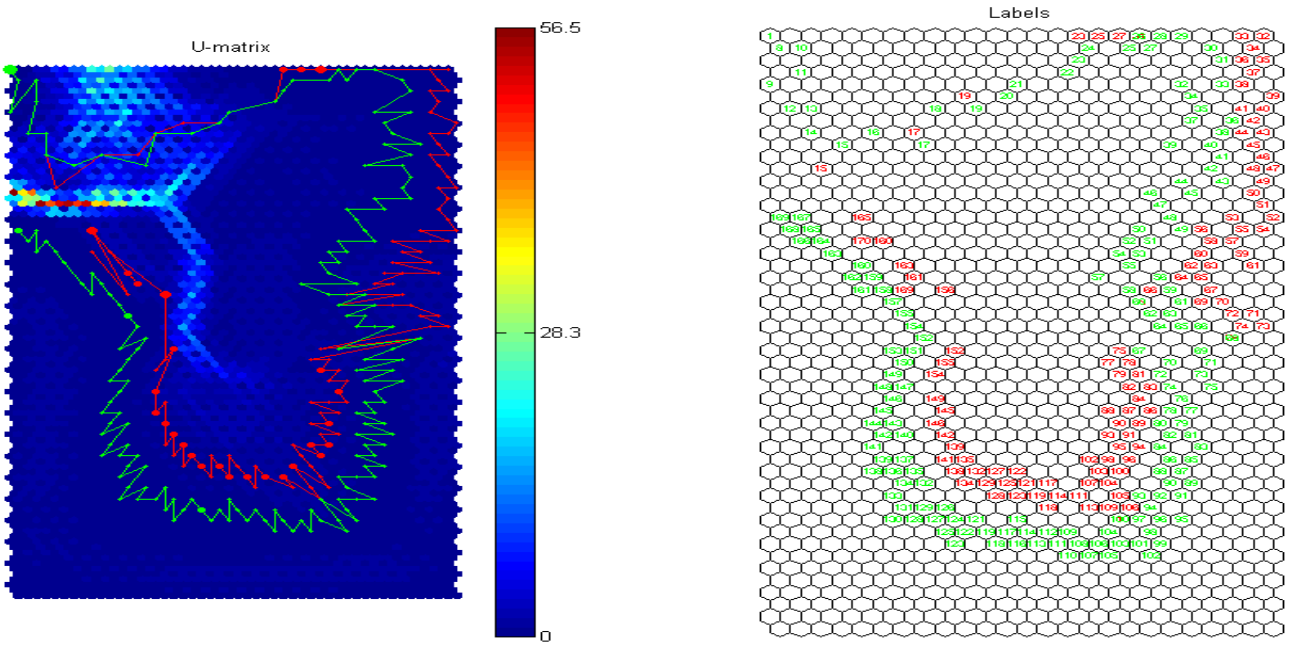

Figure 4.8: Trajectories of the normal (green) and abnormal (red) batches on the top of U-matrix

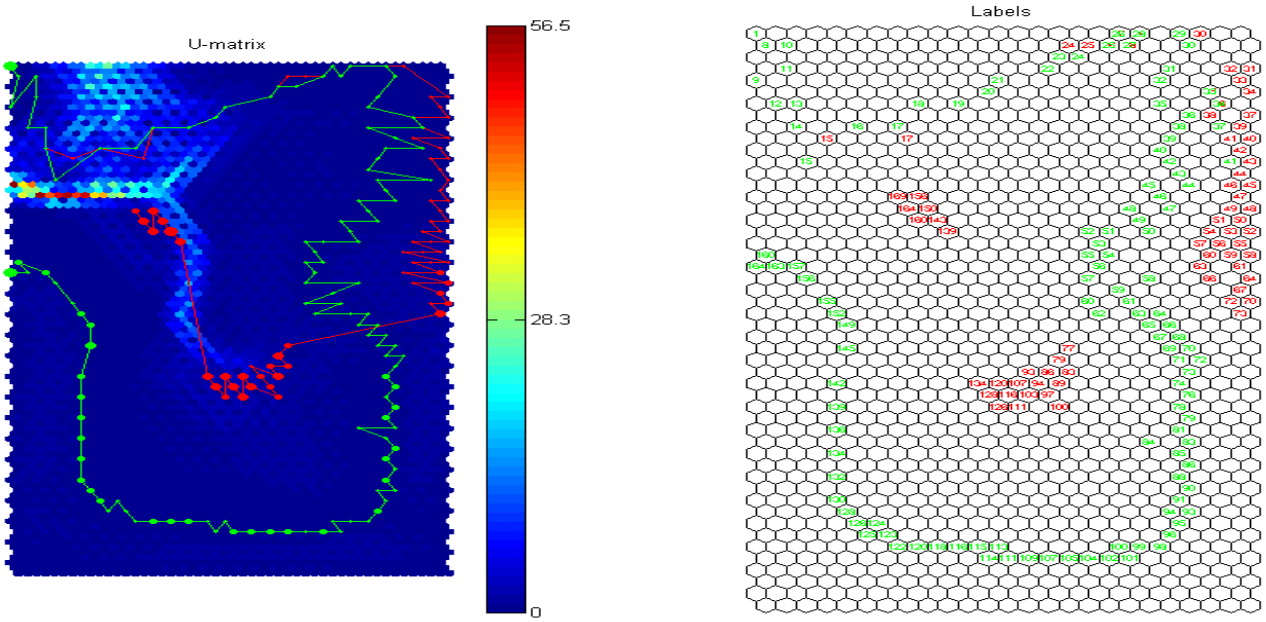

Figure 4.9: Trajectories of the normal (green) and abnormal (red) batches on the top of U-matrix

D. Application to fault detection and diagnosis (by trajectory method)

The SOM can be used to form a 2-dimensional display of the operational states of the process. The current process state and its history in time can be visualized as a trajectory on the map. This allows efficient tracking of the process

dynamics. The SOM facilitates understanding of process dynamics so that several variables and their interactions may be inspected simultaneously [Simula et al., 1999].

In the present study, the training data from 36 different batches were taken with initial values of the process variables described in Table 4.1 (duration of each batch is 170 hrs monitors at 1 hr time interval) to form a matrix. Each point is a vector comprising values of process variables measured for one batch at a particular time. The process starts from the circle at the top left corner of the map, proceeds through the map by an individual route, and ends at the other circle. The trajectory path can be easily traced by the labels. The red trajectories in Figure 4.8 and Figure 4.9 represent low production of citric acid (abnormal batch) due to low concentration of biomass and ammonium ions, respectively and the green trajectories represent the normal batches. In this manner, the progress of a batch can be monitored and its normal/abnormal behavior can be identified.

4.2.3 Conclusion

In the first case study described in this chapter, i.e., protein synthesis, the unsteady-state operating variables data from a fed-batch fermenter operating under faulty as well as normal operating conditions was explored. The SOM could reduce the dimensionality of the data from the 3 -D to 2-D with an excellent precision. Also, the reduced dimensioned data formed good clusters clearly separating the normal and faulty batches. To summarize, the results of this study suggest that the SOM is an attractive strategy for unsupervised classification analysis of multidimensional process data. The advantage of the SOM for classification analysis is that it allows process engineers and operators a convenient single-window view of the behavior of the batch. This feature is very advantageous in process monitoring, control and fault detection and diagnosis. It is also possible to use the dimensionality reduced data for modeling purposes thereby substantially lowering the numerical load.

In the second case study involving citric acid production, the SOM visualizes the dynamics of a multivariable process in the form of a two-dimensional map and brings out subtle differences between various batches. It has

been shown that efficient process monitoring can be performed from the two-dimensional projection of the process variables. It can thus be seen that SOM is a novel tool to monitor and detect faults in a complex batch fermentation process. It is very effective in the detection of typical operating regions related to an optimal process operation which can also be used to predict whether the batch will end normally or abnormally on the basis of the current values of the process variables. The proposed method is attractive in comparison with other process monitoring schemes, such as linear/nonlinear principle component analysis.

4.3 REFERENCES

1. Abonyi, J., S. Nemeth, C. Vincze and P. Arva, (2003), Process Analysis And Product Quality Estimation By Self Organizing Maps With An Application To Polyethylene Production, *Computers in Industry*, 52(3), Pages 221–234
2. Bizukojc, M. and S. Ledakowicz (2003), Morphologically structured model for growth and citric acid accumulation by Aspergillus niger, Enz. Microbial Technol. Vol-32, 268–281.
3. Dunn, J. C. (1973), A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters, *Journal of Cybernetics* 3, 32–57.
4. Kohonen, T. (1990), The self-organizing map. Proceedings of the IEEE 78, 1464–1480.
5. Kulkarni, S. G., A. Chaudhary, S. Nandi, S. S. Tambe and B. D. Kulkarni, (2003), (2004), Modelling And Monitoring Of Batch Processes Using Principle Component Analysis Assisted Generalized Regression Neural Networks (GRNN), *Biochemical Engineering Journal* 18(3), 193–210.
6. Lim, H. C., B. J. Chen and C. C. Creagan (1977), An Analysis Of Extended And Exponentially-Fed-Batch Cultures, *Biotechnol. Bioeng*. 14, 425– 433.
7. MacQueen, J. B. (1967), Some Methods for classification and Analysis of Multivariate Observations, *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, University of California Press, 1, 281–297.
8. Rantanen, J. T., S. J. Laine, O. K. Antikainen, J. Mannermaa, O. E. Simula, and J. K. Yliruusi (2001), Visualization of fluid-bed granulation with self-organizing maps, *J. Pharm. Biomed. Anal.*, 24, 343–352.
9. Simula, O., Vesanto, J., Alhoniemi, E., and Hollmn, J. (1999c). Analysis and modeling of complex systems using the self-organizing map. In Kasabov, N. and Kozma, R., editors, *Neuro-Fuzzy Techniques for Intelligent Information Systems.*, Physica-Verlag, 3–22.
10. Vasanto, J., J. Himberg, E. Alhoniemi, K. Kiviluoto and J. Parhankangas (1999), Self-Organizing Map in Matlab: the SOM Toolbox, *Proceedings of the Matlab DSP Conference*, Espoo, Finland, 35–40.
    http://lib.tkk.fi/Diss/2002/isbn951226093X/article2.pdf.

CHAPTER 5

PROCESS OPTIMIZATION

5.1 PROCESS OPTIMIZATION USING MEMETIC ALGORITHMS: A CASE STUDY OF BENZENE HYDROXYLATION TO PHENOL PROCESS*

5.1.1 Introduction

In recent years, Genetic algorithms (see Section 2.4.3), which are population-based stochastic optimization methodologies possessing a number of attractive properties, have been widely explored in chemical process optimization applications. The principle drawback of GA however, is that since the formalism performs a global search of the solution space, it can take a long time to converge even if the optimal solution lies in the neighborhood of a candidate solution. A number of GA-hybrids have been proposed to improve the efficiency of GAs and their speed of convergence. A genetic algorithm related methodology that overcomes the stated problem is known as "Memetic algorithms (MA)". The most attractive feature of MA (see Section 2.4.4), which makes it a powerful nonlinear search and optimization formalism, is that it combines a local search heuristics with the population-based global search conducted by the GA. This feature of MA greatly helps in speeding up the convergence to an optimal solution for an objective function maximization/minimization problem. The said approach has proved successful in a variety of problem domains and in particular for solving NP Optimization problems (Kollen et al., 1994; Digalakis et al., 2003). Accordingly, this section presents a study of a MA-based optimization of the zeolite (TS-1) catalyzed benzene hydroxylation to phenol process. Additionally, the optimal solutions obtained using the MA formalism are compared with the GA-based solutions obtained in an earlier study (Nandi et al., 2002). The results of

---

* Badhe Y. P., K. M. Desai, V. Wadekar, U. Phulwale, S. S. Tambe and B. D. Kulkarni, Presented in *Indian Chemical Engineering Congress* (*CHEMCON-2004*), organised by Indian Institute of Chemical Engineers, held at The Grand Hyatt, Mumbai, during 28-30 Dec. 2004.

this comparison indicate that the MA has searched a better solution and that too in a shorter time when compared to the GA-based solution.

In this work, an artificial neural network (ANN) based process model is developed first from the steady-state process data [Nandi et al., 2002]. The input space of this ANN model representing process operating variables is then optimized using the MA formalism with a view of simultaneously optimizing multiple process output variables and thereby improving the process performance. The results of the MA-based optimization have been compared with those obtained in an earlier study by Nandi et al., [2002] using the GA formalism.

5.1.2 Modeling and Optimization of Benzene Hydroxylation Reaction

A significant success of the TS-1 based triphasic system was achieved for the benzene hydroxylation to phenol reaction wherein a 15 to 20 fold increase (as compared to a biphasic system) in the benzene conversion was realized; also, selectivity for the desired product (phenol) was significantly higher (Bhaumik et al., 1998). The details of the TS-1 catalyzed benzene hydroxylation to phenol reaction are described in Nandi et al. [2002]. In this reaction, apart from phenol, which is the desired reaction product, small amounts of secondary products such as hydroquinone, catechol and *para*-benzoquinone, are also formed. Phenol being a widely consumed industrial chemical, it is economically sensible to optimize the TS-1 catalyzed benzene to phenol process. For achieving this objective, the ANN-GA hybrid modeling and optimization formalism was used by Nandi and co-workers (Nandi et al., 2002). The specific goal of this process optimization study is to develop an ANN-based process model from the reaction data, and to perform an MA based optimization of the ANN model with a view to obtain the optimal process conditions effecting: (i) higher phenol selectivity, (ii) enhanced H_2O_2 utilization, and (iii) increased benzene conversion.

5.1.3 Development and Optimization of ANN-based Process Model

A total of 24 experiments were conducted for studying the effects of five reaction (model) input variables namely, reaction time (x_1), catalyst weight percentage (x_2), reaction temperature (x_3), benzene to peroxide mole ratio (x_4) and,

water to benzene weight ratio (x_5). The catalyst performance was monitored in terms of three reaction (model) output variables namely, phenol selectivity (y_1) (mole %), peroxide utilization (y_2) (mole %), and benzene conversion (y_3) (mole %). This data set was used to develop a feed-forward ANN model; the details of the process data and development of the ANN based process model thereby are given by Nandi et al. [2002].

The multi-objective (MO) optimization problem involving simultaneous maximization of the three reaction output variables is converted into a single objective (SO) optimization problem by defining:

$$\text{Maximize } \hat{f}\left(\mathbf{x}, \mathbf{W}\right) = w_1 y_1 + w_2 y_2 + w_3 y_3; \quad \rightarrow \quad x_n^L \le x_n \le x_n^U \tag{5.1}$$

where $\hat{f}$ represents the aggregated objective function; **W** represents the ANN model's weight matrix; the N-dimensional decision vector, **x**, denotes the reactor operating conditions ($N = 5$); w_k ($k = 1$ to 3) refers to the kth weighting coefficient; x_n^L and x_n^U respectively represent the lower and upper bounds on x_n; and y_1, y_2, and y_3 respectively represent the three reaction output variables (to be maximized).

The five-dimensional input space of the ANN-based process model was optimized by the MA method; the values of MA-specific parameters used in the optimization simulations were: chromosome length (l_{chr}) = 50, population size (N_{pop}) = 30, crossover probability (p_{cr}) = 0.95, mutation probability (p_{mut}) = 0.01, and number of generations (N_g^{max}) = 250. The Tabu-based local search (see Section 2.4.2) was implemented using following values of Tabu parameters: number of neighbors (N_{neigh}) = 20, number of iterations (N_{iter}) = 15, length of the Tabu list (TL) = 100, shape coefficient of sigmoid function (σ) = 0.5 and intensification coefficient (β) = 0.5. For computing the fitness value (ξ_j) of the jth candidate solution ($\mathbf{x}_j$) in a population, following fitness function was employed:

$$\xi_j = w_1\left(\frac{y_{j1}}{100}\right) + w_2\left(\frac{y_{j2}}{100}\right) + w_3\left(\frac{y_{j3}}{100}\right) \tag{5.2}$$

where

$$\sum_{k=1}^{3} w_k = 1.0\ ; \qquad j = 1,2,...,N_{pop} \tag{5.3}$$

The weighting coefficients w_1, w_2 and w_3 represent the relative importance of the three objectives, and y_{j1}, y_{j2} and y_{j3}, respectively represent the ANN model-predicted values of the three output variables when *j*th candidate solution is applied to the network's input nodes. Since maximizing percentages of the three output variables, namely, phenol selectivity (y_1), hydrogen peroxide utilization (y_2), and benzene conversion (y_3), are equally important objectives, the values of w_1, w_2 and w_3, were fixed at 0.333. During the implementation of the MA, the search for the optimal solution was restricted to the following ranges [x_n^L, x_n^U] of the five process operating variables: (i) reaction time (x_1): [0.25, 4.0], (ii) catalyst concentration (x_2): [5.0, 30.0], (iii) temperature (x_3): [323.0, 353.0], (iv) benzene to peroxide ratio (x_4): [1.0, 8.0], and (v) water to benzene ratio (x_5): [1.0, 10.0].

5.1.4 Results and Discussion

The three optimal solutions obtained using the MA formalism are listed in Table 5.1. For comparison purposes the table also lists three GA-based optimal solutions (as obtained by Nandi et al., 2002). It is seen from the tabulated values that the MA formalism has yielded overall best solution with the fitness score of 88.63. This solution has improved the values of three process outputs namely, phenol selectivity, hydrogen peroxide utilization, and benzene conversion by 0.2 %, 0.7 % and 2.3 %, respectively when compared with the best GA solution. Figure 5.1 portrays the generation-wise evolution of the best solution obtained using the MA. For comparison, the evolution of the best GA-based solution is also shown in Figure 5.1. It is observed from the figure that the MA has taken lesser number of generations (iterations) to reach the overall optimal solution when compared to the GA. It may be noted that the inclusion of the local search requires more time to complete MA iteration. Specifically, it was observed that the MA and GA take approximately 7 and 4 milliseconds, respectively, for completing a single iteration (as measured on Windows XP workstation with 2.0 GHz P4 processor). Despite taking more time to complete an iteration, the MA approached the optimal solution faster (CPU time = 20*7 = 140 ms) when compared to the GA (time = 70*4 = 280 ms). These results clearly indicate that the local search feature of the MA has helped in reaching the optimal solution in half of the time

taken by the GA. Also, the MA has searched a better solution than the one obtained by the GA.

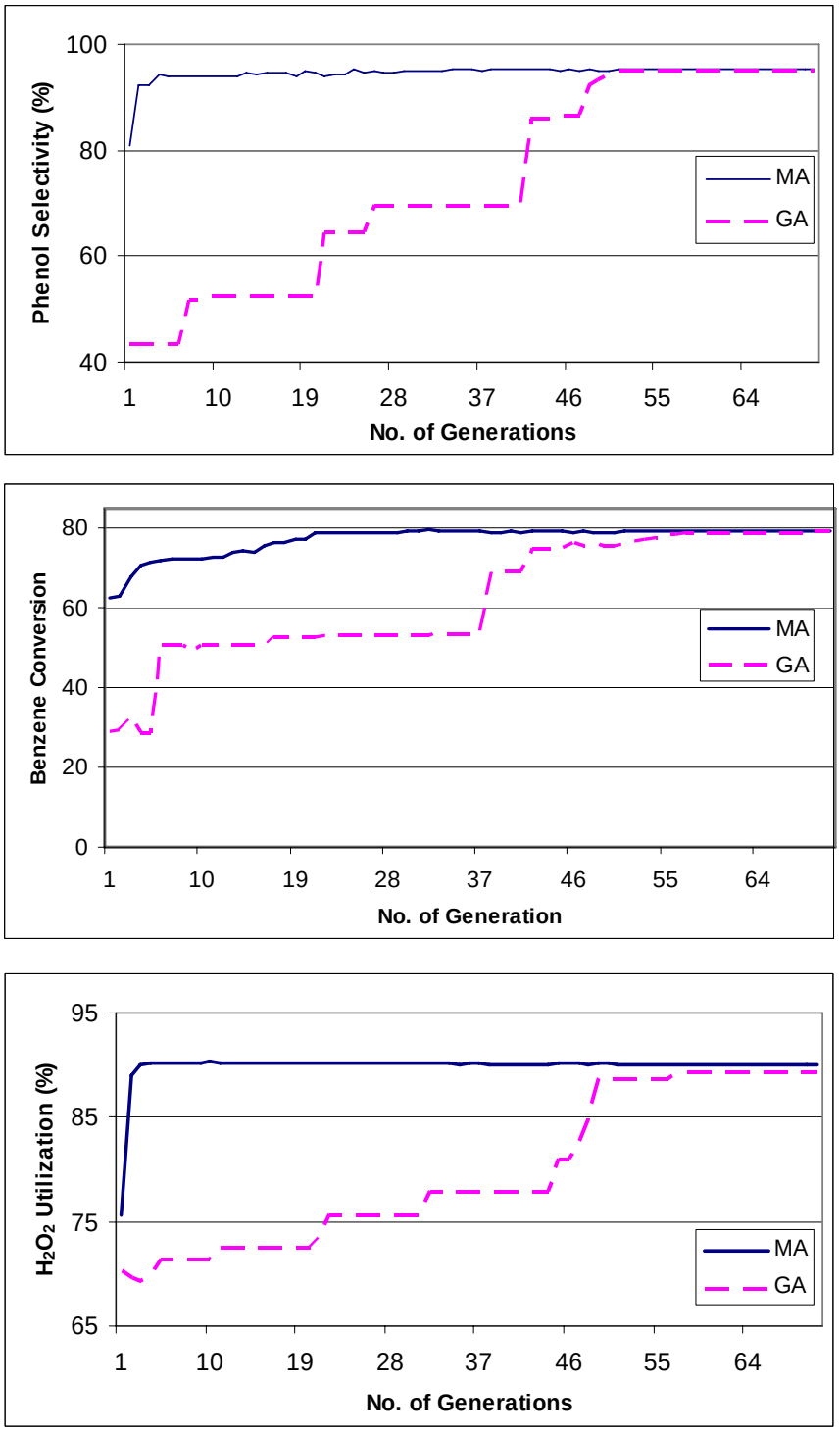

Figure 5.1: Plots showing the generation-wise evolution of the solutions given by MA and GA formalisms

Table 5.1: Optimized Operating Conditions Obtained Using MA and GA formalisms

Optimal solution		Optimized process input variables					Maximized output variables			Fitness Value (ξ)
		Time (hr) (x_1)	Catalyst Conc. (x_2)	Temp (K) (x_3)	B to P Ratio (mol/mol) (x_4)	W to B Ratio (wt/wt) (x_5)	Phenol Selectivity (%) (y_1^p)	H_2O_2 Utilization (%) (y_2^p)	Benzene Conversion (%) (y_3^p)	
1	MA	3.1	24.5	345.0	1.0	8.1	93.5	90.1	81.1	88.23
	GA	3.0	25.0	349.0	1.0	8.0	93.3	89.8	80.2	87.13
2	MA	2.8	23.4	346.1	1.0	8.2	93.1	90.7	82.1	88.63
	GA	2.8	22.5	345.5	1.0	8.6	93.3	90.0	79.8	87.7
3	MA	3.3	23.3	346.2	1.0	9.1	92.7	89.5	81.2	87.8
	GA	3.4	22.9	348.0	1.0	9.5	92.8	89.3	79.4	87.16

Note: B: Benzene, P: Hydrogen peroxide and W: Water

5.1.5 Conclusion

This study presents results of memetic algorithm based MO optimization of the zeolite (TS-1) catalyzed benzene hydroxylation to phenol process. Additionally, the optimal solutions obtained using MA are compared with the GA-based solutions obtained in an earlier study. The results of this comparison indicate that the MA has captured a better solution and that too in lesser time when compared to the GA. It can thus be concluded that the global search augmented with a local search performs better than a purely global search for securing an optimized solution.

5.2 REFERENCES

1. Bhaumik, A., P. Mukherjee, R. Kumar (1998), Triphase catalysis over titanium–silicate molecular sieves under solvent-free conditions: I. Direct hydroxylation of benzene, *J. Catal.*, 178 (1), 101–107.
2. Digalakis, J., K. Margaritis (2003), Performance comparison of memetic algorithms, *App. math and comp.* 158(1), 237–252.
3. Kollen, A., E. Pesch (1994), Genetic local search in combinatorial. Optimization, *Discrete Applied Mathematics and Combinatorial Operation Research and Computer Science*, 48, 273–284.
4. Nandi, S., P. Mukherjee, S. S. Tambe, R. Kumar, and B. D. Kulkarni (2002), Reaction Modeling and Optimization Using Neural Networks and Genetic Algorithms: Case Study Involving TS-1-Catalyzed Hydroxylation of Benzene, *Ind. Eng. Chem. Res.*, 41(9), 2159–2169.

CHAPTER 6

DATA PROJECTION, DIMENSIONALITY REDUCTION AND INPUT SELECTION

6.1 INTRODUCTION

In this chapter, we have developed/improvised a few AI and ML based algorithms for dimensionality reduction, data projection and input selection of chemical/biochemical processes. In the first sub-section, a recently proposed neural networks based Sammon's mapping is utilized for the dimensionality reduction of glass data and fault detection and diagnosis of a CSTR. In the second sub-section locally linear embedding method is proposed for the fault detection and diagnosis of invertase production process. The curvilinear component analysis is utilized for the fault detection and diagnosis of CSTR and invertase production process in the next section. The last sub-section of this chapter presents the fuzzy curves and surfaces approach for the input selection of heat exchanger and pH control process models.

6.2 NONLINEAR FEATURE EXTRACTION USING SAMMON'S MAPPING AND SAMANN*

Nonlinear principle component analysis (NLPCA), nonlinear feature extraction and nonlinear dimensionality reduction methods are important techniques in pattern recognition, exploratory data analysis, data mining, process monitoring and fault detection and diagnosis. Sammon's mapping (refer Section 2.5.5) is one of the well-known methods to conduct the stated tasks. A major drawback of this algorithm is that for the new data the mapping exercise needs to be conducted freshly. That is, the formalism is incapable of extending (generalizing) the knowledge gained in a mapping exercise to new data. To overcome this drawback, an artificial neural network-based formalism known as SAMANN (refer Section 2.5.5) has been recently proposed. This section provides a comparison of the Sammon's mapping and SAMANN methods for dimensionality reduction and low dimensional projection applications by conducting two illustrative case studies.

6.2.1 Introduction

Monitoring, classification and modeling of high dimensional multivariate process data are faced with significant practical difficulties such as inability to view high-dimensional data, complex unwieldy nature of the resultant models and models incapable of generalization. Feature extraction methods significantly assist in overcoming some of the stated difficulties. Feature extraction is essentially a dimensionality reduction technique that extracts a subset of new features from the original feature (data) set by means of some functional mapping while retaining as much information as possible [Fukunaga, 1991]. Feature extraction can avoid the "curse of dimensionality", improve generalization ability of classifiers and reduce computational load in modeling and pattern classification efforts. Principle

---

* Badhe Y. P., V. Wadekar, U. Phulwale, S. S. Tambe and B. D. Kulkarni, Proceedings of "*Conference of Research Scholars and Young Scientists* (*CRSYS*)," held at IIT, Kharagpur on Sept. 25-26, 2004, 21–27.

component analysis (PCA) and linear discriminant analysis (LDA) are commonly utilized techniques for feature extraction. A significant drawback of these methods is that they are linear methods and therefore incapable of extracting nonlinearly correlated features. The commonly used nonlinear feature extraction methods which overcome the stated difficulty are non-metric multi-dimensional scaling (MDS), principle curves, Sammon's mapping (SM) [Sammon, 1969], isomap and locally linear embedding (LLE). In the last two decades, artificial intelligence (AI) based methods namely, Kohonen's self-organizing map (SOM) and auto associative neural network (AANN) have also been utilized to perform nonlinear feature extraction. More recently, a novel neural network based formalism known as "SAMANN" (Mao and Jain, 1994; Ridder et al., 1997) has been introduced for nonlinear feature extraction. The present section demonstrates SAMANN efficacy by conducting two case studies namely (i) fault classification in a steady-state continuous stirred tank reactor (CSTR) and, (ii) dimensionality reduction of a glass data set comprising six types of glass compositions. Also, presented here is the comparison of the results using Sammon's mapping and SAMANN.

6.2.2 Case Studies

Process monitoring, control and predicting process behavior under normal and abnormal conditions are major challenges in the chemical industry. The aim of fault detection/diagnosis systems is to detect and diagnose process failures at an early stage so that corrective measures can be taken swiftly. The fault is defined as an abnormal process behavior due to equipment failure, equipment wear, or abnormal process disturbances. The task of determining whether a fault has occurred is called fault detection and determining the cause of the malfunction is termed fault diagnosis.

In this study, the results obtained using the SAMANN are compared with those from the Sammon's mapping (SM); the measure employed for performance comparison of the stated two methods is Sammon's stressn (E_{SAM}) (see Eq. 2.71). In the SAMANN based feature extraction, 80% of the randomly chosen data patterns were used for network training and the remaining 20% for testing the generalization ability of the network. SAMANN's architectural parameters and

learning rate values are given in Table 6.1; the momentum factor values were kept constant (= 0.01) for all SAMANN training experiments. For implementing the SM, a Matlab toolbox [Vesanto, 2000] was used while SAMANN code was written in VC++, and implemented on a Windows XP workstation. In SM, the value of the learning rate (α) was 0.2 for all the mapping simulations.

A. Fault classification in CSTR

Here, we consider a jacketed nonisothermal CSTR wherein an irreversible exothermic first-order reaction, A → B, takes place. The reactor is fitted with three proportional control loops that control the outlet temperature, reactor hold-up, and outlet concentration. The CSTR data of 1050 patterns was generated by simulating reactor's mass and energy balance equations [Vora et al., 1997]. The data correspond to the values of six CSTR variables and parameters namely, outlet reactor concentration, CSTR temperature, reactor hold-up, reactor output flowrate, jacket coolant flowrate and coolant temperature, corresponding to seven types of process faults. The faults are classified as: (i) input flow rate high, (ii) input flow rate low, (iii) inlet concentration high, (iv) inlet concentration low, (v) inlet temperature high, (vi) inlet temperature low, and (vii) decrease in heat transfer coefficient due to fouling [Vora et al., 1997]. The objective of Sammon's mapping and SAMANN is to reduce the dimensionality of the CSTR's six-dimensional variable and parameter space and perform feature extraction in a manner such that detection and diagnosis of faults is possible by visualizing the lower dimensional projection of the original high dimensional data set. Both SM and SAMANN methods were used to reduce the six dimensional data to two, three and four dimensional feature space and the results obtained thereby are listed in Table 6.1. The SM was implemented using all the 1050 patterns. Performance of both algorithms was evaluated in terms of the minimum value of Sammon's stress (E_{SAM}) achieved. As illustrative cases, the 2D projection of the original 6-dimensional data as obtained using SM and SAMANN methods are portrayed in Figure 6.1 and Figure 6.2, respectively. It can be seen that both methods have classified the seven fault data correctly in as many clusters in the lower (i.e., 2D) dimensional space. Subsequently, the low dimensioned mapping can be used online with the process to identify and diagnose any occurrence of single faults.

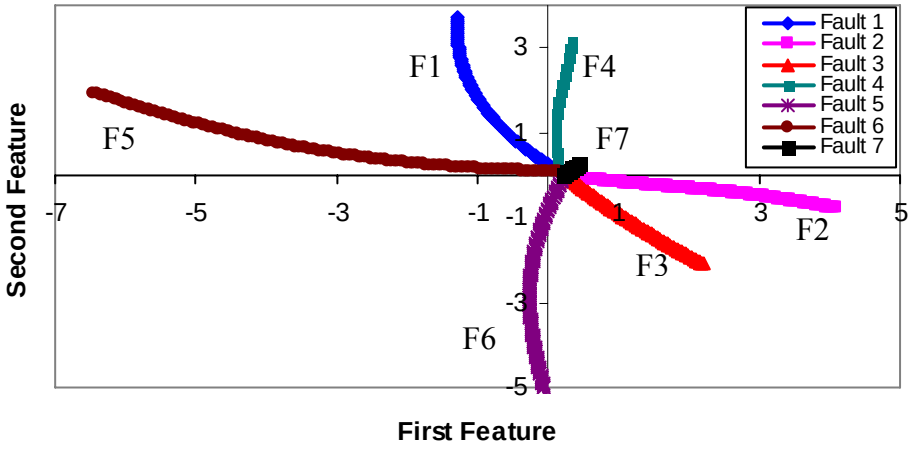

Figure 6.1: 2D projection of CSTR data by the multi dimensional scaling based Sammon's algorithm

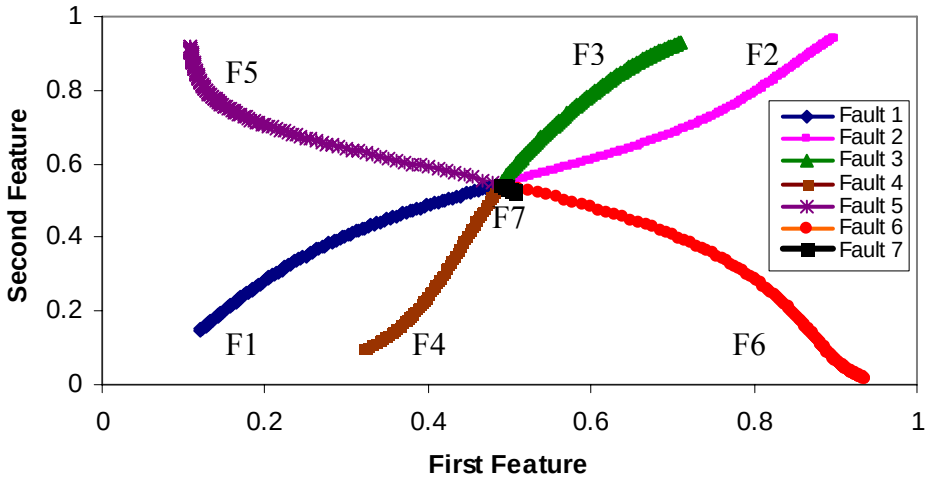

Figure 6.2: 2D projection of CSTR data by SAMANN

B. Dimensionality reduction of glass data

In this case study, a benchmark classification problem comprising glass data set (Prechelt, 1994) has been considered. The set of nine-dimensional 213 patterns contains property and composition values of six different types of glasses. The nine values correspond to the refractive index and glass composition in terms of elements namely Sodium, Magnesium, Aluminum, Silicon, Potassium, Calcium, Barium and Ferrous. Both SM and SAMANN algorithms were applied

to map the 9-dimensional inputs to lower, i.e., 2 to 5 dimensions and their performance is listed in Table 6.1.

From the stress values (E_{SAM}) listed in Table 6.1, it is seen that the SM has achieved lower stress magnitudes as compared to the SAMANN. The lower stress values yielded by the SM method in both case studies indicate that SM has preserved the inter-pattern distances in low-dimensional projections with higher accuracy when compared to the SAMANN. However, almost similar stress values for both training and test sets show a good generalization capability of the SAMANN. It is also observed that the CPU time requirements to achieve convergence are lower for the SM when compared to the SAMANN. The major advantage of the SAMANN however is that feature extraction for a new data set can be conducted without retraining the network, while in the case of SM, a fresh mapping exercise is required for each new data pattern.

6.2.3 Conclusion

This section illustrates nonlinear feature extraction using two novel methodologies namely SM and SAMANN. The efficacy of these methods is demonstrated by conducting two case studies where the objective was dimensionality reduction and the feature extraction of high dimensional data. In the case of CSTR, it was observed that both techniques are capable of efficient dimensionality reduction and clustering and therefore these can be utilized for process fault detection and diagnosis. Similarly, for glass data the methods exhibited excellent dimensionality reduction performance. To summarize SM and SAMANN are attractive methods for feature extraction and dimensionality reduction and hence can be gainfully employed in process monitoring.

Table 6.1: Results of Sammon's Mapping and SAMANN

Data sets	Projection	Sammon's Mapping (MDS based)			SAMANN				
		E_{SAM}	Time per cycle[@] (sec)	No. of Iterations	Structure[#]	E_{SAM} (training data)	E_{SAM} (test data)	Time per cycle[@] (sec)	Number of Iterations
CSTR	2D	2×10^{-3}	3.531	1000	6,32,10,2,0.5	0.052	0.051	12.6	570
	3D	1.02×10^{-4}	4.284	1000	6,20,8,3,0.65	0.037	0.033	14.3	465
	4D	8.0×10^{-6}	4.765	1000	6,30,12,4,0.45	0.0066	0.006	16.1	5,062
Glass	2D	0.0323	0.227	1000	9,40,17,2,0.5	0.074	0.083	0.75	10,000
	3D	0.0101	0.252	1000	9,34,18,3,0.75	0.047	0.05	0.81	11,811
	4D	0.0025	0.279	1000	9,36,16,4,0.35	0.039	0.04	0.92	5,746
	5D	0.0008	0.289	1000	9,28,12,5,0.65	0.0320	0.035	0.98	9,499

[#] Number of nodes in input, hidden – I, hidden – II, and output layers and η (learning rate) value.

[@] CPU time.

6.3 MONITORING AND FAULT DETECTION OF BIOCHEMICAL SYSTEMS USING LOCALLY LINEAR EMBEDDING*

A non-linear process monitoring technique based on locally linear embedding (LLE) formalism (see Section 2.5.4) is developed for a batch biochemical process. The LLE is a recently proposed method for non-linear mapping of multivariate data to low dimensional space. It is an unsupervised learning algorithm that computes low dimensional, neighborhood preserving embedding of high dimensional multivariate data. The LLE maps the data into a single global coordinate system and its optimizations do not involve local minima, which guarantees global optimality of the convergence. The efficacy of LLE in monitoring and fault detection is demonstrated for a biochemical process namely fermentative production of invertase. The performance of LLE is also compared with the well-known neural network based dimensionality reduction technique namely, auto associative neural networks (see Section 2.5.3). It is shown that for the specific fermentation process, the performance of LLE is better than that of the AANN.

6.3.1 Introduction

Historical data collected during bioprocess control typically includes information on *high*, *average* and *low* productive batches. Data also includes information on the consequences of performing specific control actions in response to problematic situations. Thus it is possible to develop appropriate models for evaluating the bioprocess performance and detecting and diagnosing faults by utilizing the routinely collected and stored records of process variables. In many processes, phenomenological knowledge such as reaction kinetics and mass transfer mechanisms underlying the system is obscure. This poses

---

* Phulwale U. S., K. M. Desai, Y. P. Badhe, V. A. Wadekar, S. S. Tambe, B. D. Kulkarni, Proceedings of *Conference of Research Scholars and Young Scientists* (*CRSYS*)," held at IIT, Kharagpur on Sept. 25-26, 2004, 78–84.

difficulties in selecting variables that will represent the system adequately and hence appropriate monitoring formalisms must be explored.

Owing to the availability of reliable hardware and biosensors, a large number of process variables are monitored during a typical bioprocess operation and thus their meaningful interpretation becomes difficult. It therefore becomes necessary to interpret the voluminous process data by efficient data reduction and projection systems. Such a system greatly simplifies the monitoring task of process operators and engineers. For the stated task, linear and nonlinear techniques, which reduce the dimensionality of the process data and thereby permit projections on a two-dimensional space, are often used.

There exist a number of methods for reducing the dimensionality of a multivariable data space without significantly losing the information content in the original data. In the commonly employed methods, new attributes (variables), which explain maximum amount of variance in the original data, are obtained by performing principal component analysis (see Section 2.5.1). Consequently, fewer variables (principal components) are required to represent a high dimensional multivariable data set. A significant drawback of the PCA method is that it captures only linear relationships between variables and therefore is not suitable if the variables are nonlinearly correlated, which is common in most chemical processes.

In the present work, a novel non-linear dimensionality reduction technique namely locally linear embedding (see Section 2.5.4) [Roweis and Saul., 2000] is successfully applied to a biochemical process. This method can be effectively used for monitoring, low dimensional projection and fault detection of biochemical processes. The case study utilizing LLE considers a batch fermentative production of invertase. The LLE formalism is used for classifying the fermentation batches into three classes namely *low*, *average* and *high* productive batches; it is also used for continuous monitoring of a given fermentation batch. Specifically, the LLE based two-dimensional projection of the original high dimensional process data is used for monitoring and classification of the fermentation process behavior. The results of LLE are also compared with those obtained from the auto associative neural network (see Section 2.5.3)

[Kramer, 1992; Aldrich, 1998 and Shimizu et al., 1998] based dimensionality reduction. The principal difference between an AANN and LLE is that while the former obtains new attributes (nonlinear PCs) which capture maximum variance in the high dimensional data, the LLE maps the high dimensional data on a low dimensional projection using a distance preserving mechanism.

6.3.2 Case Study: Monitoring Fermentative Production of Invertase

Fermentative production of invertase by *Saccharomyces carlsbergensis* by a fed batch fermentation process is chosen to demonstrate the efficiency of LLE formalism for process monitoring and fault detection. *Saccharomyces carlsbergensis* shows a biphasic nature of growth, i.e. it can utilize glucose as well as ethanol in the event of glucose scarcity [Dedem and Moo-Young, 2002]. In the first growth phase, glucose is utilized via aerobic fermentation with carbon dioxide and ethanol as major products. When glucose is completely exhausted, the ethanol produced earlier serves as a substrate for the further growth. In such cases, the extent of cellular growth and enzyme production depends upon the balance between the metabolic states of aerobic fermentation and respiratory growth. Thus, estimation of the current metabolic state can be made from the changes in the concentrations of glucose and ethanol in the broth. The major advantage of a fed batch bioreactor is that during fermentation, the feed composition and feed flow rate can be manipulated to maximize product formation. Thus, manipulation of the feed rate is an important aspect of the fed-batch operation from the control and optimization viewpoint. Recently, while performing feed profile optimization of the fed-batch system, Sarkar and Modak, [2003] have reported four different profiles for as many combinations of high and low initial substrate and biomass concentrations. Since all these feed profiles were nearly identical, only one feed profile has been considered as a reference feed profile in this study.

A. Invertase production model

The phenomenological model for the biphasic growth of *Saccharomyces carlsbergensis and* invertase production is given as [Pyun et al., 1989]

$$\frac{d}{dt}(x_t) = (\mu_G + \mu_A)x_t \tag{6.1}$$

$$\frac{d}{dt}(s) = -\frac{\psi s x_t}{v} + \frac{s_F F}{v} \tag{6.2}$$

$$\frac{d}{dt}(e) = (\pi_A - \pi_C)x_t \tag{6.3}$$

$$\frac{d}{dt}(v) = F \tag{6.4}$$

$$\frac{d}{dt}(c_{\text{inv}}) = (\eta_S \mu_G + \eta_A \mu_A)x_t \tag{6.5}$$

The following nonlinear feed rate profile that maximizes the streptokinase production [Sarkar and Modak, 2003] is used for the data generation.

q = 0.2 l/h for ($0 \le t \le 0.58$); q = 0 l/h for ($0.58 \le t \le 2.28$); q = q_c l/h for ($2.28 \le t \le 12.4$); q = 0 l/h for ($12.4 \le t \le 13$); where,

$$q_c = \frac{\psi x v}{s_F - s} + x_t v(-1.4892)(s)^{(1.2013)}(c_{inv})^{(-0.1046)} \tag{6.6}$$

The details of rate expression and kinetic model can be found in [Pyun et al., 1989]. The following values of model parameters and operating conditions are used in simulations:

$\mu_G^{\max}$ =0.39 h^{-1}, $\mu_A^{\max}$ = 0.11 h^{-1}, $Y_{x/s}^R$ = 0.52 g/g , $Y_{x/s}^F$ = 0.15 g/g , $Y_{p/s}^R$ = 0.33 g/g , $Y_{x/p}^R$ =0.67 g/g, e_0 = 0 g, $(c_{inv})_0$ = 0 kU/g, v_0 = 0.6 l, v_{max} = 1.5 l, s_F = 1.5/[v_{max} – v_0] g/l, k_p = 0.014 *g/L*, k_s = 0.021 *g/L* .

where,

x_t : Concentrations of cells (g/l)

s : Concentrations of glucose(g/l)

e : Concentrations of ethanol (g/l)

μ_G : Specific rates of growth on glucose (h^{-1})

μ_A : Specific rates of growth on ethanol (h^{-1})

ψ : Specific rates of growth on glucose consumption (g/g h)

π_A : Specific rates of ethanol production (g/g h)

π_C : Specific rates of ethanol consumption (g/g h)

η_S : Ratios of the specific invertase synthesis rate to the specific growth rate on glucose (kU/g cells)

η_A : Ratios of the specific invertase synthesis rate to the specific growth rate on ethanol (kU/g cells)

c_{inv} : Invertase activity (g/l)

q : Volumetric feed rate of glucose (l/ h)

s_F : Glucose feed concentration (g/l)

v : Fermenter volume (l)

B. Data generation

To generate process data for the invertase from *Saccharomyces carlsbergensis* process, the phenomenological model proposed by Pyun et al., [1989] has been used. It may however be noted that the model is used only to simulate the process and thereby generating process data. In real practice, historic data collected from the actual process runs can also be used for the LLE-based fault detection and diagnosis. The fed-batch invertase process is described in terms of five operating variables namely glucose concentration (s, g\l), ethanol concentration (e, g/l), bioreactor volume (v, l), biomass concentration (x_t, g/l) and invertase concentration (c_{inv}, kU/g). This case study aims at classifying the process measurements depending upon low, average or high production of invertase (c_{inv}). To generate process data involving low, average and high productive batches, the set of five ordinary differential equations (ODEs) representing the process dynamics (Eqs. 6.1 to 6.6) was simulated under varying sets of operating conditions. The ranges in which the initial values of the operating variables were varied are ($0.3 < v_0 < 0.6$, $0.05 < s_0 < 0.3333$ and $0.03 < x_{t0} < 0.1$). A total of 40

batches with varying initial conditions as described above were simulated over fermentation duration of 13 hrs. The values of four operating variables, i.e. s, v, x_t and e and the product quality variable c_{inv} computed at one hr intervals formed the process data set.

The simulated data shows significant batch-to-batch variation in the final product concentration in the range of 5 kU/g to 13 kU/g. This variation is mainly due to the differential rate of assimilation of glucose and ethanol by *Saccharomyces carlsbergensis*. An abnormal process behavior can occur due to equipment malfunction or process disturbances, which can affect final product concentration adversely. Thus, it is necessary to develop a model using historic values of process variables, which will classify a fermentation batch to be *low*, *average* or *high* productive batch. In the present case study, depending upon the final product (invertase) concentration, the batches are classified into low, average or high productive batches.

The simulated dataset can be viewed as a three-way array of size 40 (number of batches) × 4 (process variables) × 14 (time intervals). This array was unfolded into a two-way array of dimensions (560 × 4) by row-wise arranging the time dependent values of process variables from the 40 batches. The batches producing invertase upto 7 kU/g were considered as *low* productive batches; those producing invertase between 7 to 9.5 kU/g were considered as *average* productive batches and the batches producing invertase above 9.5 kU/g were *high* productive batches. In the data ten batches (140 patterns) belonged to the class *low* and 15 batches (210 patterns) each belonged to the classes *average* and *high*.

The first step in implementing LLE is to identify the neighborhoods of each data point. The results of LLE depend quite sensitively on the choice of the number of nearest neighbors. Several criteria, however, should be kept in mind while choosing this number [Roweis and Saul, 2000]. First, the algorithm can only be expected to recover embeddings whose dimensionality, d, is strictly less than the number of neighbors, K, and some margin between d and K is desirable to improve algorithm's robustness. Second, the algorithm is based on the assumption that a data point and its nearest neighbors can be modeled as locally linear; for curved manifolds, choosing too large K value will in general violate this

assumption. In the case where $K > D$ (indicating that the original data is itself low dimensional), each data point can be reconstructed perfectly from its neighbors, and the local reconstruction weights are no longer uniquely defined. In this case, further regularization must be added to break the degeneracy. In the present case the regularization parameter is taken constant and the only free parameter is, K, i.e., the number of neighbors.

6.3.3 Results and Discussions

The Figure 6.3 shows the 2D projection of the four dimensional fermentation operating variable data of all the 40 batches for $K = 12$ as provided by the LLE. Different K values were tried to get a more stable 2D projection. The projections were stable over a wide range of K values. As can be seen in Figure 6.3, the *low, average* and *high* productive batches have been collected in three separate clusters. The classification accuracies of LLE for low, average and high productive batches were 98.21%, 95.53% and 98.75%, respectively. The results of similar clustering exercise conducted using five-layered AANN are portrayed in Figure 6.4, For this analysis, an AANN of architecture 4 (number of neurons in input layer) × 12 (number of neurons in mapping layer) × 1 (number of neurons in bottleneck layer) × 12 (number of neurons in demapping layer) × 4 (number of neurons in output layer) trained using the error back propagation algorithm [Rumelhart et al., 1986] was used. In Figure 6.4, it is observed that AANN is also able to classify the data in three clusters. The AANNs classification accuracies for low, average and high productive batches were 94.82%, 95.18% and 95.00%, respectively. It may however be noted that AANN training is heuristic and therefore is much more time consuming as compared to the LLE procedure.

In another set of simulations, the intermediate time process data (4^{th} hr) was subjected to LLE–based classification and the results obtained thereby are depicted in Figure 6.5. As can be seen that, here again the LLE has classified the data in three well-defined clusters clearly indicating whether the current batch will yield either *low*, *average* or *high* concentration of the invertase. Such an indication enables the process operator in making a critical decision whether to proceed with the batch or not.

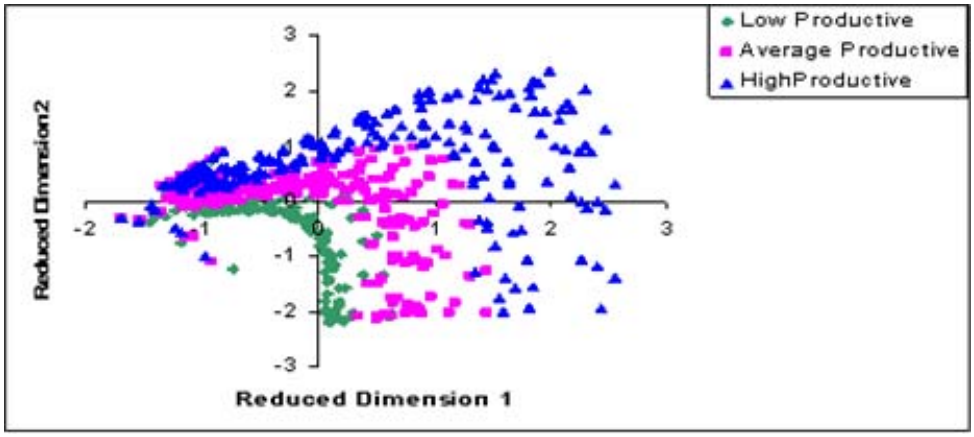

Figure 6.3: Two-dimensional Projection using LLE with K = 12

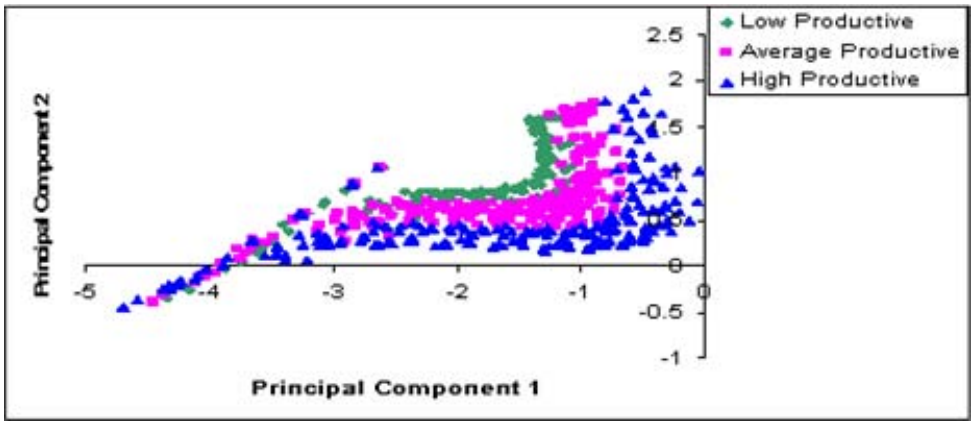

Figure 6.4: Two-dimensional Projection using AANN

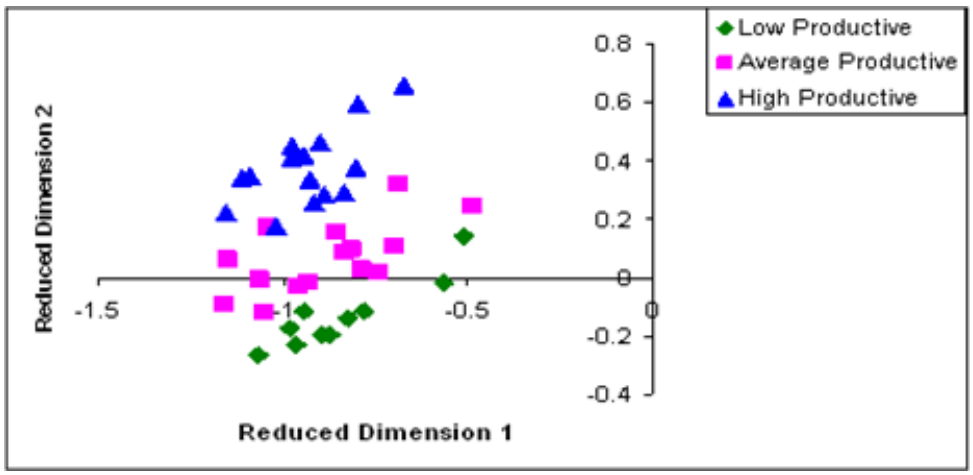

Figure 6.5: Two-dimensional Projection using LLE at the end of 10th hr for each batch

In the LLE–based low dimensional projections it was observed that during initial stages of the batch progress the LLE algorithm could not classify the batches correctly into low, average and high productive batches. This was primarily owing to the fact that the respective data did not contain sufficient variations to permit an accurate classification. However, LLE could classify the data with high accuracy during the later stages of the batch progression (see Figure 6.3 and Figure 6.4). It may also be noted that classification of data from new batches necessitates running the LLE freshly. However, the previously optimized value of K can be used in the fresh LLE simulations thus reducing the computational effort. In contrast, the AANN weights once optimized using the historic process data, can be used for projecting (or dimensionality reduction) the data from new batches.

6.3.4 Conclusion

In this section, a novel and recent non-linear dimensionality reduction technique namely LLE has been applied for monitoring and fault detection of a biochemical system. The LLE has efficiently projected the high dimensional data onto lower dimensions by means of eigen-analysis. The results are compared with an ANN based non-linear dimensionality reduction method namely AANN. Though both LLE and AANN have exhibited excellent clustering performance, the LLE was found to be much faster method than the AANN. Dimensionality reduction by LLE succeeds in recovering the underlying structure of manifolds, whereas linear embeddings by methods such as PCA maps faraway data points to nearby points in the plane, creating distortions in both the local and global geometry. Similar to PCA, the LLE algorithm is simple to implement, and its optimizations do not involve local minima. Also, it is capable of generating highly nonlinear embeddings and its main step involves a sparse eigen value problem that scales more naturally to large, high dimensional data sets.

6.4 PROCESS MONITORING AND FAULT DETECTION USING CURVILINEAR COMPONENT ANALYSIS*

This section demonstrates an application of a new nonlinear dimensionality reduction method, namely curvilinear component analysis (CCA) for the classification of chemical and biochemical process data. The CCA projects high dimensional data on to a low dimensional space by maintaining the distances between the original input data patterns in the new space. It performs vector quantization (VQ) followed by the nonlinear mapping of the quantized vectors. The aim of the present study is to demonstrate how CCA can be effectively used for the classification applications that involve nonlinear projection of a high dimensional input space on to a low, i.e. two-dimensional (2D) projected space. The study reports two case studies comprising a continuous stirred tank reactor (CSTR) and a batch biochemical (invertase production) process to illustrate the efficacy of the CCA for the process monitoring and fault detection and diagnosis applications.

6.4.1 Introduction

The modern day control systems monitor and store a large number of process variables continuously. Discovering the hidden structure and relationships among the data variables is facilitated by conducting a dimensionality reduction (DR) of the data set. The DR aims at reducing the unmanageable dimensions of data set to the data set that can be clearly and conveniently visualized without losing significantly the information content in the original data set. There are several advantages of dimensionality reduction such as compact representation of the data, ease of data storage and retrieval, filtering of noise, convenience in viewing the data and reduction in the numerical load during process modeling,

---

* Phulwale, U. S., B. Jeevan kumar, S. U. Patel, Y. P.Badhe, S. S. Tambe and B. D.Kulkarni. Presented in the "*Second Indian International Conference on Artificial Intelligence*", held on Dec 20-22, 2005, at the National Insurance Academy, Pune, India.

control, optimization, etc. Traditionally, DR is used as a data preprocessing tool for the subsequent "clustering" and "regression" tasks.

There exists a number of methods for reducing the dimensionality of a multivariable data space without losing the information content in the data significantly. In the commonly employed DR methods, new attributes (latent variables) explaining the maximum amount of variance in the original data, are obtained by performing the principal component analysis (see Section 2.5.1). Consequently, fewer latent variables (also known as "principal components") are required to represent a high dimensional multivariable data set. A significant drawback of the PCA method is that it extracts only the linear relationships between variables and therefore it is not suitable if the variables are correlated nonlinearly, which is common in most chemical and bio-processes. The nonlinear characteristics of these processes necessitate the usage of nonlinear dimensionality reduction techniques. Many techniques for nonlinear dimensionality reduction have been proposed recently in the literature such as the Nonlinear Multidimensional Scaling [Shepard et al., 1965] and Sammon's Nonlinear Mapping [Sammon, 1969]. However, these methods suffer from huge computational costs and inability to unfold strongly nonlinear data [Demartines et al., 1997].

The curvilinear component analysis (see Section 2.5.2) is a recently proposed [Demartines et al., 1997] non-linear dimensionality reduction method, which overcomes some of the major shortcomings of the other DR methods and also has the ability to reduce the dimensionality of strongly nonlinear data. The CCA aims exploring hard data structures and finding a revealing representation by unfolding the manifold spanned by the data. The major advantage of CCA is its speed of convergence and accuracy.

Process monitoring, control and fault detection/diagnosis (i.e. identifying the *normal* and *abnormal* behavior of a process and causes thereof) are the major challenges in the chemical and bioprocess industry. The aim of the fault detection/ diagnosis (FDD) systems is to detect and diagnose process failures at an early stage so that corrective measures can be taken speedily. In a continuous operation, malfunctions or faults can develop owing to significant deviations in the operating

variables. In most processes, the data pertaining to the normal and abnormal process behavior are continuously logged and archived. In the other type of process operation, that is batch mode, the monitored and archived data typically includes information on *high*, *average* and *low* productive batches. The data also contain information on the consequences of performing a particular control action in response to a problematic situation. Thus it is possible to develop appropriate models for evaluating and classifying the performance of a batch/continuous chemical or a biochemical process and detect and diagnose faults by utilizing the routinely collected and stored records of the process variables.

Interpretation of the voluminous data collected during a process operation can be achieved by using an efficient data reduction and projection system. Such a system greatly simplifies the monitoring task of process operators and engineers. Thus, in this study, the CCA is illustrated for addressing the problem of nonlinear dimensionality reduction and classification of the process data. Specifically, the CCA is used for the classification of faults in a non-isothermal CSTR and classification of batches according to the yield of an invertase process.

6.4.2 Case Study – I (Non-isothermal CSTR)

In this case study, the first order irreversible reaction carried out in a jacketed non-isothermal CSTR has been considered. The input vectors for the CCA consisted of 6 process variables namely: (i) outlet reactant concentration (C_A), (ii) CSTR temperature (T), (iii) reactor hold up (V), (iv) reactor output flow rate (F), (v) jacket coolant flow rate (F_j) and (vi) coolant temperature (T_j). The aim of the present case study is to identify the type and the magnitude of a fault occurring in a steady-state CSTR operation from the available values of the stated six variables.

We consider a few but representative single faults that can occur during the steady-sate operation of a CSTR when process parameters deviate by a fixed amount from their normal (or set) values. The specific faults which can influence the steady-state CSTR behavior have been identified as [Vora et al., 1997]: (i) input flow rate (F_0) high, (ii) input flow rate low, (iii) inlet concentration (C_{A0}) high, (iv) inlet concentration low, (v) inlet temperature (T_0) high, (vi) inlet

temperature low, and (vii) decrease in heat transfer coefficient (U) due to fouling. The inlet concentration can deviate if CSTR happens to be a downstream unit. It is assumed that these faults occur in a mutually exclusive manner; that is, only one out of the seven types of fault can occur at any given time. The process data for the fault classification analysis was generated using the CSTR's phenomenological model [Venkatasubramanian et al., 1990]. The historical data collected from a physically running process can also be used in the classification. Any deviation in the model parameters would eventually affect the steady-state values of the above stated six state variables (C_A, T, V, F, F_j and T_j) and the magnitudes of these state variables are usually available as the measured data.

A. Simulation of faults

To generate process data, model equations described in Venkatasubramanian et al. [1990] were solved to obtain the steady-state values of the six state variables. The values of the model parameters used in the steady-state simulations were: F_0 = 1.15 $m^3\ hr^{-1}$, F = 1.15 $m^3\ hr^{-1}$, $V = V_{set}$ = 1.35 m^3, C_{A0} = 8000 gmol m^3, $T = T_{set}$ = 333 K, $T_{j0} = T_0$ = 295 K, V_j = 0.1 m^3, $\alpha = 7.08\times10^{10}\ hr^{-1}$, C_j = 4 J(gmol $^0K)^{-1}$, F_j = 1.4 $m^3\ hr^{-1}$, λ = -70000 $J(gmol)^{-1}$, R = 8.3262 $J(gmolhr)^{-1}$, E = 70000 $J(gmol)^{-1}$, $U = 3 \times 10^6$, $J(hrm^2K)^{-1}$, $\rho = 8 \times 10^5$ $gmol(m^3)^{-1}$, $\rho_j = 10^6$ $gmol(m^3)^{-1}$, k_c = 0.2 $m^3(hr\ K)^{-1}$, Cp = 3 $J(gmol\ K)^{-1}$, A = 25 m^2, k_L = 10 hr^{-1}, and $k_a = 8.8 \times 10^{-5}$ $m^6(gmol\ hr)^{-1}$. All the seven faulty process conditions were simulated separately. Each of the seven data sets so generated corresponds to 0.1% to 15% deviation (at the interval 0.1%) from the normal value of the individual process parameter responsible for the malfunction. Thus, each fault is represented by 150 patterns. Subsequently, all the seven data sets were combined to form a single input set for the CCA network training and the pattern consisting of the steady state values of state variables under no fault condition was added to this cumulative set. The data structure of the resultant set is given in Table 6.2. This set can be visualized as a matrix of size [1051, 6] consisting of 1050 patterns representing seven single faults and the remaining one portraying the normal process behavior.

B. Results and discussion

For reducing CSTR's six dimensional steady-state data to two dimensions, the CCA parameters were set as follows. The algorithm (see Section 2.5.2) was run for 1000 training epochs (t = 1000); the learning rate decreased according to $\alpha(t) = \frac{\alpha_0}{(1+t)}$ where $\alpha_o = 0.5$ and F_λ was set equal to 1/(1+t). The CCA-based dimensionality reduction and the classification results are plotted in Figure 6.6. It can be seen from the figure, that the CCA has correctly classified the process data into seven faults even in the lower (i.e., 2) dimensional projection.

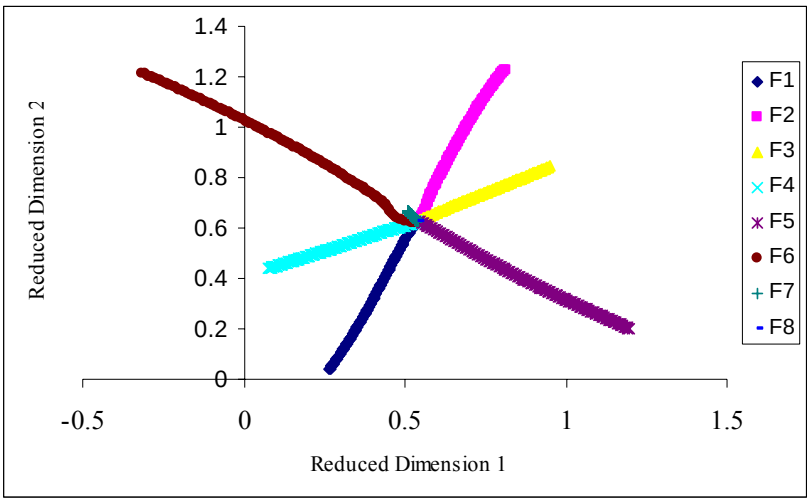

Figure 6.6: Nonlinear Projection of 6-dimensional steady-state CSTR data in 2-dimensions

Also, it is seen that the clusters depicting various faults are widely separated and their boundaries do not overlap. Thus, from the location of the 2-D projection of the steady-state values of the CSTR variables, it is possible to identify clearly the occurrence of any one of the seven faults. It is also possible to identify CSTR's normal steady-state behavior (the region at the centre of seven fault clusters).

Table 6.2: Nature and magnitudes of seven CSTR faults

No.	Nature of Fault	Fault code	Input Data for CCA	
			Pattern no(s).	Fault magnitude* (%)#
1	Input flow rate high	F1	1-150	(+) 0.1-14.9
2	Input flow rate low	F2	151-300	(-) 0.1- 14.9
3	Inlet conc. high	F3	301-450	(+) 0.1-14.9
4	Inlet conc. low	F4	451-600	(-) 0.1- 14.9
5	Input temp. high	F5	601-750	(+) 0.1-14.9
6	Input temperature low	F6	751-900	(-) 0.1- 14.9
7	Heat transfer coeff. low	F7	901-1050	(-) 0.1-14.9
8	Normal operation	-	1051	-

* Percent deviation from the normal operating value of the parameter defined in column 2.

Data generated at the deviation intervals of 0.1 %.

6.4.3 Fermentative Production of Invertase

In this case study, the batch fermentative production of invertase by *Saccharomyces carlsbergensis* is chosen to demonstrate the efficacy of the CCA formalism for the process monitoring and fault detection tasks. *Saccharomyces carlsbergensis* shows a biphasic nature of growth, i.e. it can utilize glucose as well as ethanol in the event of glucose scarcity [Pyun et al., 1989]. The major advantage of a fed batch bioreactor is that during fermentation, the feed composition and feed flow rate can be manipulated to maximize the product formation. Thus, manipulation of the feed rate is an important aspect of the fed-

batch operation from the control and optimization viewpoint. An efficient process monitoring and fault detection formalism is expected to greatly facilitate the manipulation of the feed rate.

A. Phenomenological model for invertase production

The phenomenological model for biphasic growth of *Saccharomyces carlsbergensis and* invertase production [Pyun et al., 1989] is given in section 6.3.2A. The following nonlinear feed rate profile that maximizes the streptokinase production [Pyun et al., 1989] is used for the data generation; q = 0.2 l/h for ($0 \le t \le 0.58$); q = 0 l/h for ($0.58 \le t \le 2.28$); $q = q_c$ l/h for ($2.28 \le t \le 12.4$); q = 0 l/h for ($12.4 \le t \le 13$), where,

$$q_c = -\frac{\psi s v}{s_F - s} + x_t v(-1.49)(s)^{1.2}(c_{inv})^{-0.1} \tag{6.7}$$

The details of the rate expression and kinetic model can be found in Pyun et al., [1989]. The following values of model parameters and operating conditions are used in simulations: $\mu_G^{\max} = 0.39$ h^{-1}, $\mu_A^{\max} = 0.11$ h^{-1}, $Y_{x/s}^R = 0.52$ g/g , $Y_{x/s}^F = 0.15$ g/g , $Y_{p/s}^R = 0.33$ g/g , $Y_{x/p}^R = 0.67$ g/g, $e_0 = 0$ g, $(c_{inv})_0 = 0$ kU/g, $v_0 = 0.6$ l, $v_{max} = 1.5$ l, $s_F = 1.5/[v_{max} - v_0]$ g/l, $k_p = 0.014$ *g/L*, $k_s = 0.021$ *g/L* .

B. Simulation of phenomenological model

To generate process data for the invertase from *Saccharomyces carlsbergensis* process, the phenomenological model proposed by Pyun et al., [1989] has been used. The model is used only to simulate the process and thereby generating the process data. The fed-batch invertase process is modeled in terms of five operating variables namely glucose concentration (s, g/l), ethanol concentration (e, g/l), bioreactor volume (v, l), biomass concentration (x_t, g/l) and invertase concentration (c_{inv}, kU/g). This study aims at classifying process measurements depending upon the *low, average* or *high* production of the invertase (c_{inv}). To generate process data involving low, average and high productive batches, the set of five ordinary differential equations (ODEs) (Eqs. 6.1 to 6.5 and Eq. 6.7) representing the process dynamics was simulated under

varying sets of operating conditions. The ranges in which the initial values of the operating variables were varied have been: ($0.3 < v_0 < 0.6$, $0.05 < s_0 < 0.3333$ and $0.03 < x_{t0} < 0.1$). A total of 40 batches with varying initial conditions as described above were simulated over fermentation duration of 13 hrs. The values of the four operating variables i.e. s, v, x_t and e and the product quality variable, c_{inv}, computed at one hr intervals formed the process data set.

The simulated data shows a significant batch-to-batch variation in the final product concentration in the range of 5 kU/g to 13 kU/g. This variation is mainly due to the differential rate of assimilation of glucose and ethanol by *Saccharomyces carlsbergensis*. An abnormal process behavior can occur due to the equipment malfunction or process disturbances, which can affect the final product concentration adversely. Thus, it is necessary to develop a model that will classify a fermentation batch to be either *low*, *average* or *high* productive batch.

The simulated data-set can be viewed as a three-way array of size 40 (number of batches) × 4 (process variables) × 14 (time intervals). This array was unfolded into a two-way array of dimensions (560 × 4) by arranging row-wise the time dependent values of the four process variables from the 40 batches. The batches producing invertase upto 7 kU/g were considered as *low* productive batches; those producing the invertase between 7 to 9.5 KU/g were considered as *average* batches and the batches producing above 9.5 KU/g of invertase were *high* productive batches. In the data, 10 batches (140 patterns) belonged to the class *low* and 15 batches (210 patterns) each to classes *average* and *high*.

C. Results and discussion

The parameter values used in the CCA based classification were $t = 3000$ and the learning rate decreased according to $\alpha(t) = \frac{\alpha_0}{(1+t)}$ (where $\alpha_o = 0.5$). In the above-stated classification task, the dimensionality of the four dimensional vectors of process operating variables was reduced to two. The results of the CCA-based low-dimensional projection of the process data are depicted in Figure 6.7, where it is seen that the data have been collected in three clusters describing low, average and high invertase production batches, respectively. In the region between x =

0.377 and $x = 1.67$, the points are well separated in three well-defined regions thus facilitating identification of *low*, *average* and *high* invertase production batches. The CCA could classify the data with approximately 99% accuracy. We also conducted dimensionality reduction and classification using the linear PCA. However, the PCA-based classification results were poor as compared to those obtained using the CCA. The results of the non-linear dimensionality reduction depicted in Figure 6.7 indicate that the CCA is an attractive method for viewing and classifying nonlinearly correlated bioprocess data in lower dimensions.

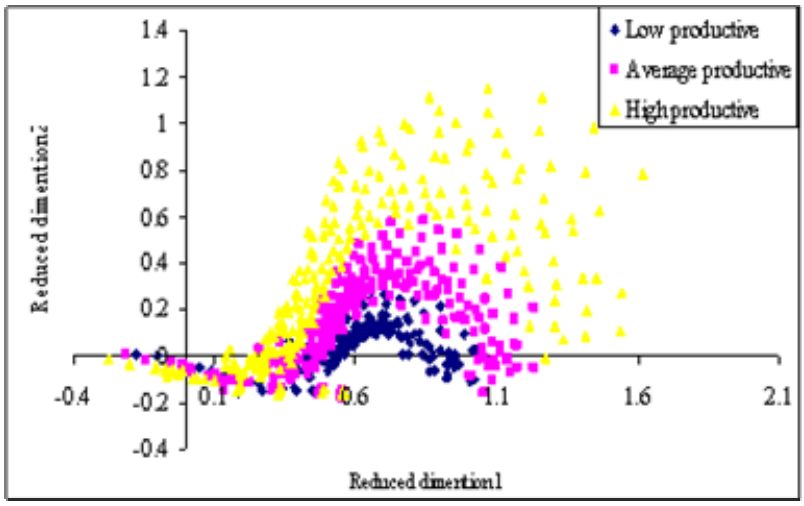

Figure 6.7 Nonlinear Projection of 4-dimensional invertase process data in two-dimensions

6.4.4 Conclusion

In this study, a state-of-the-art nonlinear dimensionality reduction method namely curvilinear component analysis has been employed to obtain low-dimensional projections of high dimensional chemical process data. In section 6.4.2, the steady-state data from a non-isothermal CSTR operating under faulty conditions was considered. The CCA could reduce the dimensionality of the data from 6D to 2D with an excellent precision. Also, the reduced dimensioned data formed well-defined clusters defining seven fault classes. In the case study described in section 6.4.3, the CCA could reduce the dimensionality of the invertase batch process data from four to two. Here, most of the data was classified in three well-defined clusters defining *low*, *average* and *high* productive batches. To summarize, the results of this study suggest that the CCA is an

attractive strategy for reducing the dimensionality of high dimensional nonlinearly correlated data. The advantage of the CCA is that it allows process engineers and operators a convenient single-window view of the process variables. This feature is advantageous in process monitoring, control and fault detection and diagnosis in that the process operator/engineer can view the process operation easily. It is also possible to use the dimensionality reduced data for modeling purposes thereby substantially lowering the computational load.

6.5 SELECTION OF MODEL INPUTS USING FUZZY CURVE AND FUZZY SURFACE METHODS

6.5.1 Preamble

Modern day chemical processes are large and complex entities with a plethora of equipment and sub-processes. These processes are characterized in terms of a number of independent and dependent variables as also parameters. A process model is required in a variety of process engineering tasks such as prediction of performance in terms of conversion, selectivity, efficiency, etc., and optimization, control, fault detection and diagnosis and process monitoring. Among the three approaches that are available for modeling, namely *phenomenological*, *empirical* and *black-box*, the first one poses significant difficulties owing to an insufficient knowledge about the physico-chemical phenomena underlying large chemical processes and therefore empirical and black-box modeling formalisms are resorted to. The implementation of these formalisms comprising, for example, linear/nonlinear regression and artificial intelligence methods becomes tedious and numerically intensive when a process is monitored in terms of a large number of input-output variables. Also, many a times the monitored variables are correlated thereby making such variables redundant. It is therefore advisable to reduce the size of the data-base used in modeling the process by identifying the influential input variables and ignoring the unimportant ones. Selection of most important process input variables leads to several advantages such as reduced numerical effort involved in modeling, faster model development, ease in process monitoring and an accurate and generalization capable model. Accordingly, in this section, a novel and recently proposed fuzzy logic based formalism is illustrated for the input selection. In what follows, an overview of various dimensionality reduction and input selection formalisms is presented followed by the results of two case studies wherein the fuzzy logic based method has been implemented for the selection of important inputs of two chemical engineering systems.

6.5.2 Introduction

In today's large and complex commercial chemical processes, a number of factors influence the reaction and mass and heat transfer phenomena. These processes are characterized in terms of a number of independent (causal) and dependent (response) variables such as reactant concentration, temperature, pressure, conversion and selectivity, as also product quality variables. The performance of a process model fitted by exclusively data-driven AI and ML based nonlinear modeling techniques is critically dependent on the dimensionality, statistical distribution and size of the input-output data set used in constructing the model. In principle, the AI-based and ML-based nonlinear empirical modeling techniques do not require to reduce the model's input space although as size of the input space increases the solution to a nonlinear modeling problem converges to a local rather than the global optimum. Also, the AI and ML based modeling algorithms are iterative in nature and most of the algorithm-specific parameters are adjusted by heuristic methods. Thus, their implementation is a numerically intensive task especially while modeling nonlinear systems with a large number of inputs and outputs. If the input space of an ANN or ML based model has a high dimensionality then the resulting model becomes complex due to the large number of terms that it contains. Such a model does not possess the much desired good generalization performance owing to which it makes poor predictions for a new set of inputs, which are not part of the data set used in constructing the model. Often, the causal variables, which form the input space of a process model, are either linearly or nonlinearly correlated. Such correlated inputs unnecessarily increase the dimensionality of the model's input space. Also, the sensitivity with which an operating condition variable (model input) affects the model output may vary significantly with some of these variables exhibiting only a negligible effect on the output. It is therefore essential to identify the influential input variables and thereby reduce the dimensionality of the input space of a process model since it leads to several advantages alluded to above. Particularly, identifying a small number of influential inputs and ignoring the unimportant ones results in a parsimonious yet accurate and reliable process model with reduced complexity and improved generalization capability. There exists a number of methods for

input identification (selection) which reduces the dimensionality of the input space.

To remove nonlinearly correlated input variables, nonlinear dimensionality reduction techniques such as autoassociative neural networks (AANN) (refer Section 2.5.3) and Sammon's mapping based neural network (refer Section 2.5.5) are commonly used. These neural network based methods are however computationally expensive. Thus, in this study a new and novel AI-based strategy known as "fuzzy curves and surfaces (FCS) (refer Section 2.5.6)" has been presented for identifying important input variables and thereby reducing the dimensionality of the input space of nonlinear process models. This method requires lesser computational effort than the other heuristic nonlinear input selection and dimensionality reduction methods such as AANN. Specifically, the FCS method has been utilized for identifying important inputs of following chemical processes: (i) heat exchanger system and (ii) pH control system. Additionally, the performance of the FCS method is rigorously compared using an ANN-based sensitivity computation method yielding important inputs of a model (see Section 3.4.7). This study shows that the FCS is capable of efficiently identifying the hierarchy of the input variables of a process model according to their influence on the model output. For comparing the performance of the FCS method, this study uses ANN-based input sensitivity method (refer Section 3.4.7) which computes the sensitivity of the output variable towards changes in each input of an ANN model. Upon computation of sensitivities, only those inputs exhibiting high sensitivity towards an output can be retained while ignoring less sensitive inputs. Section 2.5.6 highlights the FCS technique used in the identification of the important inputs.

6.5.3 Case Studies

Consider a dynamic system described as

$$y_{t+1} = f\left(y_t, y_{t-1}, ..., y_{t+\alpha}, u_t, u_{t-1}, ..., u_{t-\beta}\right) \tag{6.8}$$

where y refers to the control variable, y_{t+1} describes the one-step-ahead value of the control variable, y, u represents the manipulated variable, and α and β are the

time lags of the controlled and manipulated variables, respectively. Fitting of Eq. (6.8) from the process data is known as "process identification." The identified model can then be used in the model based controller design. In the above equation, some of the lagged variables may have more influence on $y(t+1)$ than others. Thus, in order to accurately fit the system dynamics, the number of lagged variables (α and β) must be chosen judiciously and heuristically. Also, having chosen α and β, the most significant individual lagged variables need to be determined. In the present study, FCS has been employed to determine the most influential lagged variables of the heat exchanger and pH control systems so that the resultant models accurately describe the dynamics of these systems.

In order to mimic the real-life scenario, all the simulated data sets of the stated systems were adulterated with a maximum of 1% Gaussian noise. In order to find the significant inputs, a large number of inputs (high α and β) have been subjected to the FCS method. To check the performance of the significant lagged variables selected by the FCS, two ANN models were built separately; the first of these models uses all the lagged variables as inputs and the other model uses the most significant lagged variables determined by the FCS.

A. Nonlinear heat exchanger control system

This example considers a shell and tube heat exchanger (see Figure 6.8) wherein process fluid enters the exchanger vessel with inlet temperature, T (normalized steady-state value, T_s = 0.15 ^{o}C), and flow rate, F (F_s = 0.50 l/min). Using a heater, the fluid temperature is increased to a higher value, Y, at the exchanger outlet (Y_s = 1). The heat exchanger dynamics are described by the following ordinary differential equation (Kulkarni et al., 1999):

$$\frac{dY}{dt} = \frac{-FY}{V} + \frac{FT}{V} + \frac{U^2}{RC_P V} \tag{6.9}$$

where U refers to the normalized heater voltage (U_s = 0.707) and (RC_P = 1) is the system time constant. For this process, heater voltage input, U, serves as the manipulated variable and the exchanger outlet temperature, Y, is the controlled variable. To generate input-output process data, Eq. (6.9) was integrated by randomly perturbing U between 0.5 and 1.4. Probability of variation in U was 0.15. The dynamics of the heat exchanger system (Eq. 6.9) was simulated till time equals 5000 minutes. For training the model, the process data set covering 5000

minutes of operation was divided in 80:20 ratio into two sets namely, *training* and *test* sets. Next, ANN model's architectural parameters (number of nodes in two hidden layers) were optimized using these training and test sets. A maximum of six lagged variables of y and u (α, β = 6) were utilized and an ANN model of following form was constructed using the EBP algorithm.

$$y_{t+1} = f_1\left(y_t, y_{t-1},..., y_{t-\alpha}, u_t, u_{t-1},..., u_{t-\beta}\right) \tag{6.10}$$

where f_1 refers to the function approximated by the ANN-based model with α=6 and β=6. This model although has 14 inputs, all of them are not influential in predicting y_{t+1} and thus ANN-based sensitivity analysis (see Section 3.4.7) was carried out for the above-stated 14 inputs. The results obtained thereby are plotted in Figure 6.9 in the decreasing order of the sensitivity values. The first six input variables from the plot can be identified as the most important model input variables. The ANN model when these variables are used in its input space can be written as:

$$y_{t+1} = f_2\left(y_t, y_{t-1}, y_{t-2}, y_{t-3}, u_{t-6}, u_{t-3}\right) \tag{6.11}$$

It can thus be seen that the ANN-based sensitivity computation method has reduced the input space from 14 to 6 inputs. Based on these reduced number of inputs, a new ANN model was built.

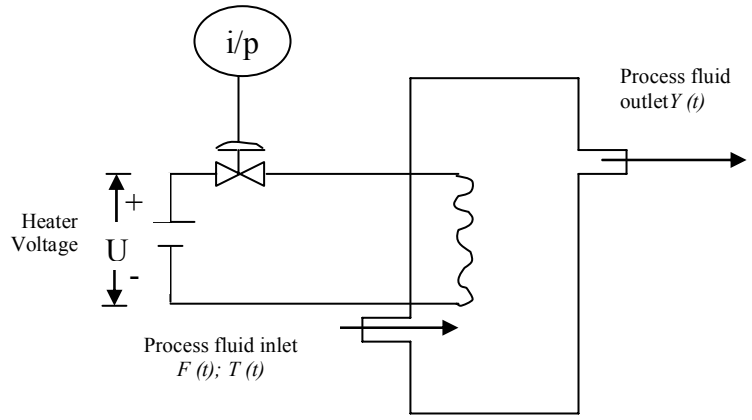

Figure 6.8: Heater voltage is the manipulated variable and exchanger outlet temperature is the controlled variable

Next, FCS was utilized to rank the influential inputs. For this purpose, the algorithm described in section 2.5.6 was utilized. The number of inputs ranked was 14. The influential input variables determined by the FCS were y_t, $y_{t\text{-}2}$, $y_{t\text{-}4}$, u_t, $u_{t\text{-}4}$ and $u_{t\text{-}6}$. Thus, the one-step-ahead predictor model for the heat exchanger can be written as:

$$y_{t+1} = f_3(y_t, y_{t-2}, y_{t-4}, u_t, u_{t-4}, u_{t-6}) \tag{6.12}$$

It can thus been seen that ANN-based sensitivity analysis and FCS have identified different sets of lagged variables as influential although y_t, $y_{t\text{-}2}$ and $u_{t\text{-}6}$ are common to both these sets.

To gauge the performance of FCS-identified important inputs an ANN model was constructed using the inputs described in Eq. (6.12). The root-mean-square-error (*RMSE*), mean percentage error (*%error*) and the correlation coefficient (*CC*) between the actual one-step-ahead y values and those predicted by the ANN model with 14 inputs as also FCS identified six significant variables for the training and test data set are given in Table 6.3. Additionally, listed in the table are the values corresponding to an ANN model constructed using the important inputs identified by the sensitivity analysis. In the table, f_1 refers to the ANN model obtained using all 14 inputs (Eq. 6.10), f_2 refers to the ANN model (Eq. 6.11) with reduced numbers of inputs obtained from the sensitivity analysis of the f_1 model, and f_3 refers to the ANN model (Eq. 6.12) obtained using the reduced number of inputs identified by the FCS. As can be noted from Table 6.3, the *RMSE* and *%error* values have decreased for models with reduced input dimension as compared to the model using all the 14 inputs. It can also be seen that the FCS reduced input space has imparted maximum improvement in the ANN model's predictive and generalization performance. The six important inputs identified by the sensitivity method also perform better than the 14 inputs in predicting y_{t+1} albeit on a lower scale when compared with those identified by the FCS.

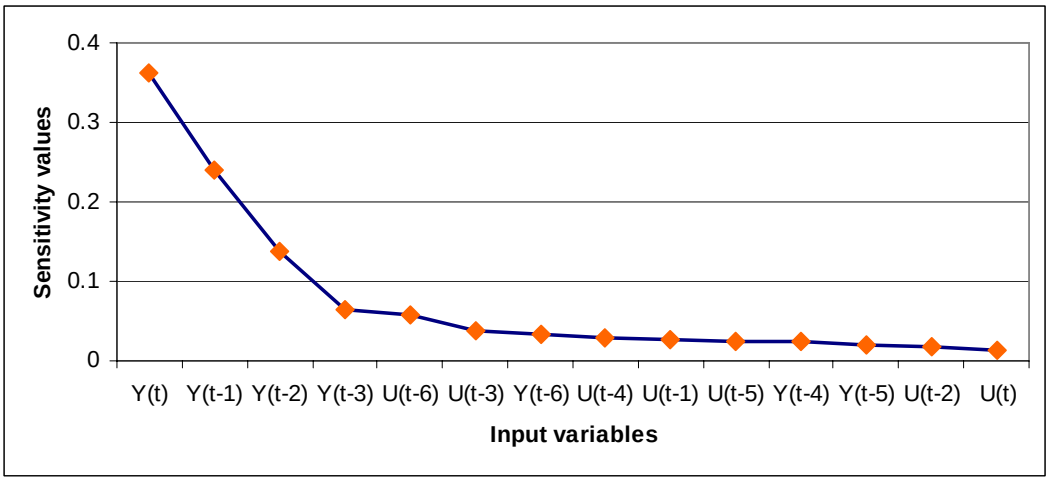

Figure 6.9: Sensitivity of lagged variables of Heat Exchanger system

B. Nonlinear pH control system

The process is a continuous stirred tank reactor (CSTR) (see Figure 6.10) wherein hydrochloric acid and sodium hydroxide streams are mixed and effluent stream *pH* is monitored. The objective is to control/maintain the effluent *pH* at a given set point by manipulating the NaOH flow rate. Dynamics of the *pH* control process assuming perfect mixing, reaction at equilibrium, constant volume and constant density is described by a single ordinary differential equation (Kulkarni et al., 1999):

$$\frac{dC}{dt} = \frac{[C + c_a C^2 - C^3]P + [C - c_b C^2 - C^3]Q}{(1 + C^2)\,60\,(V)} \tag{6.13}$$

$$pH = -\log_{10}(C/10^7) \tag{6.14}$$

where C denotes the dimensionless concentration of the hydrogen ions, *pH* (control variable) represents the effluent *pH*, c_a and c_b refer to dimensionless concentrations of HCl and NaOH inlet streams, respectively, V is the CSTR volume and, P and Q (manipulated variable) respectively represent the inlet flow rates of HCl and NaOH. To obtain the dynamic process input-output data Eq. (6.13) was integrated using Runge-Kutta method by varying Q randomly between 30 and 70 l min^{-1}; the probability of variation in Q at any instant was 0.25.

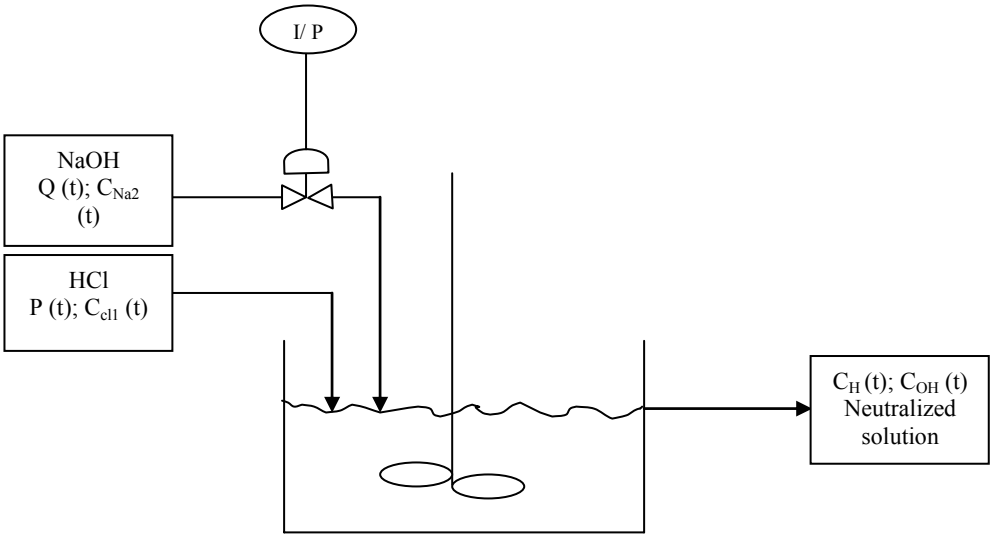

Figure 6.10: Schematic of a pH neutralization CSTR

In order to identify the dynamics of the nonlinear pH control system an ANN model with six lagged variables of the control and manipulated variables (α, $\beta = 6$) was constructed. The model has following form:

$$y_{t+1} = f_4\left(y_t, y_{t-1},..., y_{t-\alpha}, u_t, u_{t-1},..., u_{t-\beta}\right) \tag{6.15}$$

where y is the controlled variable (pH of the system), u is the manipulated variable (NaOH inlet flow rate, Q) and α, β representing the number of flags have a magnitude of six. Next, sensitivity analysis of the ANN model (Eq. 6.15) was conducted and the results are plotted in Figure 6.11 in the decreasing order of the sensitivities of model inputs. In Figure 6.11, it is seen that apart from y_t and u_t, six lagged variables of y, namely $y_{t\text{-}1}$, $y_{t\text{-}6}$, $y_{t\text{-}2}$, $y_{t\text{-}3}$, $y_{t\text{-}4}$ and $y_{t\text{-}5}$ exhibit higher sensitivity towards y_{t+1} than the six lagged variables of the manipulated variable u. Accordingly, another ANN model of the following form was built using y_t, u_t and six lagged variables of the control variable, y as model inputs.

$$y_{t+1} = f_5\left(y_t, u_t y_{t-1},, y_{t-2}, y_{t-3}, y_{t-4}, y_{t-5}, y_{t-6}\right) \tag{6.16}$$

Next, FCS was utilized for the selection of important input variables of the ANN model (Eq. 6.15) and the equation explaining the dependency of lagged values of control and manipulated variables on the one-step-ahead values of control variable (i.e., y_{t+1}) using FCS identified five important inputs is as given below.

$$y_{t+1} = f_6(y_t, y_{t-3}, y_{t-6}, u_{t-1}, u_{t-2}) \quad (6.17)$$

The *RMSE*, *%Error* and *CC* between the actual one-step-ahead *pH* (y_{t+1}) and those predicted by the ANN models using all the 14 input variables as also using the significant variables identified by the sensitivity analysis and FCS for *training* and *test* data sets are given in Table 6.3. In the table, f_4 refers to the ANN model built using all the 14 inputs as described by Eq. 6.15, f_5 refers to the ANN model (Eq. 6.16) using the inputs identified by the sensitivity analysis and f_6 refers to the ANN model (Eq. 6.17) constructed using inputs identified by the FCS. Similar to the heat exchanger control system, in this case study also an improvement is observed in the prediction and generalization performance of the ANN model built using the significant inputs identified by the sensitivity analysis and FCS as compared to the model built using all the 14 inputs. Also, FCS performance in identifying significant inputs is better as compared to the sensitivity analysis method which can be validated from the lowest *RMSE* and *%error* values and highest *CC* values (refer to Table 6.3).

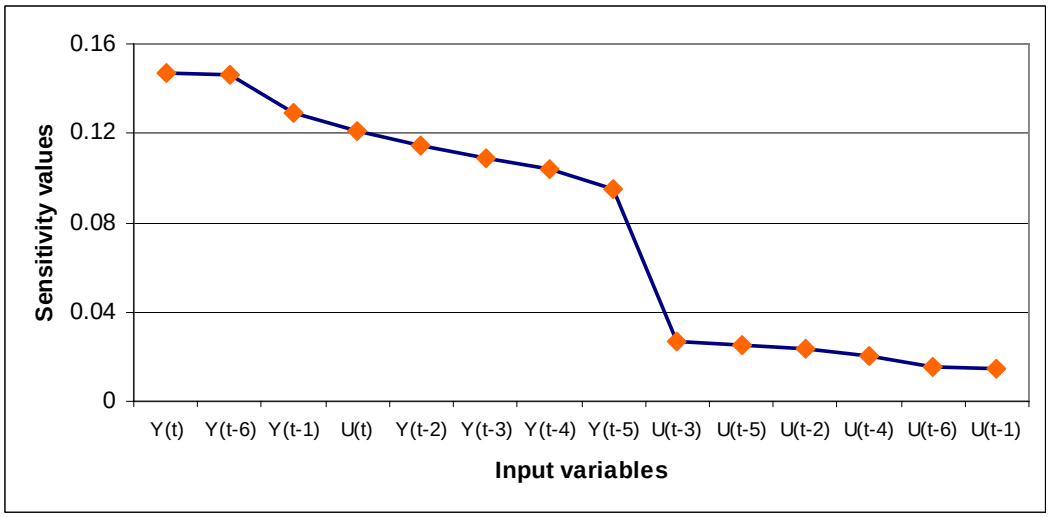

Figure 6.11: ANN model sensitivity for pH system

Table 6.3: Performances of ANN models

ANN Models	Training data set			Test data set		
	RMSE	***% Error***	***CC***	***RMSE***	***% Error***	***CC***
f_1	0.081	4.065	0.976	0.08	4.125	0.97
f_2	0.075	3.954	0.98	0.078	3.985	0.9758
f_3	0.068	3.59	0.984	0.066	3.45	0.985
f_4	1.054	5.21	0.94	1.65	6.25	0.926
f_5	0.098	5.036	0.956	1.032	5.963	0.954
f_6	0.0865	4.157	0.975	0.095	4.265	0.968

6.5.4 Conclusion

In this section, fuzzy curves and surfaces method has been explored for the identification of important inputs of a model that significantly affect the model output. The results of the FCS have been compared with another input identification method namely ANN-based sensitivity analysis. From the two case studies performed in this section it is seen that important inputs identified by both the methods improve the prediction and generalization performance of the models. It is also observed that the FCS has fared better than the sensitivity analysis method in identifying important inputs.

6.6 REFERENCES

1. Aldrich C. (1998), Visualization of Transferred Multivariate Data Sets with Auto Associative Neural Networks, *Pattern Recognition Letter*s, 749–764.
2. Dedem G. V. (1975), Moo-Young M. A, Model for Diauxic Growth, *Biotechnol. Bioeng.*, 17, 1301–1312.
3. Demartines, P. and J. Herault (1997), Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets, *IEEE Transactions on Neural Networks*, 8(1), 148–154.
4. Engelbrecht, A. P., Cloete, I., & Zurada, J. M. (1995), Determining the significance of input parameters using sensitivity analysis, From natural to artificial neural computation: *Proceedings of International Workshop on Artificial Neural Networks*. Malaga-Torremolinos, Spain: Springer, 382–388.
5. Fukunaga, K. (1991), *Introduction to Statistical Pattern Recognition*, Academic Press, London.
6. Kramer M. A. (1992), Auto-Associative Neural Networks, *Computers and Chemical Engineering*, 16(4), 313–328.
7. Lin, Y., G. Cunningham III, S. V. Coggeshall, (1996), Input variable identification - fuzzy curves and fuzzy surfaces, *Fuzzy Sets and Systems*, 82, 65–71.
8. Lin, Y., G. Cunningham III, S. V. Coggeshall, R. D. Jones, (1998), Nonlinear system input structure identification: two stage fuzzy curves and surfaces. *IEEE Transactions on Systems, Man, and Cybernetics*, Part A 28(5), 678–684.
9. Mao, J. and Jain, A. K, Artificial Neural Networks For Feature Extraction And Multivariate Data Projection, *IEEE Trans. Neural Networks*, vol. 6, 1995, 296–317.
10. Prechelt, L. (1994), Porben1 – A Set of Neural Network Benchmark Problems And Benchmarking Rule, Technical Report 21/94, University of Karlsruhe; Germany. ftp://ftp.ira.uka.de/pub/neuron/
11. Pyun, Y. R., Modak, J. M., Chang, Y. K., Lim, H. C. (1989), Optimization of Biphasic Growth of *Saccharomyces carlsbergensis* in Fed-Batch Culture, *Biotechnol. Bioeng.*, 33, 1–10.

12. Pyun, Y. R., Modak, J. M., Chang, Y. K., Lim, H. C. (1989), Optimization of Biphasic Growth of Saccharomyces carlsbergensis in Fed-Batch Culture, *Biotechnol. Bioeng.*, 33, 1–10.
13. Ramos, L. S., K. R. Beebe, W. P. Carey, E. Sanchez, B. C. Erickson, B. E. Wilson, L. E. Wangen and B. R. Kowalski (1986), Chemometrics, *Analytical Chemistry*, 58, 294R-315R.
14. Ridder, D. de. and Duen, R. P. W. (1997), Sammon's Mapping Using Neural Networks: A Comparison, *Pattern Recognition Letters*, 18, 1307–1316.
15. Roweis S. T. and Saul L. K. (2000), Nonlinear Dimensionality Reduction by Locally Linear Embedding , Science, 290, 2323–2326.
16. Rumelhart D., Hinton G., Williams R. (1986), Learning Representations by Backpropagating Errors, Nature, 323, 533–536.
17. Sammon, J. W. (1969), A Nonlinear Mapping Algorithm For Data Structure Analysis, *IEEE Trans. Comput.*, vol.18(5), 401–409.
18. Sammon, J. W. (1969), A nonlinear mapping algorithm for data structure analysis, *IEEE transactions Computers*, C-18, no. 5, 401–409.
19. Sarkar, D., Modak, J. M. (2003), Optimization of Fed-Batch Bioreactor using Genetic Algorithm, *Chem. Eng. Sci.*, 2283–2296.
20. Shepard, R. N. and J. D. (1965), Carroll. Parametric representation of nonlinear data structures,. in *International Symposium on Multivariate Analysis*: Academic Press, New York.
21. Shimizu H., Yasuoka K., Uchiyama K., Shioya, S. (1998), Bioprocess Fault Detection by Nonlinear Multivariate Analysis: Application of an Artificial Auto-Associative Neural Network and Wavelet Filter Bank, *Biotechnol. Prog.*, 79–87.
22. Sugeno, M., T. Yasukawa (1993), A fuzzy-logic-based approach to qualitative modeling, *IEEE Transactions on Fuzzy Systems*, 1(1), 7–31.
23. Sung A. H. (1998), Ranking importance of input parameters of neural networks. Expert Systems with Applications, 15, 405–411.
24. Takagi, H. and I. Hayashi (1991), NN-driven fuzzy reasoning, *Internat. J. Approx. Reason.*, 5, 191–212.
25. Takagi, T., Sugeno, M. (1985), Fuzzy identification of systems and its applications to modeling and control, *IEEE Trans. Syst. And Man Cybern.* 15 (1), 116–132.

26. Venkatasubramanian, V., Vaidyanathan, R. and Yamamoto, Y. (1990), Process fault detection and diagnosis using neural networks. *Comput. Chem. Engng.* 14, 699–712.
27. Vesanto J. (2000), *SOM Toolbox*, vs. 2, see also, http://www.cis.hut.fi/projects/somtoolbox.
28. Vora N., Tambe S. S., Kulkarni B. D. (1997), Counterpropagation Neural Networks for Fault Detection and Diagnosis, *Computers Chem. Engg.*, 21(2), 177–185.
29. Vora, N., Tambe, S., Kulkarni, B. (1997), Counterpropagation Neural Networks for Fault Detection and Diagnosis, *Computers Chem. Eng.*, 21(2), 177–185.
30. Wold, S., K. Esbensen, and P. Geladi. (1987), Principal component analysis, *Chemo. Intell. Lab. Syst.*, 2, 37–52.
31. Zurada, J. M., Malinowski, A., and Cloete, I. (1994), Sensitivity analysis for minimization of input data dimension for feed forward neural network, *Proceedings of IEEE International Symposium on Circuits and Systems*, London: IEEE Press, 6, 447–450.

CHAPTER 7

CONCLUSION

In this chapter, we present a summary of contributions made in this thesis and guidelines for deployment of the AI-based formalisms.

7.1 CONCLUSIONS

Mathematical process models are required for a variety of process engineering tasks such as steady-state and dynamic modeling, process control, fault detection diagnosis, classification, optimization, data reduction, low dimentional projection and process monitoring. Most chemical processes exhibit complex nonlinear behavior and thus development of phenomenological models, which require complete understanding of the underlying process phenomenology (kinetics, thermodynamics, heat and mass transfer mechanisms, etc.) becomes difficult. An attractive alternative in the form of AI based models has become available in the last two decades. A significant advantage of the AI based models is that they can be developed exclusively from the process data without needing any information about the process phenomenology. Likewise ANNs, a number of other AI-based formalisms have been proposed for conducting the process engineering tasks alluded to above. It may be noted that AI formalisms are generic in nature and can be employed in almost every science and engineering/technology discipline. The objective of this thesis therefore is to design, develop and apply various AI formalisms for the important tasks in chemical engineering/technology. Accordingly, the studies reported in the thesis and conclusions thereof are described below.

The first chapter provides a brief overview of the artificial intelligence (AI) and machine learning (ML) domains and highlights the major formalisms thereof as also their generic applications.

The second chapter provides algorithmic details of a number of AI and ML based formalisms used in the modeling, optimization, classification, dimensionality reduction and input selection case studies reported in the thesis.

Chapter 3 focusses on the applications and improvisation of AI and ML based formalisms for chemical/biochemical process modeling. In the first two studies, ANNs trained using an optimally noise-superimposed enlarged input-output data set has been shown to exhibit improved prediction and generalization

performance. In the third case study, ANN-based models have shown better accuracy for the prediction of gross calorific values of Indian coals over the conventionally used linear models. Next, the recently proposed ML-based formalism, namely SVR, has been utilized in a novel application involving softsensor development for biochemical processes. Also, the SVR has been shown to possess an excellent accuracy for the prediction of activity of a biochemical process. In the last study of the chapter, a modified genetic programming formalism integrating local and global search mechanisms has been explored for the modeling of the benzene isopropylation over Hbeta catalyst process.

Chapter 4 studies AI-based classification/clustering formalisms. Here, two case studies have been performed. In the first case study dealing with protein synthesis, the unsteady-state data from a fed-batch fermenter operating under faulty as well as normal operating conditions has been considered to successfully conduct fault classification. In the second case study involving citric acid production, the SOM neural network visualizes the dynamics of a multivariable process in the form of a two-dimensional map and brings out subtle differences between various batches.

In chapter 5, AI-based formalisms have been explored for process optimization. This chapter presents results of a memetic algorithm (MA) based multi-objective optimization of the zeolite (TS-1) catalyzed benzene hydroxylation to phenol process. Additionally, the optimal solutions obtained using the MA have been compared with the genetic algorithm based solutions obtained in an earlier study. The results of this comparison indicate that the MA has captured a better solution and that too in a shorter time when compared to the GA-based solution. It can thus be concluded that the global search augmented with a local search (as done by MA) fares better than a pure global search conducted by the GA.

Chapter 6 demonstrates applications and improvisation of AI and fuzzy logic based formalisms for data dimensionality reduction/low dimensional projection and input selection of chemical/biochemical process data. In the first three sections of this chapter, ANN-based Sammon's mapping, locally linear embedding and curvilinear component analysis have been used for the dimensionality reduction of the process data and fault detection and diagnosis

thereof. In the last section of the chapter, a recently proposed fuzzy logic based input selection formalism namely, fuzzy curves and surfaces, has been successfully utilized for the input selection of a pH neutralization process and heat exchanger control system.

7.2 GUIDELINES FOR DEPLOYMENT OF AI FOMALISMS

7.2.1 Thumb-rules for the Development and Deployment of ANN models

- Never "throw" non-analyzed, non-processed data at an ANN, i.e., preprocess the data before subjecting to the ANN for modeling; perform correlation and trend analysis. Also, remove outliers.
- Make sure that adequate data for training an ANN is available: data adequacy depends on the input-output dimensionality of the system to be modeled
- Employ "proper" data representation methods: input encoding, filtering, etc., should be performed
- Avoid over-training the neural network: use a test set, which is different from the training set for checking the generalization ability of the network. Due to this ability, the network model can generalize the information learned during the training phase to make accurate predictions for new inputs.
- Use different strategies for scaling the ANN inputs; for instance, simple normalization, mean centering, and z-score method for input scaling.
- Avoid trying to map multiple functions using a single neural network: develop a different network for each output.
- Develop parsimonious neural network models: That is use as few hidden neurons as possible. Generally, one hidden layer is sufficient in the ANN architecture although two hidden layers are sometimes needed.

Various types of ANNs are highly efficient in approximating nonlinear relationships existing between variables of two sets of data. These advantages have been presented in various case studies in Chpter 3. The principle drawback of ANNs however are as follows.

- Training of a majority of ANN architectures is an iterative procedure and therefore numerically intensive and time consuming.
- Being a black-box model, the parameters (weights) of the most commonly used neural network paradigm, namely *multilayer perceptron* (MLP) can not interpreted in terms of the training data.
- Irrespective of the system under investigation, the MLP network uses a generic nonlinear transfer function such as the logistic sigmoid for approximating nonlinear input-output relationships although the complexity of the fitted model varies depending upon the number of model inputs and outputs.
- There do not exist mathematically sound criteria for the number of patterns essential for training of an ANN model although a few guidelines do exist.

7.2.2 Guidelines for Using Other AI Formalisms

(A) Before exploring neural networks for classification tasks, conventional clustering algorithms such as *K-means* should be explored. In general, feed forward neural networks are suitable for supervised classification tasks whereas self-organizing neural network outperforms K-means technique for unsupervised classification tasks.

(B) Most AI based optimization techniques are iterative in nature and therefore numerically intensive. Thus, they are not ideally suited for online optimization unless optimization problem is of small size (fewer decision variables). However, the most significant benefit of AI based optimization methods is that invariably they find a global or near-global optimal solution. In addition, they are not constrained by the continuity, smoothness and differentiability of the objective function criteria required by the commonly utilized deterministic gradient based methods.

(C) In control applications, ANN should be used with at most care for following resons.

(i) ANNs are not good at predicting gains, (ii) multistep-ahead predictions made by an ANN are likely to be erroneous due to accumulation of approximation errors committed during recycling of single step-ahead predictions.

(D) Mathematically sound dimensionality reduction techniques such as principle component analysis are available for decades and extensively used. A major drawback of these methods is that they are linear in character. AI-based dimensionality reduction (DR) methods are efficient in capturing nonlinearities presented in the data. Since they are computationally more expensive than the PCA and its variants, the usage of AI-based DR techniques should be attempted only after establishing unsuitability of linear techniques.

APPENDIX A. LIST OF PUBLICATIONS

Research Papers Included in Thesis

1. "Soft-sensors for Improved Monitoring and Performance of Polyethylene Plants," Y. P. Badhe, J. Lonari, S. S. Tambe and B. D. Kulkarni, Neelamkumar Valecha, Sanjay V. Deshmukh, Bhavanishankar Shenoy and S. Ravichandran, *HP*, March 2007.
2. "Soft-sensor development for fed batch bioreactors using support vector regression," Kiran Desai, Yogesh P. Badhe, Sanjeev S. Tambe and Bhaskar D. Kulkarni, *Biochemical Engineering Journal*, 27, 2006, 225–239.
3. "Estimation of Gross Calorific Value of Coals using Artificial Neural Networks," Shagufta U. Patel, B. Jeevan Kumar, Yogesh P. Badhe, B. K. Sharma, Sujan Saha, Subhasish Biswas, Asim Chaudhury , Sanjeev S. Tambe and Bhaskar D. Kulkarni, *Fuel*, 86 (3), Feb-2007, 334–344.
4. "Support vector regression for bioprocess identification," Yogesh Badhe, Jitender Jit Singh Cheema, Manoj Potdar, Sanjeev S. Tambe and B. D. Kulkarni, BIOHORIZON, IIT New Delhi, 2003.
5. "Monitoring and Fault Detection/Diagnosis of a Batch Fermentation Process Using Self-organizing Map Neural Networks," Yogesh Badhe, Vinay Wadekar, Kiran Desai, S. S. Tambe and B. D. Kulkarni, Poster Presentation, NCL Day, Feb 28, 2004.
6. "Genetic Programming for Data-Driven Modeling of Non-Linear Chemical Processes" Phulwale U.S., Badhe Y.P., Mandge D.P., Tambe S.S., Kulkarni B.D., Poster Presentation, NCL Day, 2005.
7. "Nonlinear feature extraction using Sammon's mapping and SAMANN," Yogesh Badhe, Vinay Wadekar, Uttam Phulwale, S. S. Tambe and B. D. Kulkarni, Proceedings of "Conference of Research Scholars and Young Scientists (CRSYS)," held at IIT, Kharagpur on Sept. 25-26, 2004, 21-27.
8. "Monitoring and fault detection of biochemical systems using locally linear embedding," Uttam S. Phulwale, Kiran M. Desai ,Yogesh P. Badhe, Vinay A. Wadekar, Sanjeev S. Tambe, Bhaskar D. Kulkarni, Proceedings

of Conference of Research Scholars and Young Scientists (CRSYS)," held at IIT, Kharagpur on Sept. 25-26, 2004, 78-84.

9. "Process optimization using memetic algorithms: A case study of benzene hydroxylation to phenol process," Yogesh Badhe, Kiran Desai, Vinay Wadekar, Uttam Phulwale, S. S. Tambe and B. D. Kulkarni, Presented in Indian Chemical Engineering Congress (CHEMCON-2004), organised by Indian Institute of Chemical Engineers, held at The Grand Hyatt, Mumbai, during 28-30 Dec. 2004.
10. "Performance Enhancement of Artificial Neural Network Based Models in Presence of Noisy Data," Yogesh P. Badhe, Sanjeev S. Tambe and Bhaskar D. Kulkarni, Presented in the "First Indo-US Joint Meeting in a Global Environment," organized by Indian Institute of Chemical Engineers and American Institute of Chemical Engineers held at The Grand Hyatt, Mumbai, during 28-30 Dec. 2004.
11. "Process Monitoring and Fault Detection Using Curvilinear Component Analysis," U. S. Phulwale, B. Jeevan kumar, S. U. Patel, Y. P.Badhe, S. S. Tambe and B. D. Kulkarni. Presented in the "Second Indian International Conference on Artificial Intelligence", held on Dec 20-22, 2005, at the National Insurance Academy, Pune, India.

Other Research Papers

1. "Hybrid process modeling and optimization strategies interacting neural networks/support vector regression and genetic algorithms: study of benzene isopropylation on Hbeta catalyst," S. Nandi, Y. P. Badhe, Jayaram Lonari, U.Shridevi, B. S. Rao, S. S. Tambe, B. D. Kulkarni, Chemical Engineering Journal, 97, 2004, 115–129.
2. "Enhanced exopolysaccharide production from Lactobacillus plantarum by optimization of media using artificial intelligence techniques," K. M. Desai, S. K. Akolkar, Y. P. Badhe, S. S. Tambe, S. S. Lele, B. D. Kulkarni, Process Biochemistry, 41, 2006, 1842–1848.
3. "Self-Organizing Maps: A tool to explore relatedness of organisms based on 16S rDNA sequence features," D. V. Raje, H. J. Purohit, Y. P. Badhe,

V. A. Wadekar, S. S. Tambe, B. D. Kulkarni, communicated to Journal of Molecular Biology and Evolution.

4. "Density measurements of coal samples by different probe gases and their interrelation," Sujan Saha, B. K. Sharma1, Y. P. Badhe, S. S. Tambe, B. D. Kulkarni, Fuel, 86 (10-11), 2007, 1594–1600.
5. "Predicting yield of Indian cotton using multivariate statistical and artificial intelligence based techniques," Kiran Desai, Yogesh badhe, Sanjeev S. Tambe, D. V. Dev, and Bhaskar D. Kulkarni, communicated to Electronics and Computers in Agriculture.
6. "Artificial intelligence formalisms for process modeling and optimization," Y. Badhe, and S. S. Tambe, Invited lecture presented (Proceedings 11-18), at the "International Conference on Instrumentation (INCON 2004)," held at Pune Institute of Engineering and Technology (PIET) during December 19-21, 2004.
7. "Feature extraction of gene expression data using curvilinear component analysis (CCA) and self-organizing map (SOM)" Vinay A. Wadekar, Yogesh P. Badhe, Uttam S. Phulwale, S. S. Tambe, and B. D. Kulkarni. Presented in "Fourth Indo-US Workshop on Mathematical Chemistry," Organized jointly by Natural Resources Research Institute, Univ. of Minnesota Duluth (USA) and Univ. Of Pune, Pune and held at University of Pune, Pune during January 8-12, 2005, Paper No. O16.
8. "Monitoring and fault detection of a batch fermentation process using self-organizing map (SOM) neural networks," S. U. Patel, B. Jeevan Kumar, U. S. Phulwale, Y. P. Badhe, S. S. Tambe and B. D. Kulkarni, Presented in the Indian Chemical Engineering Congress (CHEMCON-2005), organised by the Indian Institute of Chemical Engineers, held at Indian Institute of Technology (IIT), New Delhi, during 14-17 Dec. 2005.
9. "Multi-model scheme for prediction of monthly rainfall over India," J. R. Kulkarni, Savita G. Kulkarni, Y. Badhe, S. S. Tambe, B. D. Kulkarni and G. B. Pant, Research Report No. RR-101, ISSN 0252-1075, 1-28, Indian Institute of Tropical Meteorology, Pune 411 008, India (December 2003).
10. "Artificial Intelligence formalism for process modeling and optimization," Y. P. Badhe, S. S. Tambe, International Conference on Instrumentation, Pune (Dec. 2004).