To the Graduate Council:

I am submitting herewith a thesis written by George Mathai entitled "AUTOMATED METHODS FOR SIGNAL VALIDATION AND ANOMALY DETECTION." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Donald W. Bouldin

D. W. Bouldin, Major Professor

We have read this thesis
and recommend its acceptance:

BR Upadhyaya

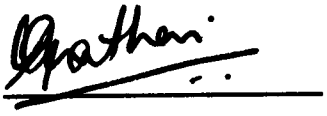
Accepted for the Council:

Vice Provost
and Dean of The Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature 

Date 04/26/89

AUTOMATED METHODS FOR SIGNAL VALIDATION AND ANOMALY DETECTION

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

George Mathai

May 1989

Dedication

This thesis is dedicated to my parents, Saramma and John Mathai, to all my teachers, past and present, and to my fiancé and best friend, Lisa. To each I owe so much.

Acknowledgments

I am grateful to my committee, Drs. D. W. Bouldin, B. R. Upadhyaya and J. M. Bailey, for their help and support. Special thanks go to Dr. B. R. Upadhyaya who was constantly available to advise on the research and writing of this thesis.

I also wish to thank the members of the N.E. Coffee Club for their help and plentiful supply of coffee. In particular, I wish to mention Evren Eryurek, Lamartine Guimaraes, John Berkan, Keith Holbert, Tony Hurst and Zsolt Frei, who have helped me along the way.

I gratefully acknowledge the Measurement and Control Engineering Center, University of Tennessee, for sponsoring this research.

Abstract

Instrument fault detection and process diagnostics are necessary to verify the validity of process variables channeled to control systems, protection systems and plant monitoring systems. Three techniques are applied for validating sensor signals and classification of spectral signature functions. The characterization of a data cluster representing a certain process behavior is achieved by steady-state nonlinear modeling of one or more critical signals as a function of one or more process variables in the system. This prediction model is used to detect sensor maloperation or process anomaly by comparing the measurement and prediction of the same variable. Bidirectional associative memory is applied to classification of frequency domain spectral sigantures. Its noise tolerant properties make it suitable for classification of power spectral density (PSD) functions which are difficult to classify due to wide variations possible in spectra derived from the same process. Backpropagation Networks (BPN) are applied to predictive modeling and classification of PSD functions. The BPN allows arbitrary nonlinear mapping of data to data and performs well as a noise tolerant classifier and as a predictive model.

TABLE OF CONTENTS

CHAPTER PAGE

CHAPTER PAGE

LIST OF TABLES

LIST OF FIGURES

FIGURE PAGE

FIGURE PAGE

FIGURE PAGE

CHAPTER 1

Introduction

1.1 Purpose of the Present Study

In all process control systems, signals from several instrument channels are used in control systems, protection systems, improving product quality, enhancing the reliability of operator decision, as an aid for scheduling maintenance, and minimizing plant downtime. Validating these instrument channels on-line and off-line are of importance in ensuring normal plant operation. The volume of data and the proliferation of signals make operator verification of signals difficult. Also nonredundant signals require validation using isolated channels of verification to eliminate crosstalk between the measurement and verification paths. This is necessary to avoid the propagation of fault states to the validation scheme. Large processes with multiple states of operation and fluctuating conditions are difficult to characterize and validate, requiring a systematic breakup of process states and independent handling of each state.

Frequency-domain spectral signatures are often used to analyze random signals from processes and the physical interpretation of these signatures require human interface. Power Spectral Density (PSD) functions used in this analysis

are generated using Fast Fourier Transforms (FFT) and autoregressive time series modeling of short data records [1]. The inherent nature of these signatures, limitations in sampling rate and approximations in data acquisition result in a wide variety (though essentially similar) of variations in spectra derived from the same process state. Thus automated classification and recognition of these functions present a duanting task. Supervised analysis of baseline spectra reveal the various states in the process and their physical interpretations. A suitable classifier can thus provide automated fault detection and isolation and be capable of sifting through vast amounts of data without operator assistance.

1.2 Overview of the Methodology

The work presented in this document addresses the two problems namely, instrument output validation and classification of spectral signatures. Pattern recognition methods are used for the clustering of the baseline data into several process states. Independent characterization of each process state with a prediction model greatly enhances the prediction accuracy as compared to representing the entire baseline data set with one prediction model. Nonlinear empirical models are generated using the measured data. It is useful in processes which are too complex or have unknown physics for characterization by physical models. The instrument signal to be validated is expressed as a function of other process signals. From a signal validation viewpoint, empirical modeling has the

significant advantage of not requiring redundancy in sensors. Clustering of the baseline data into different process states is done using the *K-Means* algorithm and the *single linkage cluster analysis* algorithm. The *Signal Validation Module* (SVM) is a software package designed to validate instrument outputs. It has a database containing the various clusters and their corresponding models. The SVM is capable of validating instrument outputs by classifying it to the closest cluster and applying the corresponding prediction model. The SVM is a final consolidated form of the clustering and empirical modeling.

The *Bidirectional Associative Memory* (BAM) [2] is applied to the classification of spectral signatures. The BAM is a two-layer, fully-connected neural network that utilizes Hebbian learning. The noise tolerant property of the BAM makes it suitable for spectral signature classification. Such an automated classification is useful in machine diagnosis of fault states. The BAM also features a characteristic *energy of association* between the trained domain and test patterns, which quantifies the closeness of the test pattern to the trained domain.

The *Backpropagation Network* (BPN) [3] is widely regarded as the workhorse among the various neural network paradigms. Its ability to learn arbitrary nonlinear mappings and its fault tolerance make it suitable for a wide variety of tasks. It is inherently fault tolerant owing its massive parallelism. The BPN is applied, both to signal validation and as a classifier for spectral signatures. Pattern coding schemes, network structuring, and the control parameters are discussed for both the applications.

The BAM has been applied to image recognition and resource allocation [2]. The primary development with the BAM has been towards its optical implementation. The BPN has seen a wide variety of applications in different areas. BPN has been used for acoustic signal classification [4], phenome recognition [5], and image processing [6] among other applications. The ISODATA clustering alogrithm has been used in classifying time-dependent noise power spectra derived from nuclear reactors [7]. Multivariate statistical pattern recognition methods have also been applied to reactor noise analysis [8].

1.3 Contributions of the Thesis

Empirical prediction models, data clustering, and Neural Networks are applied for signal validation and anomaly classification. Two techniques, each for signal validation and spectral signature classification, are implemented and their performance analyzed. The SVM is developed as a general method to validate nonredundant signals where an operational baseline data set is available. A scheme for representing large data sets with empirical prediction models is presented. The BAM is applied as a classifier for spectral signatures. A pattern coding scheme is developed to enhance the discriminating features of the various classes and produce binary pattern vectors compatible with the BAM. The BPN is applied to signal validation and signature classification. The scaling and parametric studies are presented for the BPN application. A method to expedite

the time period required to train the network is developed.

1.4 Organization of the Thesis

Chapter 2 contains empirical prediction models and the clustering algorithms. Empirical prediction models are shown with different approaches to clustering the data. Chapter 3 discusses the SVM, its data base, and the results from its application to test data. The BAM and its application to spectral signature classification is discussed in chapter 4. Chapter 5 pertains to the BPN and its application to signal validation and spectral signature classification. Summary and conclusions are given in chapter 6.

CHAPTER 2

Empirical Prediction Modeling and Data Clustering

2.1 Introduction

Changes in process signal behavior are either due to instrument degradation or due to process anomaly (actuators, controllers, etc.). The main objective is to develop an automated system to monitor sensor systems and process dynamics. Empirical models are constructed using only the measured data. These models are used to compute estimated values of the sensor outputs. The actual sensor reading and the estimated value of that sensor are arrived at through independent paths, so an anomaly in the actual sensor output will not influence the estimated value once the empirical model has been constructed. It is important to note that sudden excursions of the sensor signal from normal behavior followed by its recovery in a short period of time may not reflect significantly on the mean value of the sensor output and hence may not be detected. Changes in system behavior can also be detected as gradual variations from expected values.

The accuracy of fitting an empirical model would be improved if the functional form of the fit is not highly nonlinear. Pattern recognition methods are

used to cluster the data, so that individual models may be applied to each cluster. Figures 2.1 - 2.4 show the measured mean coil data for the four inter-stand tensions. The horizontal lines indicate the one-standard deviation and the two-standard deviation lines on either side of the mean value of the signal.

2.2 Nonlinear Empirical Modeling for Process State Prediction

The general algorithm [9] [10] facilitates the construction of a model using only the measurements available in the system. The model consists of cross product terms of input variables and a constant term. The general form of the model is given by,

$$y = C_0 + \sum_{i=1}^{N} C_i \phi_i(\mathbf{x}) \tag{2.1}$$

The order of nonlinearity can be specified, which sets an upper limit for the sum of power in each cross-product term. Consequently, there is a proliferation of the number of cross-product terms with even a small increase in the number of input signals considered. This is significant from a computational viewpoint, since a large number of cross-product terms would result in a large matrix, whose inversion might lead to computational instability (for example, if the number of input variables is 4 and the order of the model is 3, this results in a 34×34 matrix). Figure 2.5 shows the structuring of the modeling methodology.

$X_1, \ldots, X_n$ are the input signals used to predict $\boldsymbol{Y}$.

Multiple measurements of the same signals are used to build an Euclidean

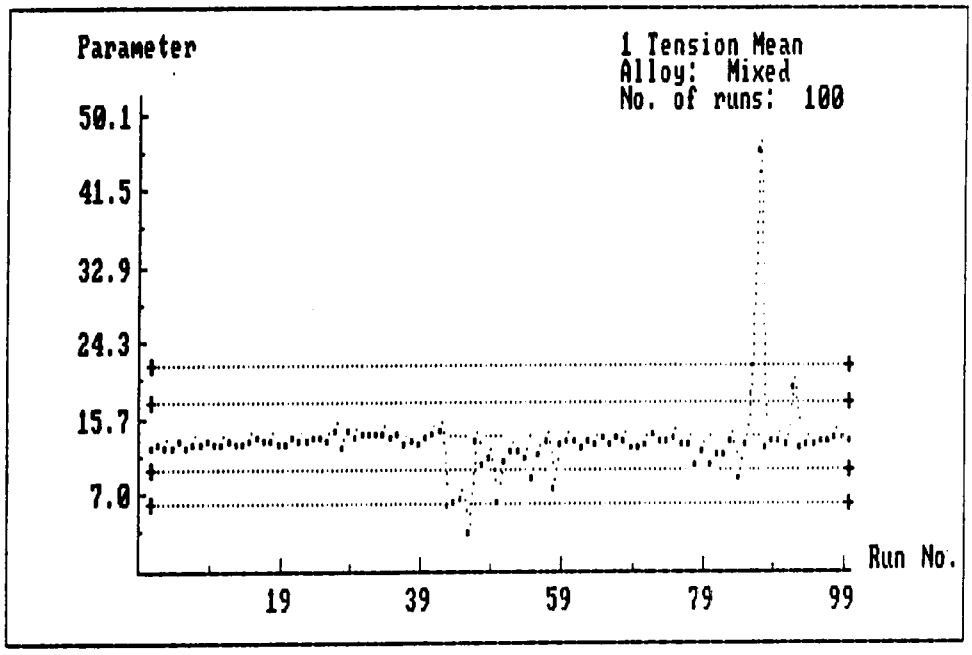


Figure 2.1: Measured values of interstand 1-2 tension (mean coil data) (* 100 PSI).

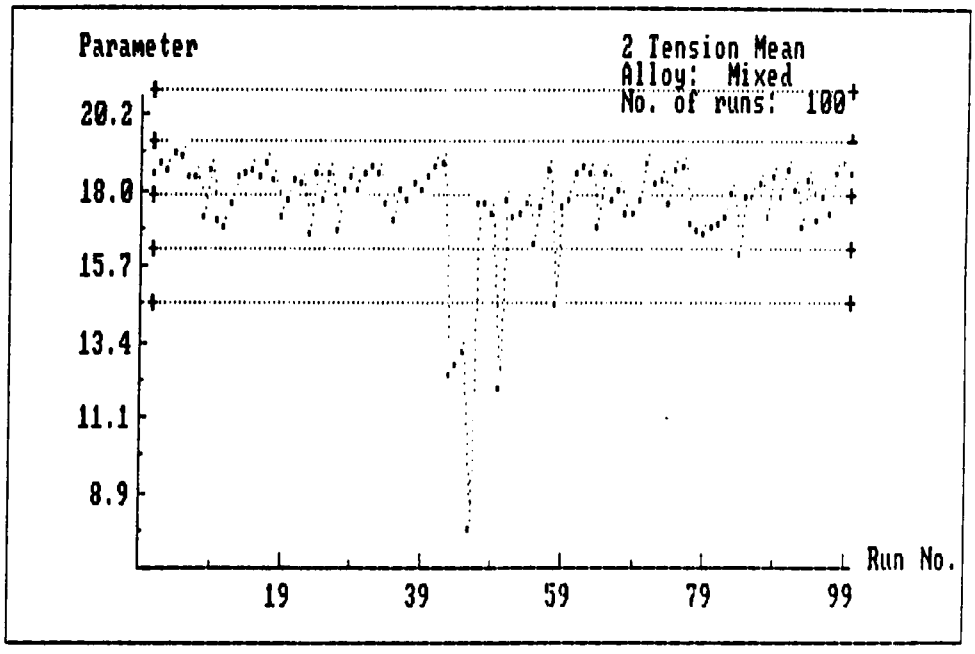


Figure 2.2: Measured values of interstand 2-3 tension (mean coil data) (* 100 PSI).

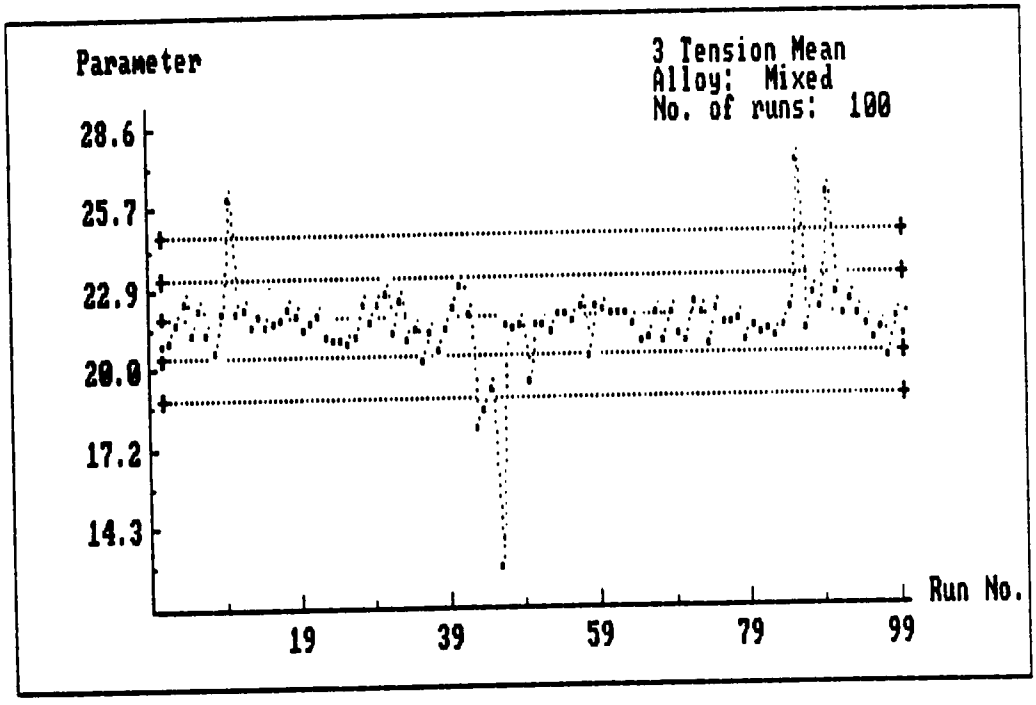


Figure 2.3: Measured values of interstand 3-4 tension (mean coil data) (* 100 PSI).

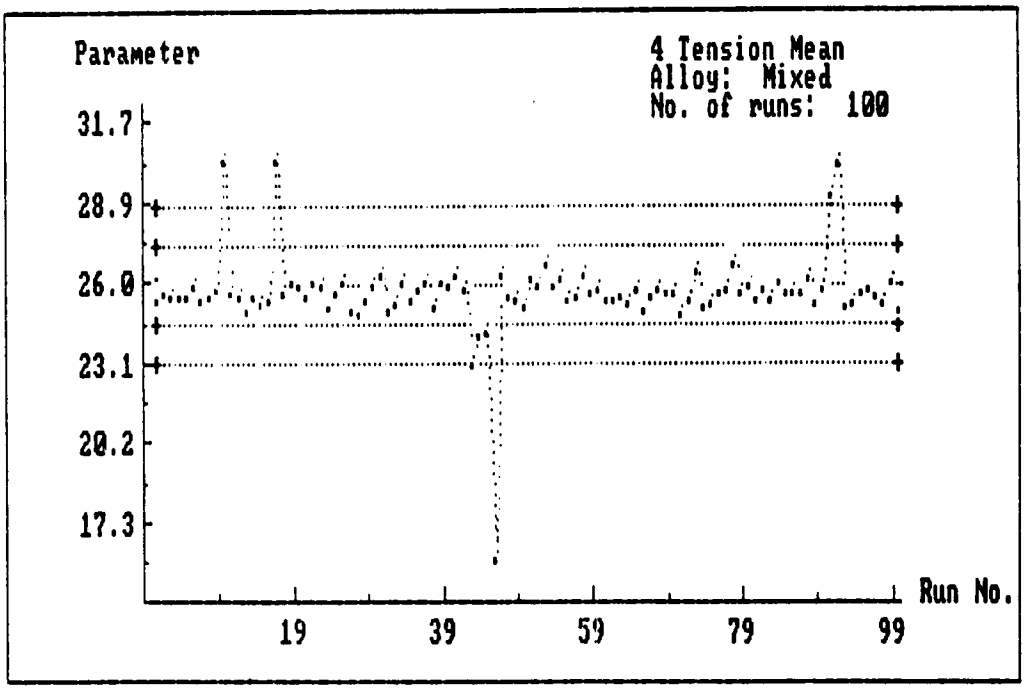


Figure 2.4: Measured values of interstand 4-5 tension (mean coil data) (* 100 PSI).

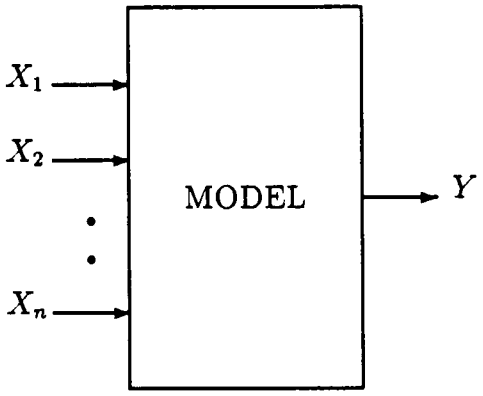


Figure 2.5: Block diagram showing the structuring of the empirical model.

vector space. Let L be the number of measurement vectors, and let M be the number of possible cross-product terms. Using each measurement vector, M vectors of dimension L can be calculated as,

$$\mathbf{v}(i) = \phi_i(x(k)) \tag{2.2}$$

where $i = 1, \ldots, M$ and $k = 1, \ldots, L$. ϕ_i corresponds to the i^{th} cross-product, and $x(k)$ is an input vector of length n, where n is the number of input variables. Thus an L-dimensional Euclidean vector space is constructed with M vectors in it. If there are at least L linearly independent vectors among the M calculated, the vector space can be determined correctly. Also, in this vector space the output values can be represented as an L-dimensional vector (assuming one output variable). Some of the vectors are then chosen from the basis vectors in the vector space and the system is modeled using these vectors only. The goal is to select N vectors, where $N \leq M$, in an optimal fashion (from a least-squares point of view). The algorithm is described below.

- Determine the closest (Euclidean distance) vector to the output vector. The term (cross-product) corresponding to the selected vector will be the one term model of the system (the coefficients are determined later).

- If additional terms are needed, the algorithm projects all the remaining vectors into a subspace orthogonal to the subspace selected. The program also projects the output vector into this subspace. This is due to the fact that the component of the output vector parallel to the selected vector

has already been represented. Now again a search is done for the closest vector in this subspace, in order to represent the remaining components of the output vector. This vector is selected as the second term of the model. This process goes on until the required number of terms are selected.

- Once the desired number of terms are selected, the final step is to fit a linear type model using these terms by computing the coefficients of each term and a constant term according to Eq. (2.1).

The above steps result in an N-term model. During the procedure described above, the error after the generation of each term (the square of the scalar length of the remaining part of the output vector), can be monitored and the process controlled according to the desired error level. Once the desired error level is reached, the coefficients $(C_0, C_1, \ldots, C_N)$ are determined.

2.3 Clustering Methodology

In order to obtain different clusters from the data, some pattern recognition methods are applied to the data. The break up of the data into different clusters, each cluster containing members that lie close (the measure of closeness is the Euclidean distance) to each other in N-dimensional space, allows us to compute a separate model for each cluster. These models are more accurate for their respective clusters than for the whole data set. The discussion of the clustering algorithms used and related concepts are given in this section.

2.3.1 Choice of Pattern Type

Before any of the clustering concepts can be applied, it is necessary to form *patterns* from the data set. A pattern is a position vector of dimensionality N which defines a unique point in a N-dimensional Euclidean space. When the classification is made using the mean values of the parameters, these values can be used directly to form patterns. The dimension of the sample pattern vectors depends on the number of signals considered. It is important to choose correctly the parameters that constitute a pattern. In this case these parameters are all the input and output signals considered for the nonlinear model (for example, if there are four input signals used to estimate one output signal, then the five signals are combined to form a 5-dimensional pattern).

2.3.2 The General Clustering Algorithms

In order to study the clustering properties of the data two cluster seeking algorithms are used. Samples from the same class should exhibit characteristic clustering in N-dimensional space, and these clusters should ideally be well demarcated from the samples of another class. In case of multidimensional data (greater than 3) it is difficult to visualize the geometrical properties of the data. Two clustering algorithms are applied to the data [11] [12] [13].

1. K-Means algorithm.

2. Single Linkage Cluster Analysis algorithm.

In order to facilitate proper interpretation of the results from the clustering and to allow a partial insight into the geometrical properties of the data, the distances between all the cluster centers and the variances of the cluster domains about the mean values are evaluated. The use of these clustering algorithms requires considerable experimentation with the number of clusters generated and with different values of thresholds. This approach is partly heuristic and is greatly dependent on the data.

K-Means Algorithm

This algorithm is based on the minimization of a performance index, which is the sum of squared distances from all the points in a cluster to the cluster center. The algorithm [11] is summarized.

1. Choose arbitrarily K cluster centers, $Z_1(1), Z_2(1), \ldots, Z_k(1)$, as any K number of samples from the sample set. In this notation the subscript indicates the cluster and the number in parentheses indicates the iteration number. The cluster centers are initially chosen as the first K samples where K is the number of clusters desired.

2. Compute distances from each sample to all the cluster centers, and assign the sample to the cluster whose cluster center is the closest to the sample. At the k^{th} iteration the samples are distributed among the K cluster

domains, according to the relationship

$$\mathbf{x} \in S_j(k) \quad if \quad \| \mathbf{x} - \mathbf{Z}_j(k) \| < \| \mathbf{x} - \mathbf{Z}_i(k) \| \tag{2.3}$$

for all $i = 1, 2, \ldots, k$, $i \neq j$, where $S_j(k)$ represents the samples whose cluster center is $Z_j(k)$. All ties are resolved arbitrarily. Each cluster now has a membership of samples associated with it.

3. The new cluster center, $Z_j(k+1)$, $j = 1, 2, \ldots, K$, for each cluster is updated as the centroid of the membership of the cluster at that iteration such that the sum of the squared distances of all points in $S_j(k)$ to the new cluster center is minimized. The performance index for that is,

$$J_j = \sum_{x \in S_j(k)} \| \mathbf{x} - \mathbf{Z}_j(k+1) \|^2, \quad j = 1, 2, \ldots, K \tag{2.4}$$

The new cluster centers are computed using the relation,

$$\mathbf{Z}_j(k+1) = \frac{1}{N_j} \sum_{\mathbf{x} \in S_j(k)} \mathbf{x}, \qquad j = 1, 2, \ldots, K \tag{2.5}$$

where N_j is the number of samples in the j^{th} cluster at the k^{th} iteration.

4. If the cluster centers generated at the $(k+1)^{th}$ iteration are the same as those at the k^{th} iteration then terminate, else go to step 2. That is, if $\mathbf{Z}_j(k+1) = \mathbf{Z}_j(k)$, for $j = 1, 2, \ldots, K$, then terminate.

The result of the algorithm is influenced by the choice of initial cluster centers and the ordering of the samples. This is because a different ordering would result in the initial cluster centers being different, which causes the membership of the clusters to change. So the algorithm may arrive at a different termination point.

Single Linkage Cluster Analysis Algorithm

This algorithm [12] [13] is also a cluster seeking algorithm, which generates a specified number (from apriori knowledge) of clusters. The first step towards implementation of this algorithm is to generate a *Minimum Spanning Tree* which links all the points in the sample set with a minimum number of edges. The algorithm is summarized below.

1. Compute all the distances of each and every point from all the other points.

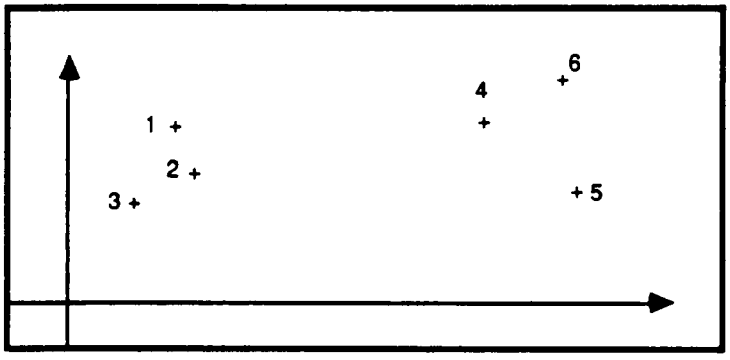
$$d(x_i, x_j), \qquad \forall i, j,\ i \neq j$$

2. These nodal distances are arranged in ascending order of magnitude.

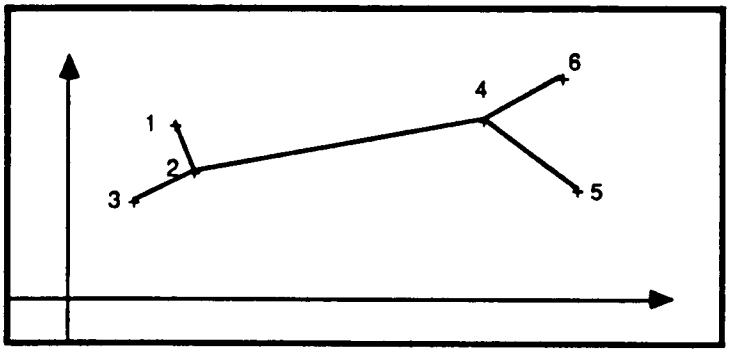
3. Starting from the first node pair edges are created between the nodes in the list. This process is continued sequentially, unless the creation of a particular edge causes the formation of a closed loop ; in which case the node pair is rejected. This process results in a minimum spanning tree.

4. If N cluster are to be formed, then $N-1$ edges are broken from the bottom of the list. Thus N cluster are formed with each cluster having a known membership.

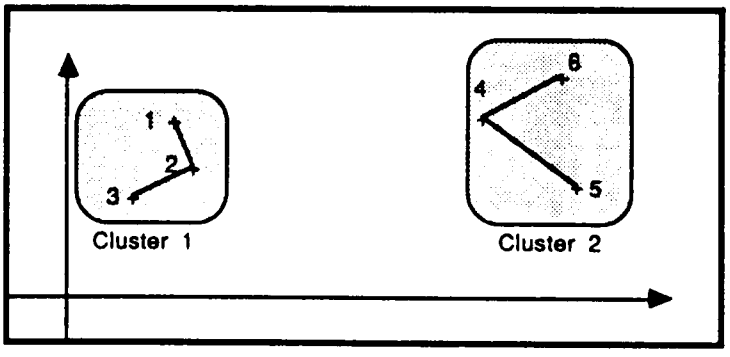
To illustrate the algorithm, consider a simple example of a two dimensional data set (Fig 2.6) containing six sample points. Figure 2.6a, shows the distribution of the sample points in two dimensional space. The total number of edges that can be created are $C_2^6 = 15$. The 15 distances are computed and arranged



(a) Sample Points.



(b) Minimum Spanning Tree.



(c) Formation of two clusters after breaking the largest edge (edge 2-4).

Figure 2.6: An example showing the principle of the Single Linkage Cluster Analysis algorithm.

in ascending order. Next the edges are created sequentially between the node pairs as follows:

(1 - 2)

(2 - 3)

(4 - 6)

(1 - 3)

But an attempt to create the edge (1 - 3) causes the formation of a closed loop, and hence it is rejected. This process continues until all 15 edges are covered and it results in a MST (Fig. 2.6b). If two clusters are to be formed ,$2 - 1 = 1$, edges from the bottom of the list are broken, which in this case is edge (2 - 4). The two clusters formed are shown in Fig. 2.6c.

2.4 Discussion of Results

The development of empirical models of any process requires the knowledge of the physics of a system in order to identify all the important signals which have a significant contribution to the variable to be modeled. The automated algorithm allows experimenting with different combinations of input variables to arrive at an optimum model. The computer program fits optimal polynomial functions to the measured mean value data. The optimal models are then used for predicting different variables of a process control system.

An important factor to be considered during the construction of the model is

that all the driving data must be validated as "normal". Otherwise the algorithm would try to force a fit to "abnormal" signals. In the latter case, the model may incorrectly identify a faulty signal as normal.

2.4.1 ALCOA Mean Coil Data

The results presented in this section illustrate the application of the data-driven empirical modeling to operational data from the ALCOA Hot Line Facility. Figure 2.7 shows a schematic of a rolling mill stand. There are 66 instrument readings available for each run characterizing the complete set of process parameters during the run. Also included in the data set are the standard deviation of each signal from the mean value. Each signal is an average over a time period of 3-4 minutes. The important signals considered are

- Interstand tension, stands 1 through 5,
- Roll force, stands 1 through 5,
- Roll speed, stands 1 through 5,
- Screw position, stands 1 through 5.

All the signals show a definite relationship with the interstand tensions. Scatter plots are used to study these relationships. Figures 2.8 - 2.11 show different signals plotted against interstand 4-5 tension. The characteristic clustering of the points signify a relationship between the signals and interstand 4-5 tension.

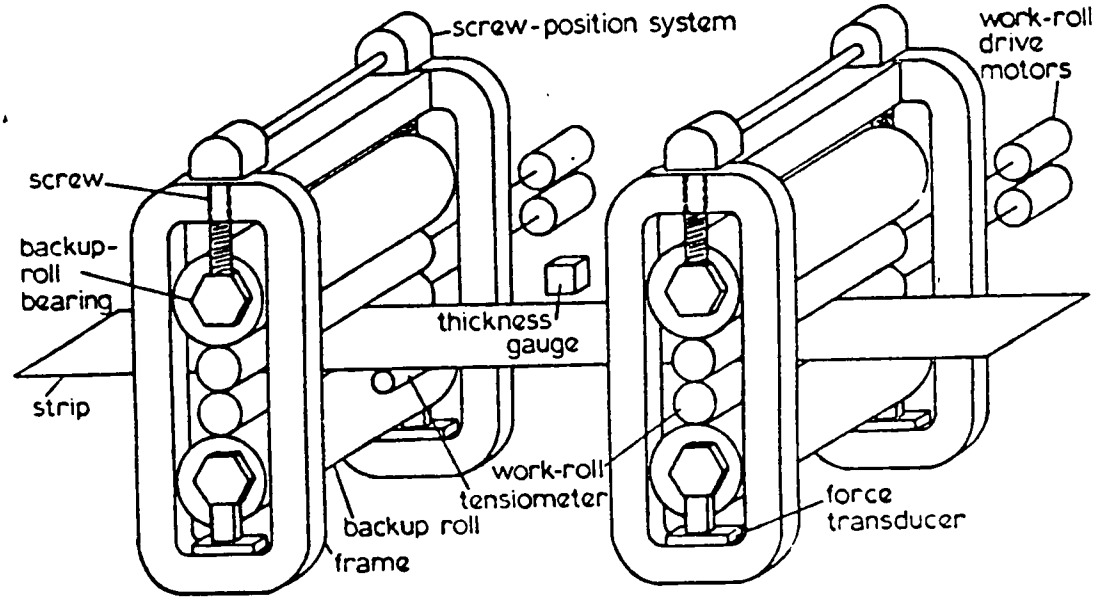


Figure 2.7: A schematic of a typical rolling-mill showing two stands.

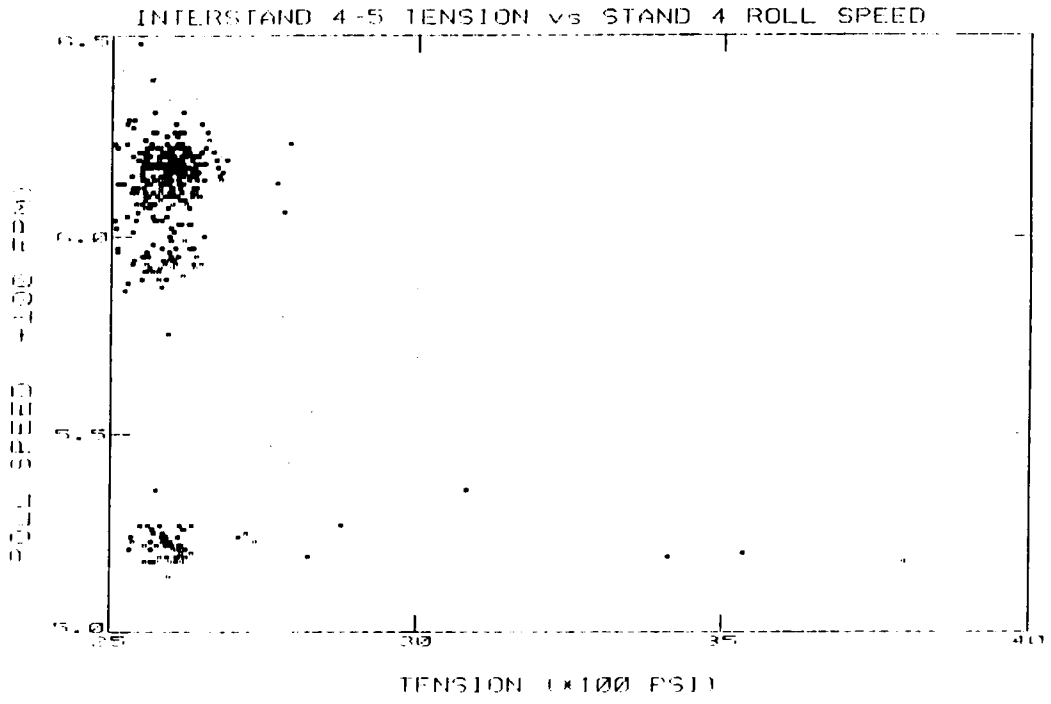


Figure 2.8: Interstand 4-5 tension plotted against stand-4 roll speed.

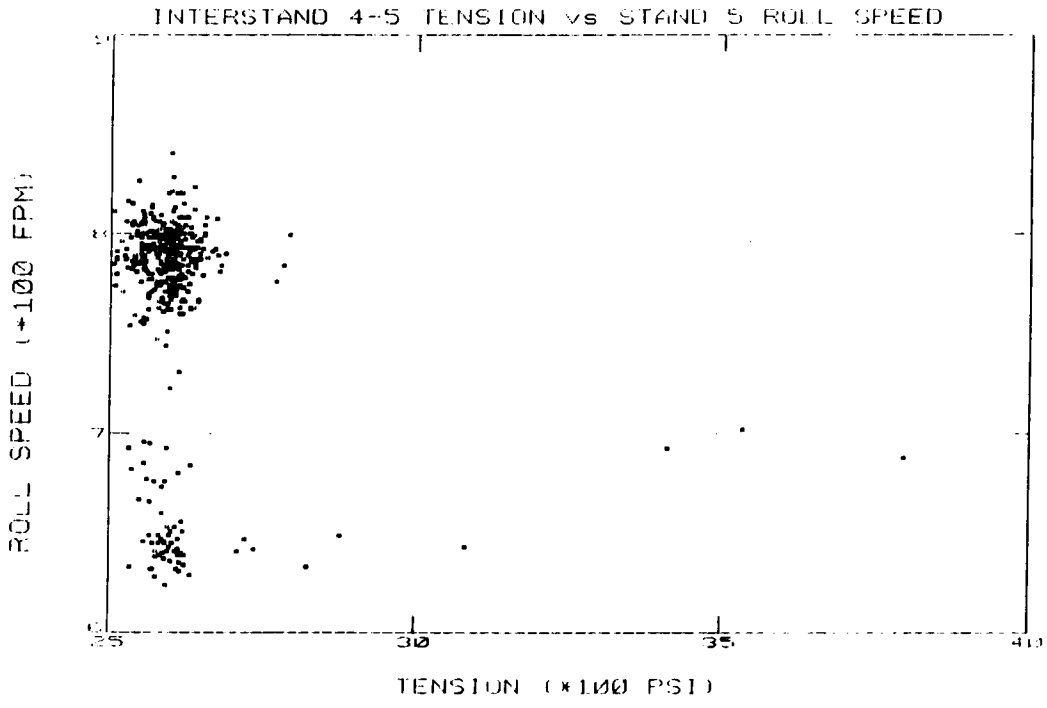


Figure 2.9: Interstand 4-5 tension plotted against stand-5 roll speed.

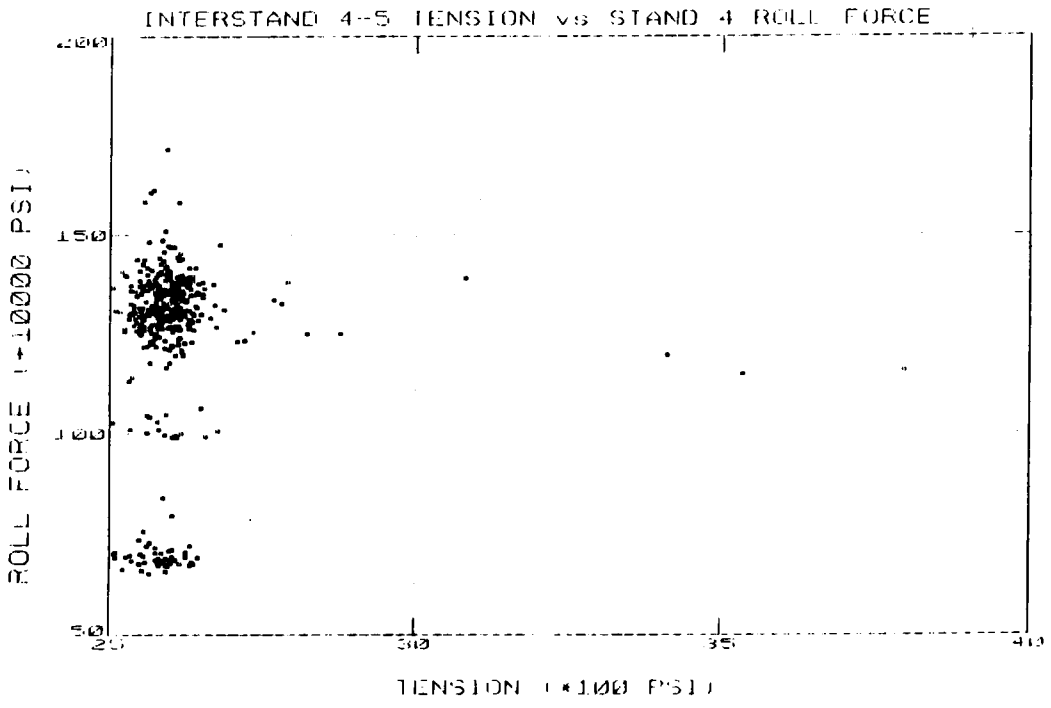


Figure 2.10: Interstand 4-5 tension plotted against stand-4 roll force.

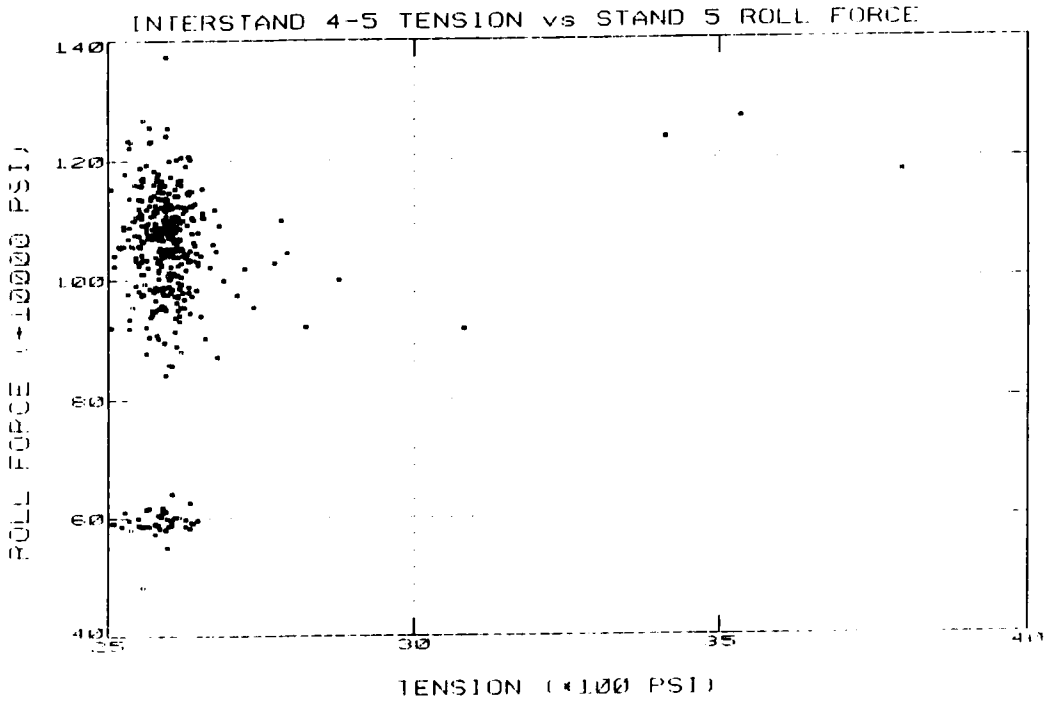


Figure 2.11: Interstand 4-5 tension plotted against stand-5 roll force.

2.4.2 Scaling of the Data

The input signals vary from each other in their magnitudes, and upon computation of higher order cross-product terms this variation increases. The algorithm requires the inversion of a matrix containing the cross-product terms, and large differences in magnitudes between the components of the matrix cause computational problems while inverting the matrix. The algorithm is implemented in a PC and differences in magnitude between terms exceeding 10^5, are found to cause mathematical overflow errors or lead to instability. To avoid this problem, all signals were scaled up or down so that the maximum difference between them did not exceed 10^3. The following scaling factors are used.

- All tensions scaled down by a factor of 100.
- All roll speeds scaled down by a factor of 100.
- All roll forces scaled down by a factor of 10,000.
- All screw positions scaled up by a factor of 10,000.

Once the model is constructed, the signals are scaled back to obtain the actual value of the variable estimate.

2.4.3 Clustering of Data

The accuracy of the model is enhanced greatly if the driving data used for the construction of the model are similar and tend to cluster together. So it is

required to break up the data set into different clusters and to fit a model to each cluster. This division of data was made using two approaches:

1. Manual clustering.

2. Automated clustering.

The data set contained 5 different alloy-types. Intuitively this suggests a division of data based on alloy-type. The various alloys-types are `3004`, `CS82`, `C817`, `5806`, and `SP30`. The results shown here, using manual clustering, are for alloy-type `3004`. The individual models for interstand tensions 1, 2, 3, and 4 were computed. Figures 2.12 - 2.15 show the empirical models developed for different tension variables and Figures 2.16 - 2.19 show the results from the application of empirical models developed for the interstand tensions.

The important factors to be considered in model development are

- selection of parameters as inputs signals,
- order of the model,
- number of terms in the model,
- number of measurements to be used.

The selection of the input signals requires some knowledge of the physical system being modeled, (for example, to model the interstand tension between stand 4 and stand 5, the physics of the system would dictate that the primary contributing factors would be the state of the two adjacent stands, that is, stands

```
IN THIS CASE, THE NUMBER OF THE MEASUREMENTS WAS: 24
IN EACH MEASUREMENT VECTOR WE HAD  4 VARIABLES.
THE ORDER OF THE NONLINEARITY WAS:  2
IN THIS CASE THE NUMBER OF THE POSSIBLE CROSS-PRODUCTS:  14
THE ERROR LEVEL WAS:       .0001
THE LIMIT NUMBER OF THE INCLUDED CROSS-PRODUCTS WAS:  10


THE COMPUTER REACHED THE GIVEN ERROR LEVEL IN THE  10-TH STEP


THE NONLINEAR MODEL:



THE X(1) INPUT VARIABLE IS THE 1 ROLL SPEED MEAN
THE X(2) INPUT VARIABLE IS THE 2 ROLL SPEED MEAN
THE X(3) INPUT VARIABLE IS THE 1 ROLL FORCE MEAN
THE X(4) INPUT VARIABLE IS THE 2 ROLL FORCE MEAN
THE OUTPUT VARIABLE IS THE 1 TENSION MEAN
```

COEFF.:	X(1):	X(2):	X(3):	X(4):	X(5):	X(6):
-94.409290	1	0	0	0		
27.875100	0	1	0	0		
-.681578	0	0	1	0		
.362632	0	0	0	1		
-.342173	1	0	1	0		
.399883	0	1	1	0		
-.148724	0	1	0	1		
72.899320	2	0	0	0		
-21.959420	0	2	0	0		
.000292	0	0	2	0		

CONST= .9509E+02

Figure 2.12: Nonlinear prediction model for interstand 1-2 tension with manual clustering (3004 alloy).

```
IN THIS CASE, THE NUMBER OF THE MEASUREMENTS WAS: 24
IN EACH MEASUREMENT VECTOR WE HAD  4 VARIABLES.
THE ORDER OF THE NONLINEARITY WAS:  2
IN THIS CASE THE NUMBER OF THE POSSIBLE CROSS-PRODUCTS:  14
THE ERROR LEVEL WAS:       .0001
THE LIMIT NUMBER OF THE INCLUDED CROSS-PRODUCTS WAS:  10

THE COMPUTER REACHED THE GIVEN ERROR LEVEL IN THE  10-TH STEP

THE NONLINEAR MODEL:


THE X(1) INPUT VARIABLE IS THE 2 ROLL SPEED MEAN
THE X(2) INPUT VARIABLE IS THE 3 ROLL SPEED MEAN
THE X(3) INPUT VARIABLE IS THE 2 ROLL FORCE MEAN
THE X(4) INPUT VARIABLE IS THE 3 ROLL FORCE MEAN
THE OUTPUT VARIABLE IS THE 2 TENSION MEAN
```

COEFF.:	X(1):	X(2):	X(3):	X(4):	X(5):	X(6):
203.424700	0	1	0	0		
.353929	0	0	1	0		
95.207370	1	1	0	0		
.492865	1	0	1	0		
-.278461	1	0	0	1		
-.387381	0	1	1	0		
.002234	0	0	1	1		
-99.666660	2	0	0	0		
-39.839790	0	2	0	0		
-.000624	0	0	2	0		

```
CONST= -.4375E+03
```

Figure 2.13: Nonlinear prediction model for interstand 2-3 tension with manual clustering (3004 alloy).

```
IN THIS CASE, THE NUMBER OF THE MEASUREMENTS WAS: 24
IN EACH MEASUREMENT VECTOR WE HAD  4 VARIABLES.
THE ORDER OF THE NONLINEARITY WAS:  2
IN THIS CASE THE NUMBER OF THE POSSIBLE CROSS-PRODUCTS:  14
THE ERROR LEVEL WAS:       .0001
THE LIMIT NUMBER OF THE INCLUDED CROSS-PRODUCTS WAS:  10

THE COMPUTER REACHED THE GIVEN ERROR LEVEL IN THE  10-TH STEP

THE NONLINEAR MODEL:


THE X(1) INPUT VARIABLE IS THE 3 ROLL SPEED MEAN
THE X(2) INPUT VARIABLE IS THE 4 ROLL SPEED MEAN
THE X(3) INPUT VARIABLE IS THE 3 ROLL FORCE MEAN
THE X(4) INPUT VARIABLE IS THE 4 ROLL FORCE MEAN
THE OUTPUT VARIABLE IS THE 3 TENSION MEAN
```

COEFF.:	X(1):	X(2):	X(3):	X(4):	X(5):	X(6):
23.705190	1	0	0	0		
-286.112800	0	1	0	0		
3.427699	0	0	0	1		
-9.411033	1	1	0	0		
.236540	1	0	0	1		
-1.037130	0	1	0	1		
-.006196	0	0	1	1		
37.863080	0	2	0	0		
.002351	0	0	2	0		
.011311	0	0	0	2		

CONST= .6248E+03

Figure 2.14: Nonlinear prediction model for interstand 3-4 tension with manual clustering (3004 alloy).

```
IN THIS CASE, THE NUMBER OF THE MEASUREMENTS WAS: 24
IN EACH MEASUREMENT VECTOR WE HAD  4 VARIABLES.
THE ORDER OF THE NONLINEARITY WAS:  2
IN THIS CASE THE NUMBER OF THE POSSIBLE CROSS-PRODUCTS:  14
THE ERROR LEVEL WAS:       .0001
THE LIMIT NUMBER OF THE INCLUDED CROSS-PRODUCTS WAS:  10


THE COMPUTER REACHED THE GIVEN ERROR LEVEL IN THE  10-TH STEP


THE NONLINEAR MODEL:



THE X(1) INPUT VARIABLE IS THE 4 Roll Speed Mean
THE X(2) INPUT VARIABLE IS THE 5 Roll Speed Mean
THE X(3) INPUT VARIABLE IS THE 4 Roll Force Mean
THE X(4) INPUT VARIABLE IS THE 5 Roll Force Mean
THE OUTPUT VARIABLE IS THE 4 Tension Mean
```

COEFF.:	X(1):	X(2):	X(3):	X(4):	X(5):	X(6):
-313.713000	1	0	0	0		
6.622376	0	0	1	0		
-20.386910	1	1	0	0		
-1.227183	1	0	1	0		
.405870	1	0	0	1		
.244548	0	1	1	0		
-.009901	0	0	1	1		
48.338580	2	0	0	0		
6.134407	0	2	0	0		
-.006308	0	0	0	2		

CONST= .5460E+03

+

Figure 2.15: Nonlinear prediction model for interstand 4-5 tension with manual clustering (3004 alloy).

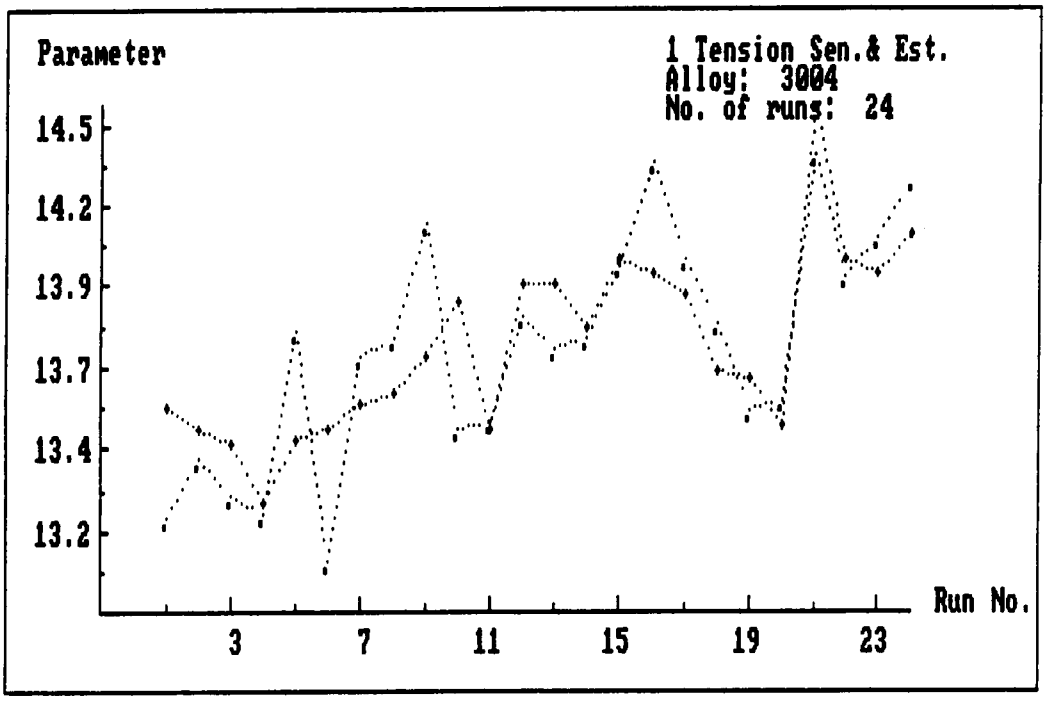


Figure 2.16: Interstand 1-2 tension - measured and predicted values (* 100 PSI) with the manual clustering approach.

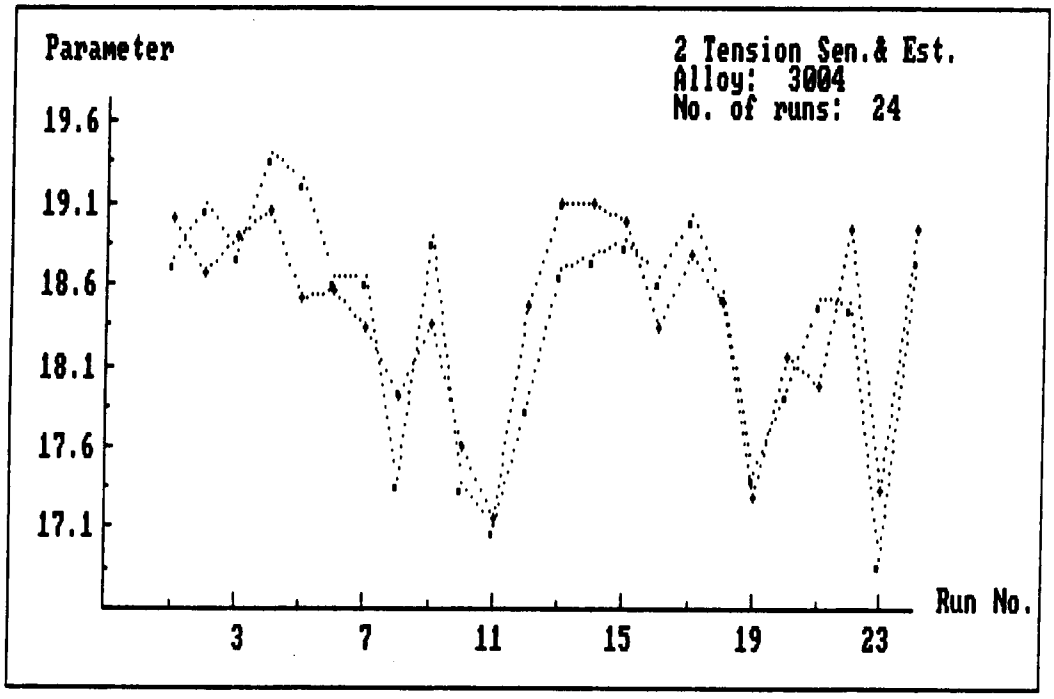


Figure 2.17: Interstand 2-3 tension - measured and predicted values (* 100 PSI) with the manual clustering approach.

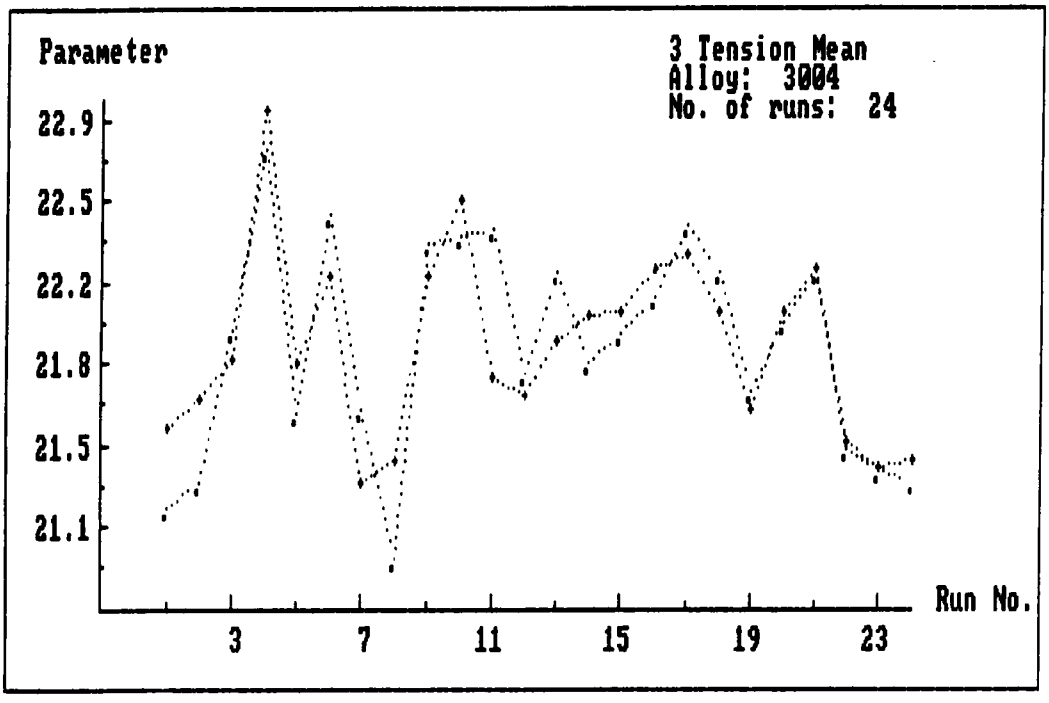


Figure 2.18: Interstand 3-4 tension - measured and predicted values (* 100 PSI) with the manual clustering approach.

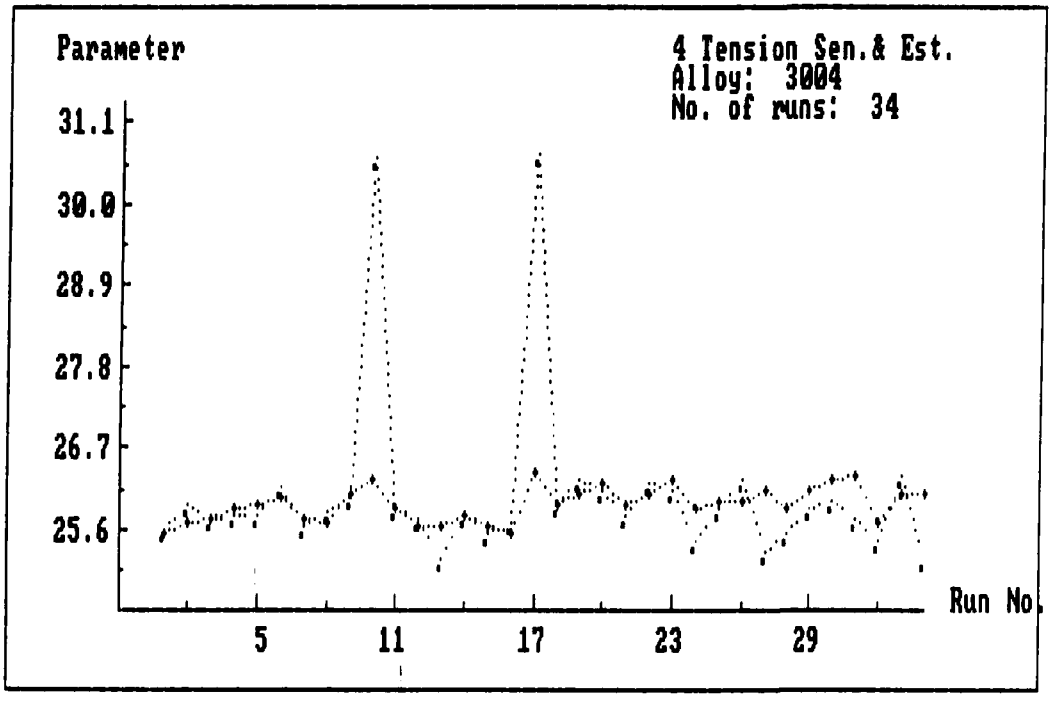


Figure 2.19: Interstand 4-5 tension - measured and predicted values (* 100 PSI) with the manual clustering approach.

4 and 5, rather than the stands farther away). The selection process further requires experimentation with various combinations of signals. Optimum results for tension were obtained using the following combinations:

Interstand tension 1 :	Stand 1 roll speed
	Stand 2 roll speed
	Stand 1 roll force
	Stand 2 roll force
Interstand tension 2 :	Stand 2 roll speed
	Stand 3 roll speed
	Stand 2 roll force
	Stand 3 roll force
Interstand tension 3 :	Stand 3 roll speed
	Stand 4 roll speed
	Stand 3 roll force
	Stand 4 roll force
Interstand tension 4 :	Stand 4 roll speed
	Stand 5 roll speed
	Stand 4 roll force
	Stand 5 roll force

For selecting the maximum order of each of the polynomial terms, the algorithm was run with different orders specified. In case of the four interstand tensions, a second order model was found to be generally optimal. The penalty

for increasing the order of the model is reflected in computational difficulty.

The algorithm returns the total least-squares error after including each additional polynomial term. This allows termination of the program when sufficiently low error is reached. In case of the four interstand tensions it was found that including 10 polynomial terms resulted in an error of less than 2 % between measured and predicted values. To facilitate the comparison of the models for different parameters, the number of polynomial terms in each model is kept the same (equal to 10). Figure 2.19 shows an example of the detection of an abnormal state by the prediction model. The interstand 4-5 tension signals from coil number 10 and 17 are predicted by the model in the same range as the other readings. Figure 2.20 shows a graphical display of the model output. The length of the horizontal bar (upper left) is proportional to the error between the sensor and the estimate. The display shows an anomaly being detected by the model accompanied by a *beep* to warn the operator.

The method used here for computing the model from a cluster of data, generated using automated clustering, is exactly the same as that used for manual clustering. Two algorithms used are the K-Means and the Single Linkage Cluster Analysis (SLCA) algorithms. The results of application of the SLCA algorithm are discussed in chapter 3.

The sample patterns used for clustering are of dimension 5 and are constructed using the signals

- stand 4 roll force,

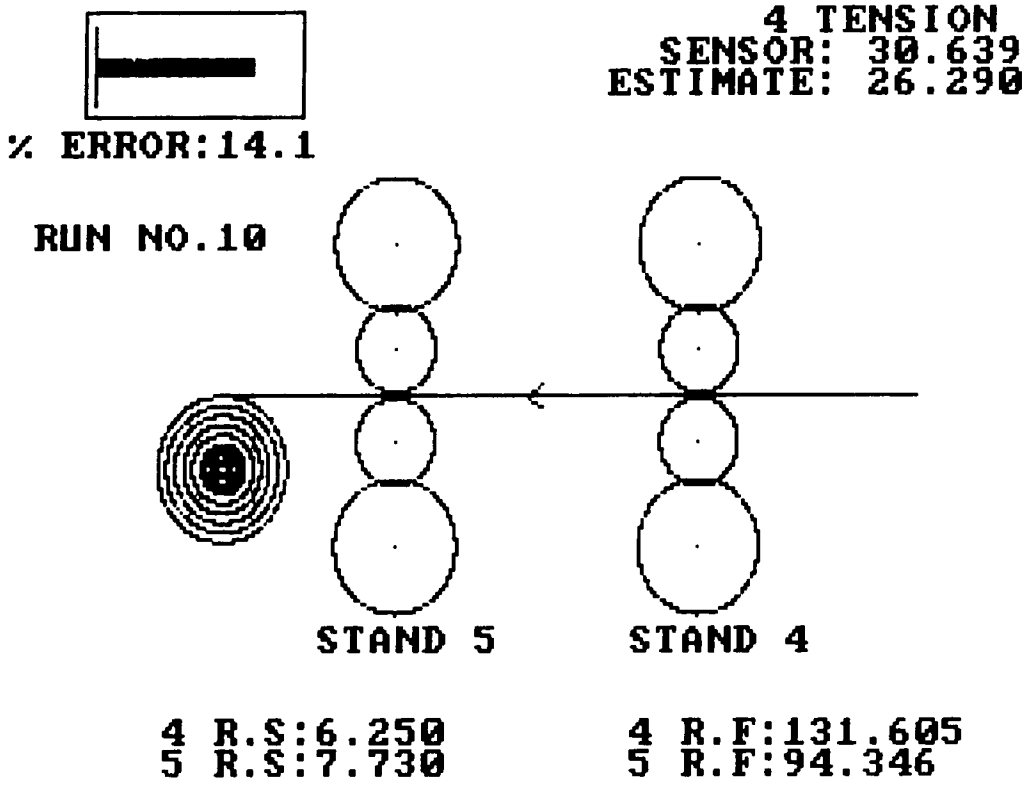


Figure 2.20: Graphical display showing the measured and predicted values of interstand 4-5 tension for coil number 10. The sensor reading is identified as abnormal as indicated by a large value of error.

- stand 5 roll force,
- stand 4 roll speed,
- stand 5 roll speed, and
- interstand 4-5 tension, which is the parameter being estimated.

Figure 2.21 shows the results of clustering the data using the K-Means algorithm. The four clusters and their membership are shown. A model was developed for each cluster and the results obtained were much superior to the model obtained with manual clustering. Figure 2.22 gives the model obtained for cluster 1 and Fig. 2.23 shows a plot of the measured and predicted values of interstand 4-5 tension.

2.5 Description of Software

2.5.1 Nonlinear Empirical Modeling

A software system has been developed [9] for nonlinear empirical modeling and is implemented in an IBM PC. In this section the program is discussed in detail.

The body of the program is divided into three major parts. The first part contains several subroutines for entering the data and the parameters of the fit (polynomial order, number of terms, error level, etc.). The second part performs the actual steps of the algorithm. The final section fits a linear-type model to the selected cross product terms and prints out all the results.

SOLUTION CLUSTER CENTERS:

25.8935700	6.1421430	7.9621430	126.2366000
103.4069000			
26.2677500	5.8365010	7.3495000	142.0838000
104.6272000			
25.8200000	6.1533340	7.4633330	130.3177000
90.6686400			
25.6363400	5.7558540	7.1658540	82.1826900
66.2589500			

CLUSTER NO.: 1	CLUSTER NO.: 2	CLUSTER NO.: 3	CLUSTER NO.: 4
1	2	3	4
5	7	11	27
6	8	12	28
19	9		29
20	10		30
21	13		31
22	14		32
23	15		33
24	16		34
25	17		35
26	18		36
66	59		37
97	60		38
98	61		39
	62		40
	63		41
	64		42
	65		43
	67		44
	68		45
	69		46
	70		47
	71		48
	72		49
	73		50
	74		51
	75		52
	76		53
	85		54
	86		55
	87		56
	88		57
	89		58
	90		77
	91		78
	92		79
	93		80
	94		81
	95		82
	96		83
			84

Figure 2.21: The four clusters formed using the K-Means algorithm.

```
IN THIS CASE, THE NUMBER OF THE MEASUREMENTS WAS: 14
IN EACH MEASUREMENT VECTOR WE HAD  4 VARIABLES.
THE ORDER OF THE NONLINEARITY WAS:  2
IN THIS CASE THE NUMBER OF THE POSSIBLE CROSS-PRODUCTS:  14
THE ERROR LEVEL WAS:       .0001
THE LIMIT NUMBER OF THE INCLUDED CROSS-PRODUCTS WAS:  10


THE COMPUTER REACHED THE GIVEN ERROR LEVEL IN THE  10-TH STEP


THE NONLINEAR MODEL:



THE X(1) INPUT VARIABLE IS THE 4 Roll Speed Mean
THE X(2) INPUT VARIABLE IS THE 5 Roll Speed Mean
THE X(3) INPUT VARIABLE IS THE 4 Roll Force Mean
THE X(4) INPUT VARIABLE IS THE 5 Roll Force Mean
THE OUTPUT VARIABLE IS THE 4 Tension Mean
```

COEFF.:	X(1):	X(2):	X(3):	X(4):	X(5):	X(6):
-1018.790000	1	0	0	0		
-3.147631	0	0	1	0		
-17.798700	0	0	0	1		
-31.719330	1	1	0	0		
2.086563	1	0	0	1		
.636381	0	1	1	0		
1.157218	0	1	0	1		
86.392910	2	0	0	0		
-.008163	0	0	2	0		
-.021277	0	0	0	2		

CONST= .4253E+04

Figure 2.22: Nonlinear prediction model for interstand 4-5 tension (cluster 1).

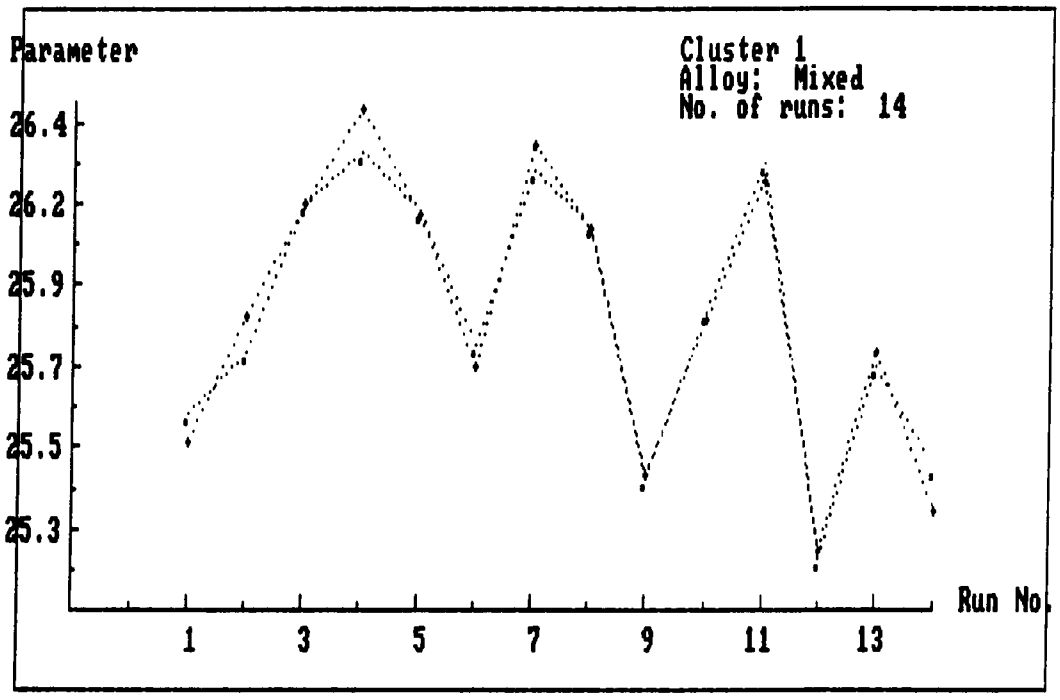


Figure 2.23: Interstand 4-5 tension - measured and predicted values (* 100 PSI) for cluster 1.

There are two supporting programs written to prepare the data for the nonlinear modeling program. The raw data file consists of 66 signals from each run, the first program allows selection of the specific measurements out of the 66 available. It further allows the selection of the number of runs to be used. That is, out of the complete data set the program facilitates the selection of a specific number of runs from any part of the data set. All the selected runs and measurements are written into a direct access file which is an input file for the second program.

The second program allows the user to specify any of the measurements as inputs or outputs. This program stores the column numbers of the input and output variables, the identification numbers of the measurements at the beginning of a new file, and also stores the names of the signals in the measurement vectors.

The first part of the nonlinear modeling program loads the desired measurement data file. It asks for the identification numbers of the measurement vectors to be included in the calculations, the order of nonlinearity, error level, and the number of terms in the model.

The second part of the program is the most important from a computational viewpoint. There are 5 subroutines to perform the calculations.

Subroutine 1 Calculates all the projection matrices dependent on the place-

ment of the M vectors in the subspace,

$$\mathbf{P}(i) = \frac{[\mathbf{v}^j(i)][\mathbf{v}^j(i)]^T}{[\mathbf{v}^j(i)]^T[\mathbf{v}^j(i)]}, \qquad i = 1, \ldots, M \tag{2.6}$$

where $j = 0$ at the first step and,

$$\mathbf{v}^0(i) = \mathbf{v}(i) \tag{2.7}$$

where $\mathbf{v}(i)$ is determined by Eq. (2.2) and j is increased step by step.

Subroutine 2 Projects the actual output vector in different directions, and calculates the scalar length of the projected vectors as,

$$\mathbf{z}^j(i) = \mathbf{P}(i)\ \mathbf{z}^j \tag{2.8}$$

and the length is given by,

$$R(i) = [\mathbf{z}^j(i)]^T[\mathbf{z}^j(i)] \tag{2.9}$$

Subroutine 3 Finds the largest scalar length (the largest number among the $R(i)$'s) and identifies the cross-product to be included as the next term in the model.

Subroutine 4 Calculates the projection matrix and projects the rest of the vectors into a subspace orthogonal to the selected vector as,

$$\mathbf{P}(i_k) = \frac{[\mathbf{v}^j(i_k)][\mathbf{v}^j(i_k)]^T}{[\mathbf{v}^j(i_k)]^T[\mathbf{v}^j(i_k)]} \tag{2.10}$$

and,

$$\mathbf{v}^{j+1}(i) = \mathbf{M}^j\mathbf{v}^j(i)\ , \quad i = 1, \ldots, M\ ;\ i \neq i_l\ ;\ l = 1, \ldots, j \tag{2.11}$$

where,

$$\mathbf{M}^j = \mathbf{I} - \sum_{l=1}^{j} \mathbf{P}(i_l) \tag{2.12}$$

and $P(i_l)$'s are the projection matrices corresponding to the terms already selected.

Subroutine 5 Projects the output vectors into the next subspace and checks the error level given by,

$$\mathbf{z}^{j+1}(i) = \mathbf{M}^j \mathbf{z}^j(i) \tag{2.13}$$

The last part of the program regenerates the original vectors of the initial vector space depending on the selected terms. It then fits a linear model, calculating the coefficients of the cross-products and a constant term which increases the precision of the fit greatly. The final step is to print out the calculated output values and compare them to the measured values.

CHAPTER 3

Signal Validation Module

3.1 Introduction

The *Signal Validation Module* (SVM) is designed to validate instrument outputs. The software system consists mainly of a database , data classification routines, input preprocessing, and a method to check for gradual degradation of sensors. The SVM represents the integration of the clustering and empirical modeling described in chapter 2. A structured data base is formed from mean coil data acquired from the ALCOA Hot Line Facility. Though the SVM is demonstrated using this data, the technique can be applied to any process control system. The SVM is able to preprocess signals to be validated, assign them to a known cluster, and apply the prediction model for that cluster to validate the signals. The results are presented using highly interpretive graphics.

3.2 The Data Base

Figure 3.1 shows the flow chart for the construction of the data base. The root level is the data set containing 400 coils of mean value data. The signal being validated is the interstand 4-5 tension. This and the four input signals are

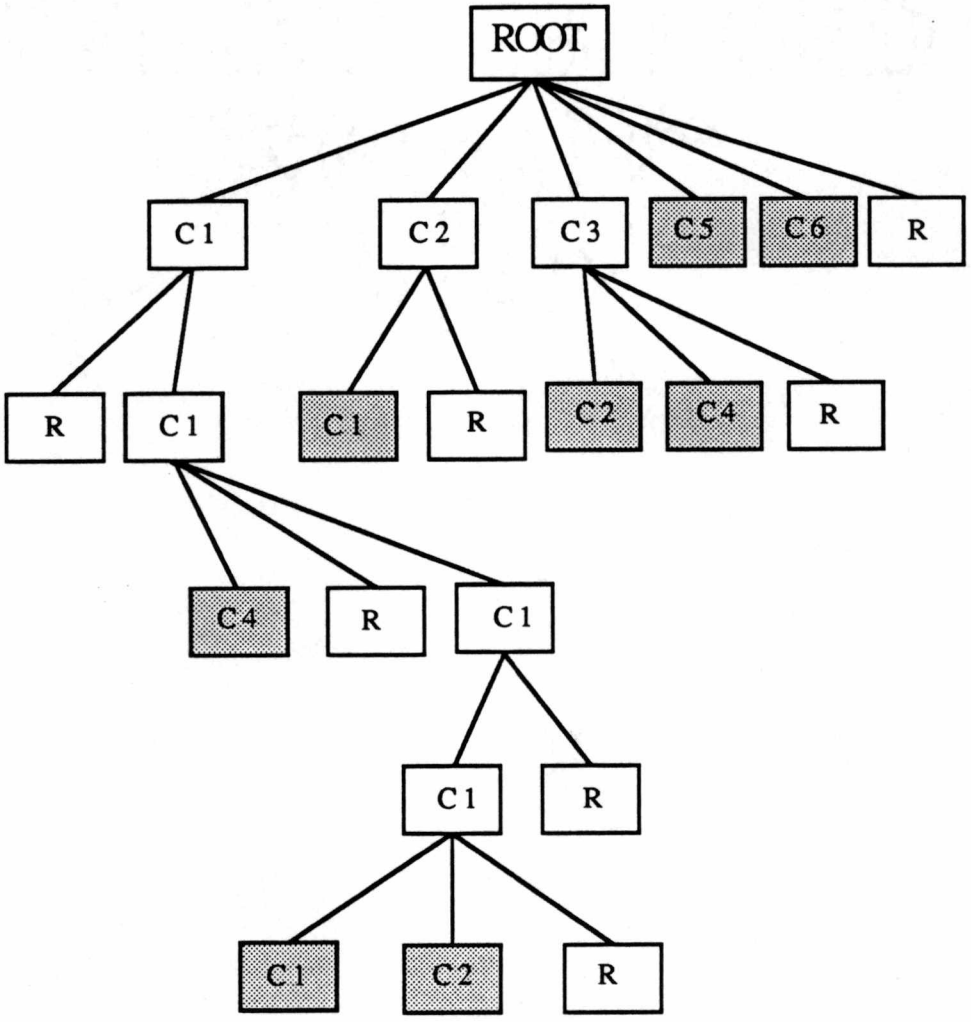


Figure 3.1: Tree showing the hierarchal clustering of the data. The shaded boxes are the final clusters that are included in the data base. 'R' indicates the clusters that are rejected.

formed into a five-dimensional vector corresponding to each run. Each pattern vector has the form

$$\underbrace{\{Y^1, X_1^1, X_2^1, X_3^1, X_4^1\}}_{5\ dimensional\ vector}$$

The use of these parameters for empirical modeling ensures that all the different process states of each of these parameters are represented in the clustering. A scheme called *hierarchical clustering* is used to cluster the data. This method is explained in Section 3.2.3.

3.2.1 Scaling of Pattern Vectors

Scaling is performed so that all the components of a multidimensional pattern vector have the same order of magnitude. The components of the pattern vectors are derived from different process parameters and vary from each other in magnitude. This causes the larger components to have greater influence than the smaller components in the outcome of the clustering process. It is necessary to scale each component of the pattern vector so that they have equal contribution to the clustering process. The scaling factors are automatically determined by the program and are stored to enable rescaling of the final results.

3.2.2 Determination of the Optimal Number of Clusters

The number of divisions of the data set is a compromise between achieving compact clusters and not having too many clusters. A performance index is used to determine the quality of clustering for different cluster combinations.

This is the cumulative mean-squared error given by

$$J = \sum_{j=1}^{K} \sum_{x \in S_j} \| x - Z_j \|^2 \tag{3.1}$$

where J is the preformance index, K is the number of clusters and Z_j is mean of cluster domain S_j. The error is computed for different number of clusters and Fig. 3.2 shows a plot the error versus the number of clusters. The optimal number is selected at a break point of the curve where there is a significant reduction in the cumulative error after which any increase in the number of clusters does not entail a significant drop in the error. Figure 3.2 shows the break point at eight clusters.

3.2.3 Hierarchical Clustering

Figure 3.1 shows the flow chart for hierarchical clustering. The resolution of the clustering outcome is increased by this method. It allows for further break-up of large clusters by using the members of large cluster domains as inputs to the cluster seeking program. This results in further break-up of large clusters at each level of hierarchy and finally producing compact clusters having a roughly balanced membership. The clustering algorithm used is the *single linkage cluster analysis algorithm* described in chapter 2. This algorithm is sensitive to noise, which is used advantageously since all the abnormal (noisy) points fall out in single membership clusters at each hierarchical level. Abnormal points are thus weeded out at each cluster level and rejected. The appendix shows the clusters generated at the second level of hierarchy. The 46 input patterns belong to clus-

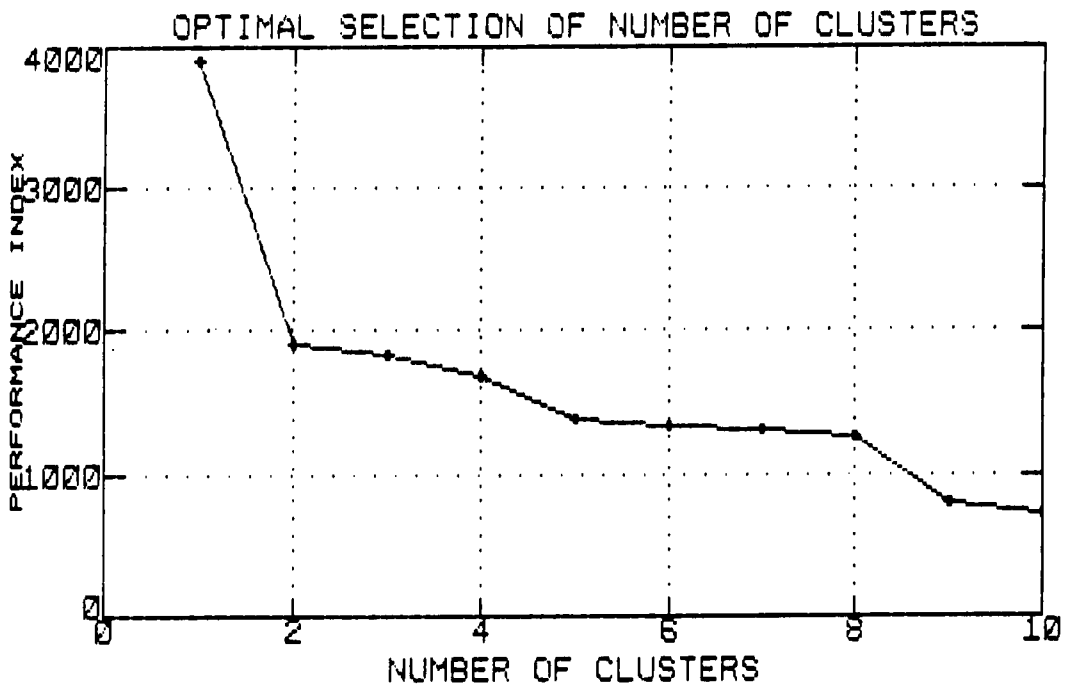


Figure 3.2: Determination of the optimal number of clusters using the cumulative mean squared error as the performance index.

ter 2 generated at the first level of hierarchy. Note that the single membership clusters constitute the rejected pattern vectors.

The geometrical properties of a multi-dimensional space is impossible to visualize, making the evaluation of the clustering results difficult. Therefore some methods are employed that allow partial insight into the geometrical properties of the resulting cluster domains. The variance of a cluster domain about its mean value is a useful parameter in determining the distribution of the data points in a cluster domain. The appendix includes the variance table for each of the cluster domains in that level of hierarchy. The primary spread or elongation of each cluster domain is in the direction of stand 4 roll force and the stand 5 roll force while the other three dimensions are fairly compact in comparison. Table 3.1 shows the variances of clusters at different levels of the hierarchical tree but along one branch of the tree. Note the reduction in the variances after each level indicating an increasing compactness in the cluster domains. Each row in the table corresponds to a level in the tree hierarchy and the columns represent individual components of the multi-dimensional pattern vector. The abbreviations RS and RF stand for Roll Speed and Roll Force, respectively.

Computing the mean of each cluster domain also lends some insight into the relative positioning of the cluster domains with respect to each other. Table 3.2 shows mean of all the cluster domains at the bottom of the tree. The values indicate distinct spacing between the clusters. This is partly because of the compactness of the clusters and removal of all the abnormal points.The final

Table 3.1: Variances of the cluster domains at different levels of hierarchy indicating increasing compactness.

level	*4-5 ten*	*4 RS*	*5 RS*	*4 RF*	*5 RF*
1	3.46	1.17	5.97	29.86	44.06
2	3.05	1.07	5.45	22.34	38.07
3	2.79	0.74	2.65	21.14	28.64
4	2.46	0.61	2.52	20.52	25.59
5	2.35	0.44	1.58	18.37	25.47
6	1.90	0.44	1.25	15.55	13.49

Table 3.2: Mean values of all the cluster domains at the bottom of the hierarchal tree.

cluster	*4-5 ten*	*4 RS*	*5 RS*	*4 RF*	*5 RF*
1	25.89	6.15	8.09	126.47	111.86
2	25.99	5.21	6.44	123.601	98.02
3	26.10	5.93	7.84	68.22	59.20
4	25.54	5.96	7.82	69.69	58.91
5	25.74	5.24	6.80	122.41	120.08
6	25.94	6.08	7.78	101.56	89.53
7	25.87	6.13	7.61	129.33	98.09
8	25.85	6.15	8.05	136.08	116.54

number of clusters are eight, as determined by the cluster validity criterion.

Nonlinear empirical models are generated for each of the cluster domains as described in chapter 2. Membership data of each cluster is used as the training data to construct the model characterizing that cluster. Each model and cluster domain is stored in separate direct-access files and they collectively form the data base.

3.3 Description of the Signal Validation Module

The SVM serves to coordinate the data base, assign new test patterns to known clusters, apply the correct model, check for slow degradation and use extensive graphics to present the results. Figure 3.3 shows the flow chart describing the operation of the SVM.

To validate a test input signal the program first picks out the signals to form a 5-dimensional pattern vector as described in Sect. 3.2. A 1-Nearest Neighbor classifier [14] is used to assign the test pattern to one of the known clusters in the data base. The 1-Nearest Neighbor classifier assigns the test pattern to the cluster domain of its closest (according to the distance measure used) pattern. Once the pattern is classified, the model characterizing the cluster to which the assignment was made, is applied to make a prediction of interstand 4-5 tension. The prediction is compared to the measured value and an alarm is indicated if the error exceeds the specified threshold. The display produces an error bar of

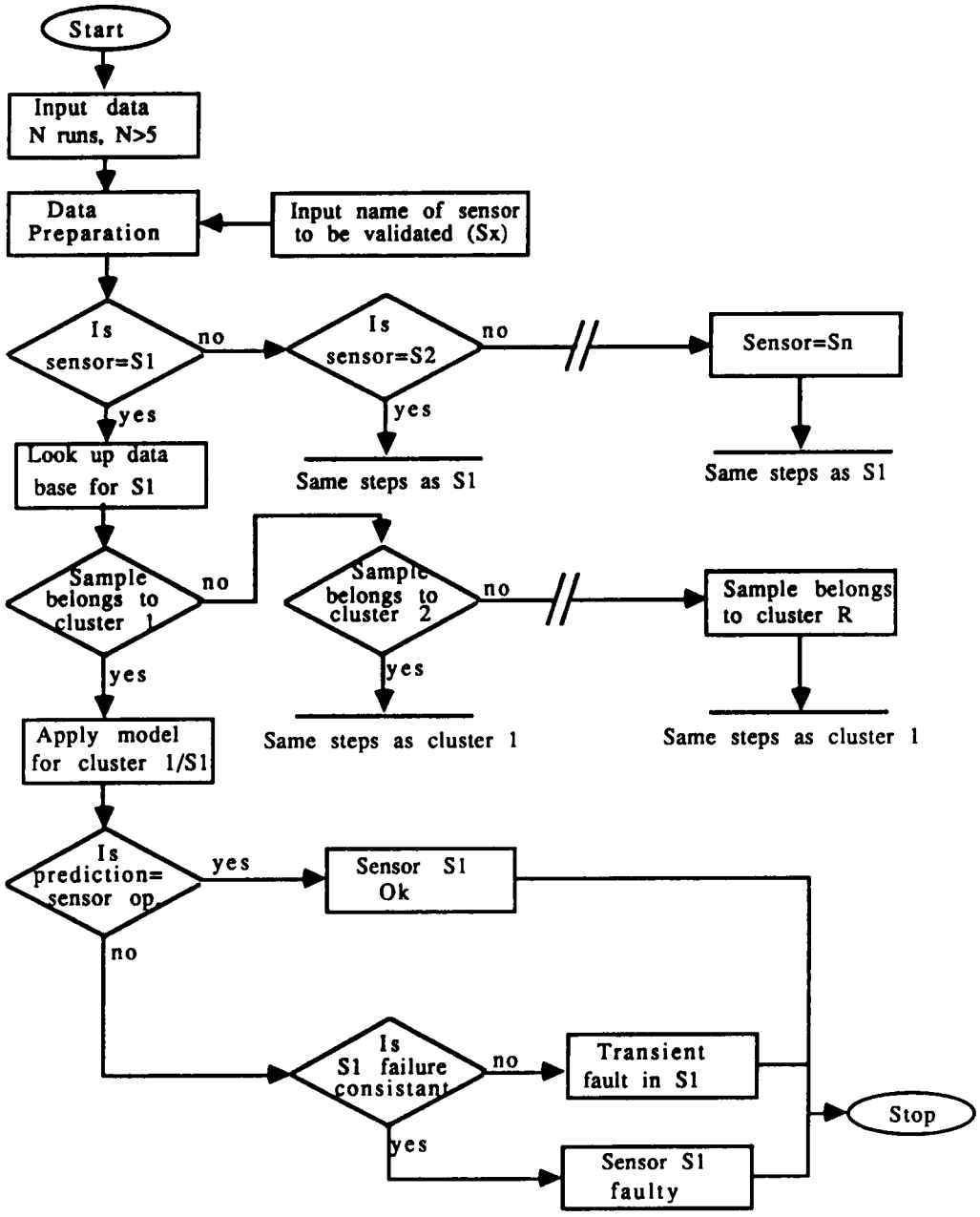


Figure 3.3: Block diagram showing the working of the Signal Validation Module.

length proportional to the error between the measured and the predicted values. The display also shows the cluster to which the test pattern is assigned and the cluster's mean value. This enables a comparison of the test pattern with the mean of the cluster assignment which may yield information about the deviation in any of the driving signals used for prediction.

Before declaring a sensor as faulty a check is performed to determine if the error is propagated through consecutive test samples. The propagation of error is indicative of a consistent failure of the sensor. When a signal being estimated is normal, the probability of the estimated value being marginally above the measured value is equal to the probability of it being marginally below the measured value. To exploit this property a counter is set up to count down on a negative deviation and to count up on a positive deviation. If a slow degradation exists the counter will increase in magnitude towards either direction. A threshold is defined beyond which a slow degradation is said to exist.

Figures 3.4 and 3.5 give the prediction models for cluster 2 and cluster 6. Figures 3.6 and 3.7 show the measured (+) and predicted (x) values for cluster 2 and cluster 6 respectively. The measured and predicted points follow each other very closly indicating a high model accuracy within the cluster domain. Table 3.3 shows the percentage error for each prediction model after the inclusion of the specified number of polynomial terms. The model error is within 1 % for each of the cluster domains, whereas it is in the range of 3-5 % for the models developed

Output of the nonlinear system modeling:

PARAMETERS:

The number of measurements: 17
The number of input variables: 4
The order of the nonlinearity: 2
The number of possible cross-products: 14

The number of terms in the model: 10

THE MODEL:

Coefficient:	x(1):	x(2):	x(3):	x(4):
.4500E+03	1	0	0	0
.3185E+02	0	1	0	0
.4000E+02	0	0	1	0
-.6022E+01	1	0	1	0
-.8179E+01	1	0	0	1
-.2286E+01	0	1	1	0
.2887E+00	0	0	1	1
-.2079E+02	2	0	0	0
.2222E+00	0	0	2	0
.9265E+00	0	0	0	2

Add const. : -.1520E+04

The x(1) input variable is the 4 ROLL SPEED
The x(2) input variable is the 5 ROLL SPEED
The x(3) input variable is the 4 ROLL FORCE
The x(4) input variable is the 5 ROLL FORCE
The output variable is the 4-5 TENSION

Figure 3.4: Prediction model for interstand 4-5 tension which characterizes cluster 2.

Output of the nonlinear system modeling:

PARAMETERS:

The number of measurements: 16
The number of input variables: 4
The order of the nonlinearity: 2
The number of possible cross-products: 14

The number of terms in the model: 10

THE MODEL:

Coefficient:	x(1):	x(2):	x(3):	x(4):
-.7139E+03	1	0	0	0
.2122E+03	0	1	0	0
.6649E+02	1	1	0	0
.1280E+02	1	0	1	0
-.5438E+01	1	0	0	1
-.1435E+02	0	1	1	0
.1814E+01	0	0	1	1
-.2352E+01	2	0	0	0
-.1669E+02	0	2	0	0
.1543E+01	0	0	2	0

Add const. : .7212E+03

The x(1) input variable is the 4 ROLL SPEED
The x(2) input variable is the 5 ROLL SPEED
The x(3) input variable is the 4 ROLL FORCE
The x(4) input variable is the 5 ROLL FORCE
The output variable is the 4-5 TENSION

Figure 3.5: Prediction model for interstand 4-5 tension which characterizes cluster 6

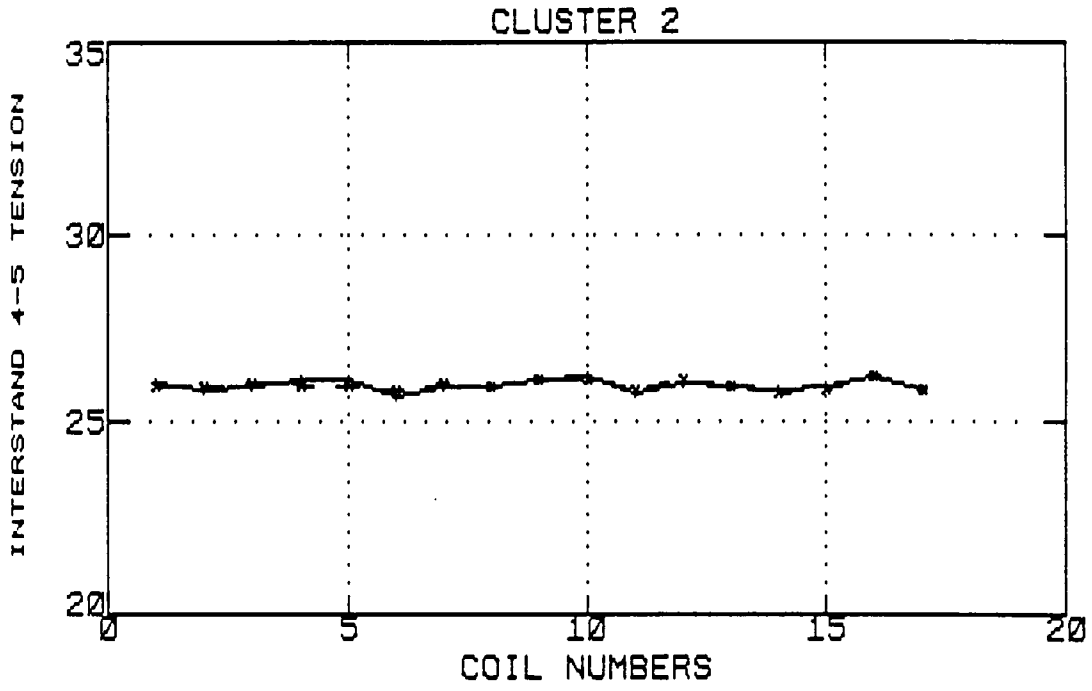


Figure 3.6: Plot showing the measured and predicted values of interstand 4-5 tension within cluster 2.

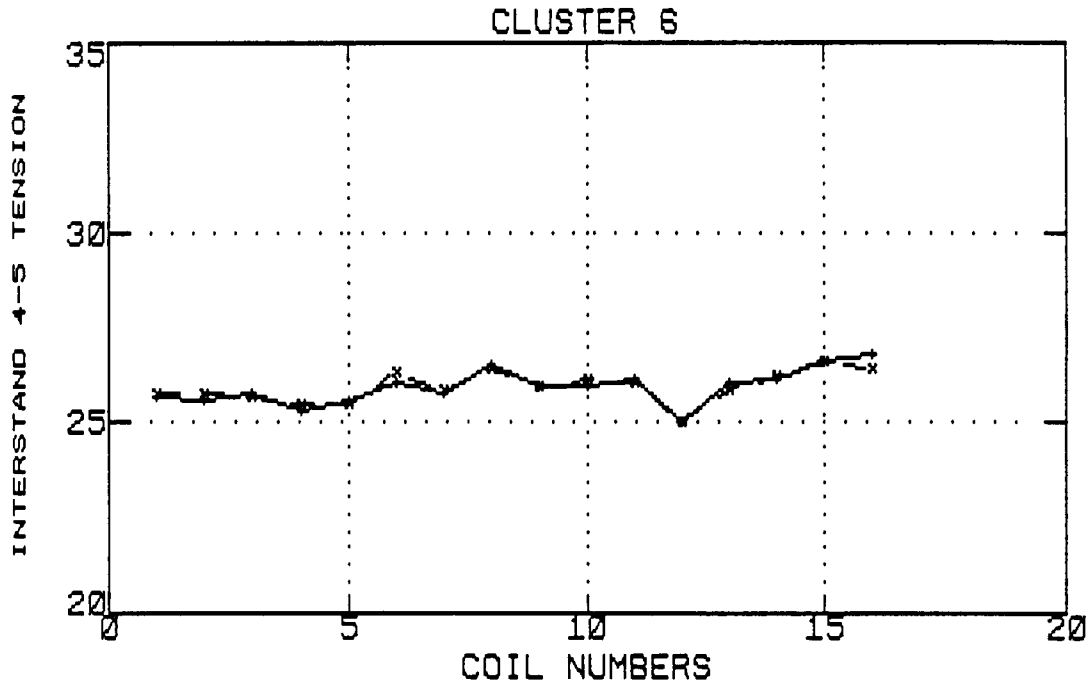


Figure 3.7: Plot showing the measured and predicted values of interstand 4-5 tension within cluster 6.

Table 3.3: Percentage error and the number of terms in the prediction models for each cluster.

cluster	*no. of terms*	*% error*
1	7	0.279
2	10	0.344
3	8	0.165
4	10	0.974
5	10	0.308
6	10	0.637
7	10	0.536
8	8	0.901

prior to clustering. This reduction in modeling error is due to the compactness of the cluster domains. Figure 3.8 shows a comparison of the prediction accuracy with test data using the SVM and using a single model characterizing the entire data base. The plot shows the distinct improvement in prediction accuracy with the SVM.

3.4 Description of Software

The software is written in `FORTRAN` and implemented in an IBM PC. It has a modular structure to allow for easy expansion to include more clusters. There are three files used to represent each cluster and each of them is named using a standard nomenclature.

CLU_*.SVM This is the file containing the membership of that cluster.

MOD_*.SVM This file contains the prediction model characterizing that cluster.

SCA_*.SVM This contains the scaling factors used in the prediction model.

All the files used have a direct access format shown in Table 3.4. The length of each record in the file is four bytes. The file containing the test data (to be validated) is stored in a file with a structure similar to the file `CLU_*.SVM` shown in Table 3.4.

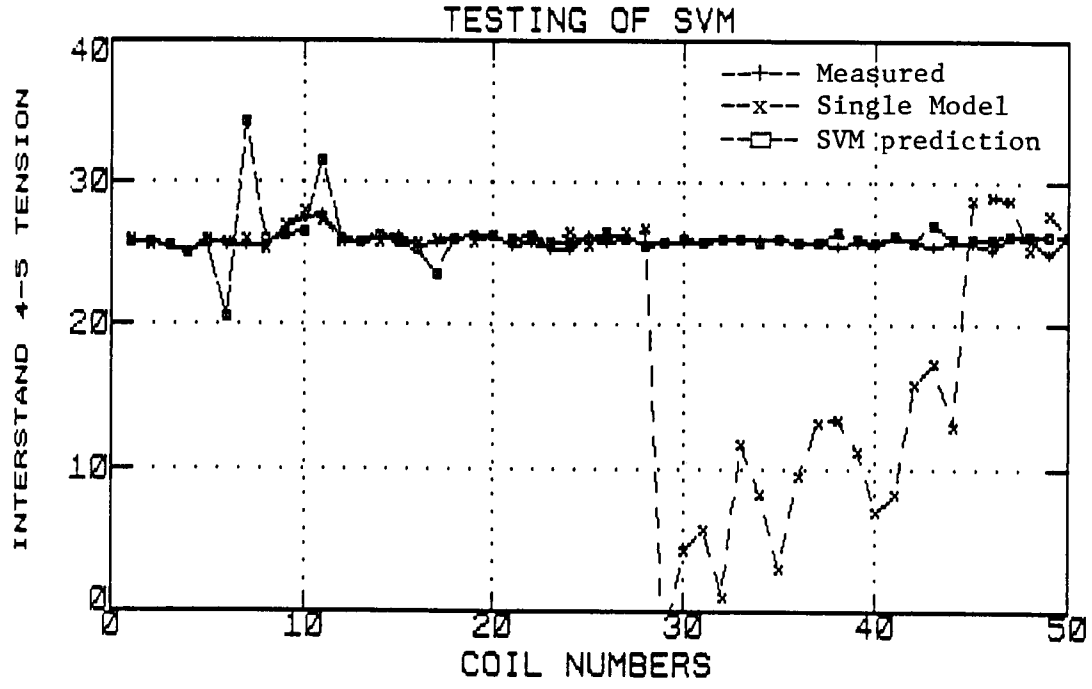


Figure 3.8: Comparison of the prediction accuracy between the SVM and a model characterizing the entire database.

Table 3.4: The structure of the database files in the SVM.

rec.	*CLU_*.SVM*	*MOD_*.SVM*	*SCA_*.SVM*
1	dimension	# of cross-products	dimension
2	# of patterns	# of terms	# of patterns
3	-	% error	-
4	-	expansion slot	-
5	pattern 1	# of inputs	scale factor 1
6	-	coeff. of term 1	-
7	-	order of term 1	-
9	-	order of term 2	-
10	-	order of term 3	-
11	pattern 2	order of term 4	scale factor 2
12	-	-	-
13	-	-	-

The program is executed in a DOS environment with the command 'SVM'. The inputs required by the SVM are

- filename containing test patterns, and
- number of test patterns to be validated.

The output display shows the cluster assignment of each test pattern, the mean of the assigned cluster, measured and the predicted values of the interstand 4-5 tension, and the gradual degradation counter. The program is designed to minimize user interface and ease of operation.

CHAPTER 4

Bidirectional Associative Memories

4.1 Introduction

This chapter and the next address an extension of the classification and validation of instrument outputs applied to the fluctuating components of the signals derived from processes. Some of the methods of neural computing are applied to this problem. The Bidirectional Associative Memory (BAM) [2] allows mapping from data to data and represents a mathematical abstraction from the associative structure of human learning. Any input spatial pattern can be used to recall an associated pattern stored in the associative memory. The BAM resembles a transfer function of a physical system through which a transformation is performed on the input signal. The BAM operation is a two-stage process involving BAM encoding and BAM decoding. The encoding constitutes the learning stage where multiple associations from one pattern to another are learned and stored in the BAM. The decoding process involves presenting a new input pattern to the BAM and recalling the closest association learned in the encoding stage. An optimally designed BAM has considerable error correcting properties and has the ability to generate a successful recall even if only a part of the input pat-

tern, or a noisy input pattern is presented. This property of the BAM makes it suitable for recognition of frequency domain spectral signatures, since members of the same class may differ considerably from each other, making automated classification with conventional techniques difficult. This chapter explains the technique and the results obtained from the application of this paradigm, used for classification of power spectral density functions (PSD) generated from data acquired from `Celanese` company. The learning rule is *Hebbian*, where the connection weights of an interconnection is strengthened if the input is high and the desired output is high, and weakened otherwise. In biological terms, the strength of a pathway increases each time it is used.

4.2 BAM Algorithm for Fault Classification

4.2.1 Outline of BAM

The BAM is a two-layer feedback network of interconnected neurons or nodes. Each node a_i in layer F_A is totally connected to every node b_j in layer F_B and conversely (Figure 4.1). The neurons are also known as processing elements (PE). None of the nodes within a layer is connected to another node within the same layer. These interconnections have weights associated with them which are set during the BAM *learning* phase. The input pattern is presented to layer F_A and passes through the interconnections such that a weighted sum of the inputs reaches the output layer F_B. The current value of each node in F_B is thresholded

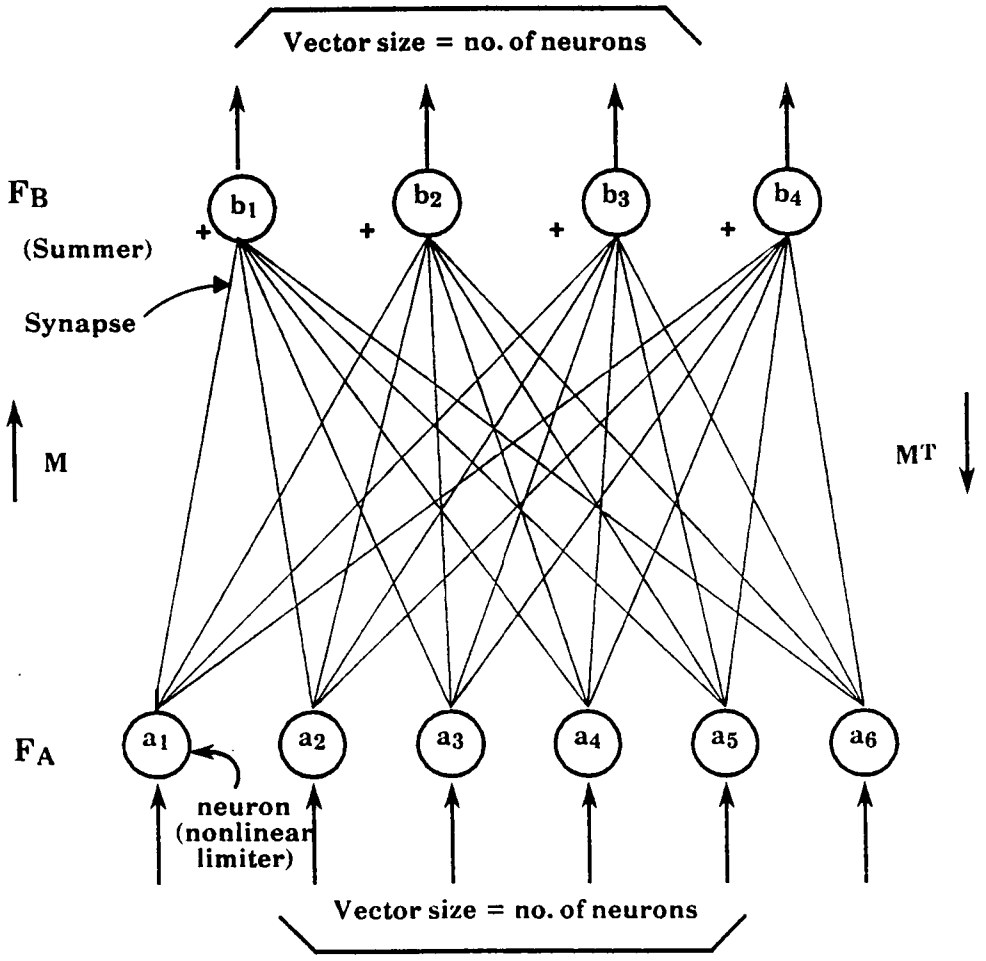


Figure 4.1: Topology of a BAM showing the two fields of neurons and the interconnections.

by a hard limiting thresholding function. The thresholded outputs are fed back to the inputs until the BAM converges. The bidirectionality of the BAM allows a pattern b_j to be fed to layer F_A recalling the corresponding element a_i of the learned association (A_i, B_j).

4.2.2 Spectral Coding

Coding of the patterns to be classified is an important step and should be based on the following :

- Determination of the criterion to be used as a basis for distinguishing between the various classes.
- Find the number of representative classes.
- Preprocess the patterns so that the above criterion is highlighted.
- Suppress irrelevant information and noise in the patterns.
- Generate the patterns to be as close to the optimal BAM input patterns, as possible (Discussed in Section 4.2.3).

The apriori determination of the number of classes is a very subjective procedure and requires a heuristic evaluation of all the available training patterns. Furthermore, the BAM accepts binary inputs only, requiring that the spectral signatures be coded in a binary form. A study of various spectral signatures from `Celanese` showed two classes were sufficient to characterize all the training patterns. Once the number of training patterns are determined, a typical

pattern from each class is chosen to train the network. Figure 4.2 shows the two typical patterns chosen to represent their respective classes. The first class is characterized by peaks around 14 Hz and 26 Hz while the second class has peaks at 32 Hz and 39 Hz. A *bit-mapping* procedure is used to code the PSD's in a binary format. Prior to bit-mapping a nonlinear scaling function (Eq. 4.1) is applied to the discrete PSD to enhance the peaks and to normalize the waveform.

$$\frac{\ln(1+t_i)}{\ln(1+t_{max})} \tag{4.1}$$

where t_i is the magnitude at frequency point i.

Bit-mapping constitutes windowing the PSD along both the X and the Y axes. The Y dimension is partitioned into r windows and the X dimension is partitioned into s windows. The value of s is determined by the number of discrete frequency points of the PSD function. This definition of s ensures that always one and only one discrete frequency point will lie within each X window. This results in a $r \times s$ grid (Figure 4.3) and a '1' is assigned at the intersection of each X and Y window if any discrete PSD value lies within it, otherwise the grid element is given a value of '0', given by

$$G_{xy} = \begin{cases} 1 & \text{if } \frac{t_{max}(y-1)}{r} \leq t_x < \frac{t_{max}(y)}{r} \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

where, $x = 1, \ldots, s$ and $y = 1, \ldots, r$.

The binary matrix is converted to a vector of length $r \times s$ by concatenating the rows of the matrix. The vector represents the original PSD in a form that

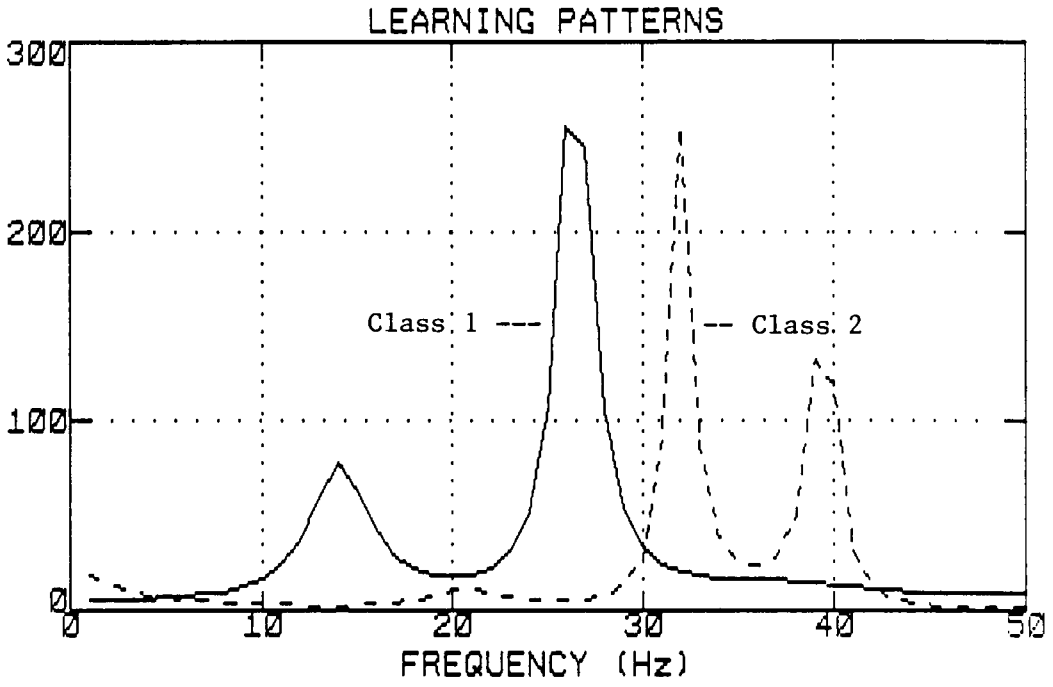


Figure 4.2: A typical representative of each class used to train the BAM.

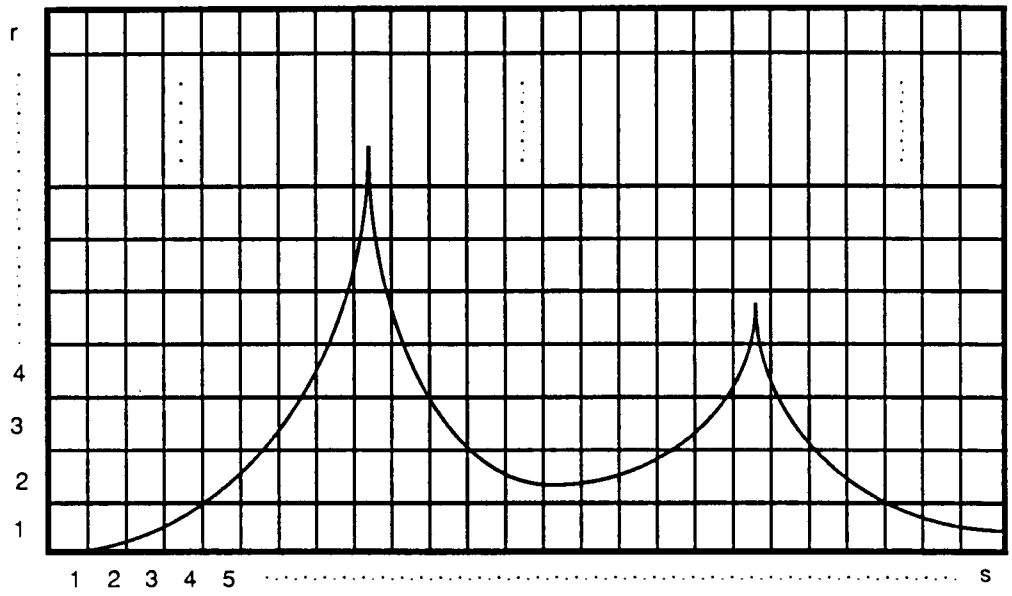


Figure 4.3: The bit-mapping scheme used to code the APSD's as binary vectors.

is ready to be presented to the network. By changing r the resolution of the bit-map can be varied and r can be reduced sufficiently to nullify the effect of low amplitude points while retaining significant peaks in the PSD. Keeping s constant at a high value makes the coding sensitive to the frequency shifts in the pattern. This feature is desirable since the primary criterion for determining the class membership of a spectral signature is the set of frequencies at which the peaks occur.

4.2.3 BAM Encoding

BAM encoding consists of a learning phase, and multiple associations are used to encode the BAM. The form of learning used is *autoassociative*. In autoassociative learning, a mapping is created from the input pattern to itself, and is of the form

$$\{(A_1, A_1), (A_2, A_2), \ldots, (A_n, A_n)\} \tag{4.3}$$

and $A_i = B_i$, $p = q$ for $i = 1, \ldots, n$

where p and q are the number of nodes in layer F_A and F_B respectively, and n is the total number of associations used for training the BAM. Selection of autoassociative encoding also satisfies the condition of the continuity of associations, which must hold for robust recall:

$$\frac{1}{pH(A_i, A_j)} \approx \frac{1}{pH(B_i, B_j)} \quad \forall i, j \tag{4.4}$$

which for autoassociative encoding reduces to

$$\frac{1}{pH(A_i, A_j)} \approx \frac{1}{pH(A_i, A_j)} \quad \forall i, j \tag{4.5}$$

and is trivially satisfied. $H(A_i, A_j)$ is the *Hamming* distance between A_i and A_j. The hamming distance from A_i to A_j is defined as the total number of corresponding bits that differ between A_i and A_j.

Kohonen [15] has shown that for optimal BAM recall all the training patterns $A_i, i = 1, \ldots, n$ should be orthogonal. While it is difficult to ensure this condition without corrupting the patterns, we have found that changing the bit-mapping resolution r in the Y direction tends to satisfy this condition more accurately.

The BAM requires the input patterns to be in a binary form and the associated recall is also in binary. This method is particularly suited for classifying patterns with few characteristic classes, as the classification accuracy decreases with an increase in the number of learning patterns. The limit number of patterns that can be learned is given by

$$n < \min(p, q) \tag{4.6}$$

where p and q are the number of nodes in fields F_A and F_B respectively, and n is the number of learning patterns. Equation (4.6) represents a rough limit for the number of learning patterns, but the actual number for a robust classifier should be less by at least a factor of 2. This limitation can be overcome by increasing the bit-mapping resolution r, though it proves expensive in terms of computer memory utilization.

For successful recall the binary training patterns $(A_1, A_1), \ldots (A_n, A_n)$ must be converted to bipolar vectors $(X_1, X_1), \ldots (X_n, X_n)$ by replacing all zero's with -1's. The BAM matrix is given by,

$$M = \sum_{i=1}^{n} X_i^T X_i \tag{4.7}$$

M is a $p \times p$ symmetric matrix when autoassociative encoding is performed.

4.2.4 BAM Decoding

BAM decoding is a process of associative recall. The new pattern (in binary form) is presented to layer F_A which causes triggering of nodes in F_A. The input fan out over the interconnections to F_B whre it is summed up at the nodes of F_B and thresholded. Mathematically

$$Z^k = Z^{k-1} M \tag{4.8}$$

where Z^k is the output of the BAM at iteration k and Z^{k-1} is the binary test input.

From Eq. 4.8 the BAM output at node j of F_B can be described as,

$$z_j^{k+1} = F\left(\sum_{i=1}^{p} z_j^k M_{i,j}\right) \tag{4.9}$$

The thresholding function $F(x)$ is most optimal [16] if it is a sigmoid

$$F(x) = \frac{1}{(1 + e^{-x})} \tag{4.10}$$

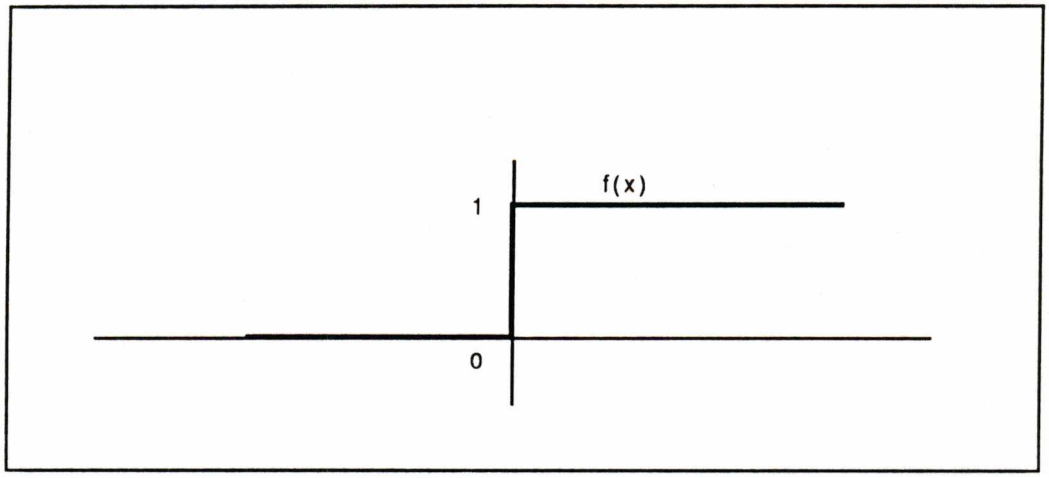


Figure 4.4: Hardlimiting threshold used in the BAM.

The thresholding function used for the BAM may be considered as an extreme case of a sigmoid (Fig. 4.4), given by

$$z_j = \begin{cases} 1 & \text{if } ZM_j > 0 \\ 0 & \text{if } ZM_j \leq 0 \end{cases} \tag{4.11}$$

Z^k in Eq. (4.8) is the BAM output at iteration k. This output is sent to the thresholding function (Eq. 4.11) which results in a binary output. The output is again presented to the input of the BAM

$$\begin{aligned} Z &\rightarrow M \rightarrow Z^1 \\ Z^1 &\rightarrow M \rightarrow Z^2 \\ Z^2 &\rightarrow M \rightarrow Z^3 \\ &\vdots \end{aligned}$$

$$Z^{k-1} \rightarrow M \rightarrow Z^k.$$

This process continues until the BAM converges such that

$$Z^k = Z^{k-1}. \tag{4.12}$$

The convergence condition (Eq. 4.12) is always reached in a few iterations.

A program written in Fortran is used to implement the BAM and the various coding schemes. Typical representations of each of the two classes (Fig. 4.2) were used to encode the BAM. Once the learning stage was completed the BAM was tested for stability by using the learning patterns themselves as test patterns. The BAM recalled both patterns thus indicating that the associations were stored as stable points. Every association (A_i, B_i) has a corresponding energy level E and is given by

$$E(A_i, B_i) = -A_i M B_i^T \tag{4.13}$$

The BAM energy can be viewed as the strength of association (or similarity) between the two patterns in a pattern pair. Computed energy levels for each of the learning patterns was found to be -2516. Figures 4.5, 4.6, and 4.7 show the results of some of the test patterns presented to the BAM. In each case the energy of the associations were computed as -1156, -1124, and -900 for the test patterns `cel014.psd`, `cel026.psd`, and `cel011.psd` respectively. The energies of the test pattern associations (Z_i, A_j) were always higher then that of the trained associations (A_i, A_j). This indicates that the BAM stored the trained

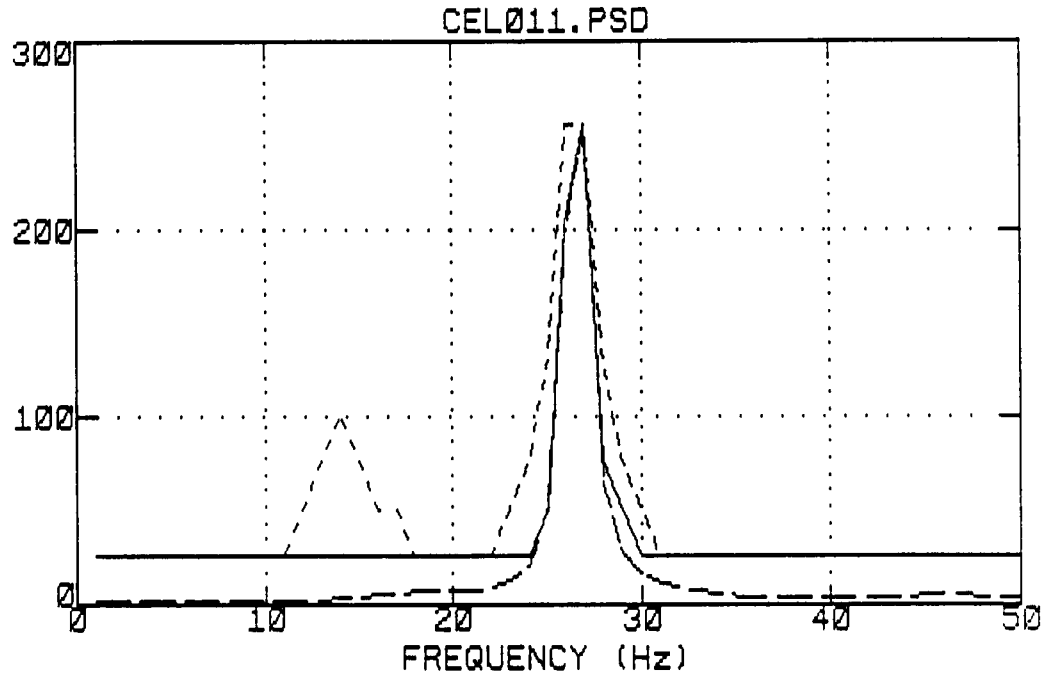


Figure 4.5: Plot showing the scaled pattern(– –), bit-mapped pattern(——), and the BAM recall (- - -) for the test pattern cel011.psd.

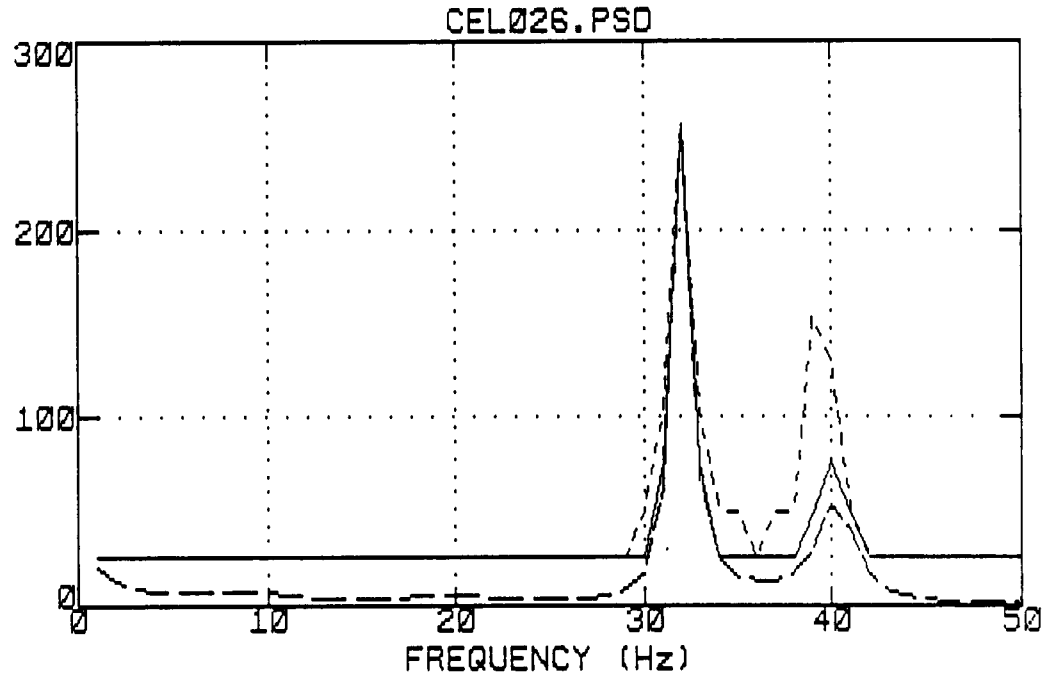


Figure 4.6: Plot showing the scaled pattern(– –), bit-mapped pattern(——), and the BAM recall (- - -) for the test pattern cel026.psd.

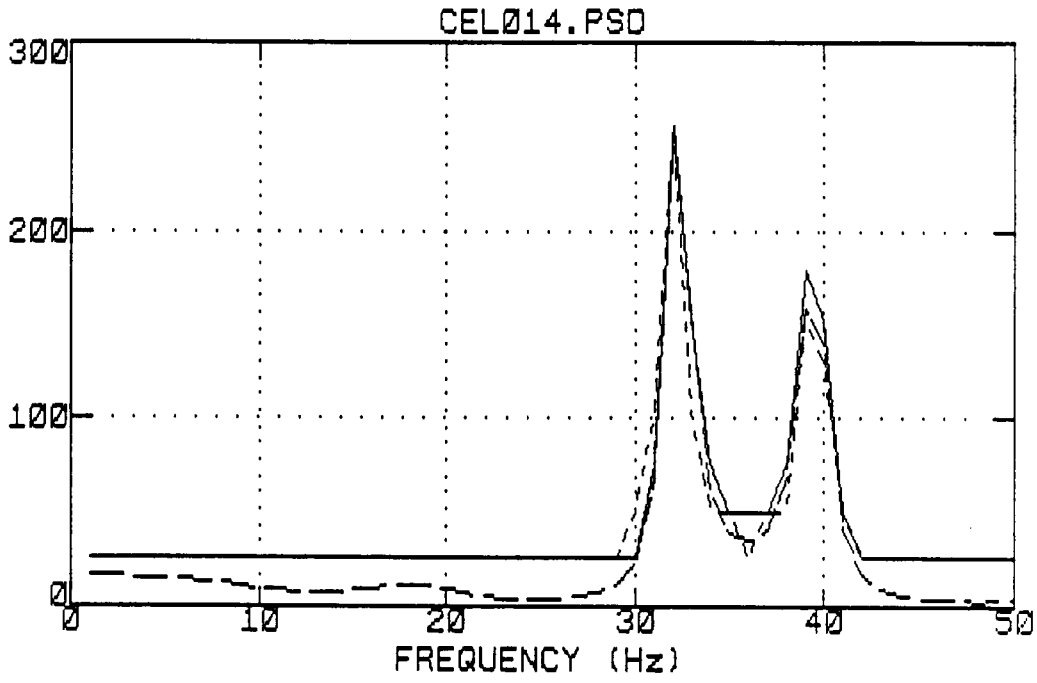


Figure 4.7: Plot showing the scaled pattern(– –), bit-mapped pattern(——), and the BAM recall (- - -) for the test pattern cel014.psd.

patterns at local energy minima. In all the cases the BAM correctly classified the test patterns Z_i by bringing out a learned association A_i that characterizes the class to which the test pattern belongs. The resolution setting for bit-mapping is $r = 10$ for these examples. Increasing the resolution much beyond this level resulted in some misclassifications due to the dominance of the low level signals over the peaks in the PSD. While the peaks correspond to class 1 (which is the correct class) the low amplitude components correspond more to class 2. From the point of view of the BAM, all the frequency points have equal weighting and the low amplitude points outnumber the peaks, hence it misclassifies the pattern as belonging to class 2.

It must be emphasized that the coding scheme used to transform the patterns to binary vectors must be capable of nullifying the effect of the irrelevant information (in this case, the low amplitude points) and stress the important information (PSD peaks). Reducing the resolution of bit-mapping effectively solves this problem by bringing all the low amplitude points to a single amplitude level. The correct classification as a result of this is shown in Fig. 4.5. In Figures 4.5, 4.6, and 4.7 the long dashed curve is the scaled PSD and the solid line is a result of bit-mapping the PSD with $r = 10$. The BAM recall is the learned characteristic pattern of the class, shown by the uniformly dashed curve. Figure 4.5 shows the correct classification of the pattern to class 1 in spite of a peak (at 14 Hz) being absent in the pattern. This demonstrates the fault tolerance and the error correcting properties of the trained network. Fig-

ures 4.6 and 4.7 show two test patterns that belong to class 2 and correctly bring out the exemplar pattern of that class. Figure 4.8 shows the energy levels in the BAM state space by a simulated test pattern with a single sharp peak of amplitude equal to the highest peak in each of the learning patterns. The simulated test pattern is presented repeatedly to the BAM, and each time the peak is shifted by 1 Hz until the entire frequency range is covered. The plot of the energies showed an interesting relationship with the location of the peaks in the bit-mapped learning patterns (Fig. 4.8). Each time the simulated test pattern coincided with a peak in either of the learning patterns the energy showed a sharp decrease. The minimum energies at 26-27 Hz and 32 Hz occur when the amplitude *and* the frequency of the test pattern coincide with class 1 and class 2 respectively. A local energy minimum at 30 Hz is due to the overlap of the two characteristic patterns at that frequency. Energy of association of the test pattern is lower with class 2 at -1180 than with class 1 at -1160. So the BAM exhibits a tendency to bring out class 2 over class 1 when presented with an arbitrary pattern that does not match with either class.

4.3 Description of Software

This section explains the usage of software written to implement this algorithm. The main program is contained in one file which works in conjunction with several other support files. The program was designed to be flexible so as

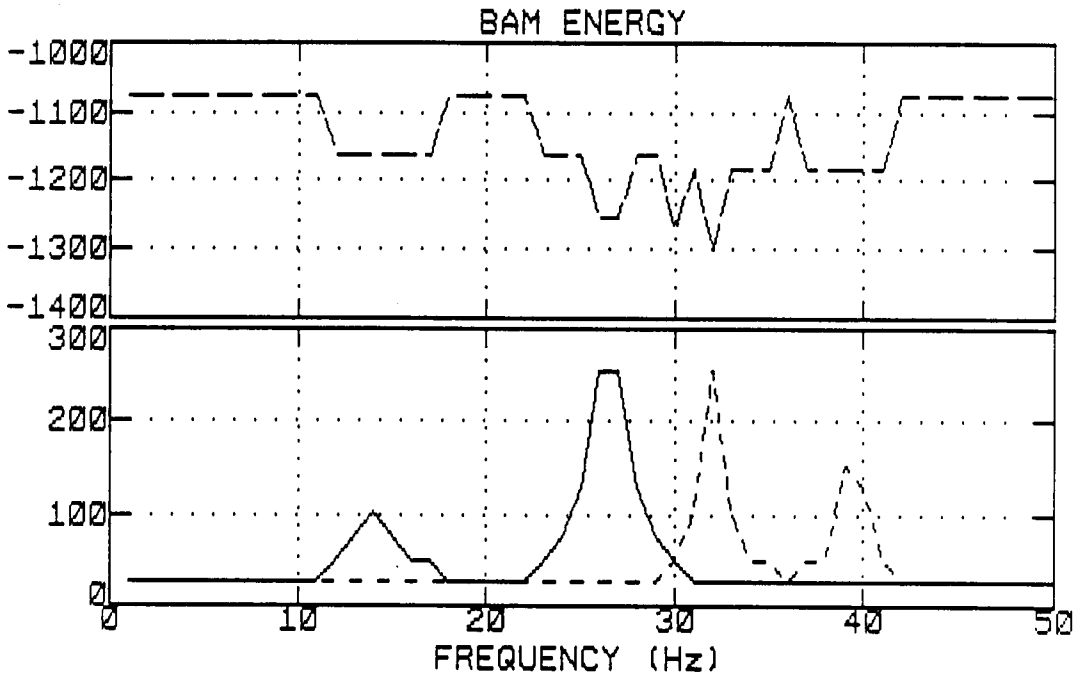


Figure 4.8: The energy of association between the stored patterns and a simulated test pattern with a single peak that is scanned across the entire frequency range. Note the correspondence of the bit-mapped training patterns (bottom plot) to the energy.

to support many different applications. It is menu-driven for ease of use and has extensive error reporting capability. The entire software was implemented in the DEC 2000 workstation. Most network specifications are given in the *parameter specification file* which has the format

```
NUMBER OF LEARNING PATTERNS : 2
NUMBER OF POINTS IN PATTERN : 50
NUMBER OF TEST PATTERNS : 4
PLOT OUTPUTS - YES OR NO (1/2) : 1
SCALE PATTERNS - YES OR NO (1/2) : 1
THRESHOLD : 0.0
FILENAME OF LEARNING PATTERNS : cel_trn.psd
FILENAME OF TEST PATTERNS : cel_tst.psd
```

This file contains all the parameters to determine the size of the network, and provide the names of the files containing the data. The data files are of direct access type with a record length equal to four bytes. The training patterns are concatenated in the data file, while the test data can be structured as the training data or be stored in separate files. In the latter case the `Number of patterns` parameter is set to one and alternative file names can be successively supplied by suitable menu options within the program environment. This feature is useful in specifying different patterns contained in separate files. Many error trap routines are provided in the software to prevent errors that occur within

the program environment from developing into fatal system errors. For example if the file name supplied does not exist, the program opens an *error window* that signal the message,

```
BAM Error :  FILE NOT FOUND
```

and requests a new file name.

Within the program environment all options are specified as menus from which options can be selected using a mouse. The menu hierarchy is the following

1. **Training the BAM**

2. **Run the BAM**

   (a) **Load BAM matrix only**

   (b) **Run BAM without loading**

   (c) **Load BAM matrix and run**

   (d) **Specify new test filename**

   (e) **Exit this menu**

3. **Exit this menu**

Loading the BAM matrix from the disk takes about five minutes, so item (a) in the run environment allows the matrix to be loaded and then be resident in the RAM. Test patterns can be repeatedly applied by specifying item (d) followed by item (b) in the run environment.

CHAPTER 5

Multilayer Perceptron Network

5.1 Introduction

This chapter discusses the application of the multilayer perceptron network to classification of power spectral density (PSD) functions and prediction of process signals (signal validation). The general multilayer perceptron consists of three or more layers of processing elements, including the input and the output layer (Fig. 5.1). The network is fully connected between layers. The first layer serves only as a buffer for the input vectors and contains no summing nodes.

The method used to train the network is the Backpropagation (BP) algorithm [3]. The BP algorithm is trained off-line and requires multiple sets of input and target output vectors. The algorithm makes connection weight adjustments according to the error between the target outputs and the actual outputs. The local error at the output is propagated back to the previous layers so as to minimize the global error.

The BP algorithm is capable of training a well-configured network to create arbitrary mappings from the input to the output. This property is exploited by using the network as a classifier for spectral signatures and as a predictive

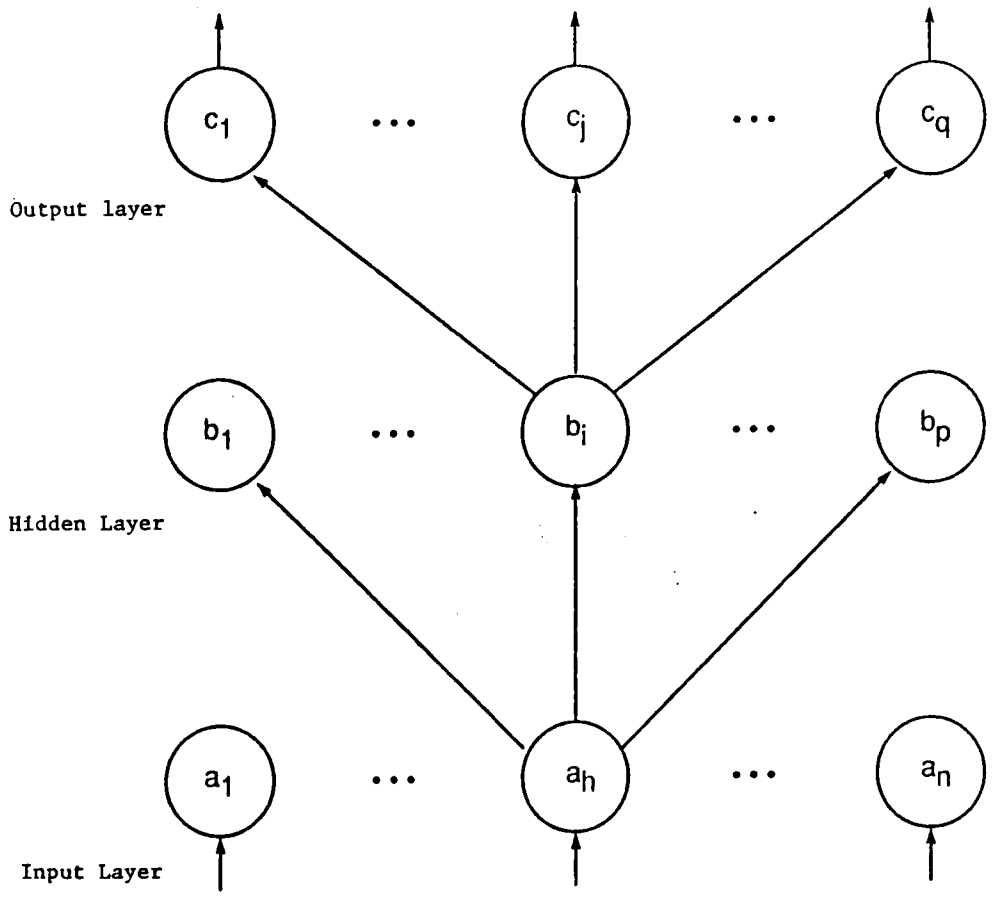


Figure 5.1: Topology of a three layer Backpropagation network

model for validating sensor signals. Later sections discuss these applications in detail.

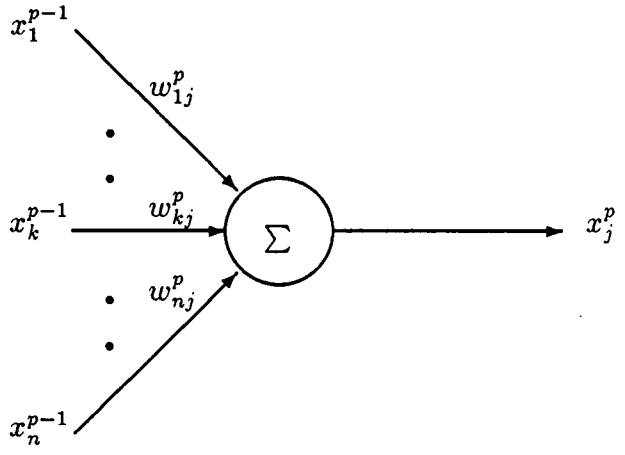


Figure 5.2: The processing element in the backpropagation algorithm.

5.2 Overview of the Backpropagation Algorithm

5.2.1 The Processing Element

The Processing Element (PE) is the basic building block of the network (Fig. 5.2). It sums up the product of the inputs and the connection weights from the previous layer of PE's and then thresholds it by a nonlinear thresholding function. The output of a PE in layer p forms an input to the PE's in layer

$p+1$. The value of the PE j in layer p is given by,

$$x_j^p = F\left(\sum_i w_{ij}^p x_i^{p-1} + \theta_j^p\right) \tag{5.1}$$

and the thresholding function $F(x)$ is given by

$$F(x) = \frac{1}{(1+e^{-\beta x})}, \qquad \beta > 0 \tag{5.2}$$

where x_j^p is the value of processing element j in layer p and w_{ij}^p is the connection weight from PE i in layer $p-1$ to PE j in layer p. θ_j^p is the bias associated with PE j in layer p.

5.2.2 The Generalized Delta Rule

The generalized delta rule allows a gradient descent in the sum squared error at the outputs, with the PE's using nonlinear activation functions. The error in the output layer can be computed easily as the difference between the target output and the actual output. The error for the hidden layers cannot be computed directly as the target outputs for the hidden layer PE's are unknown. The delta rule allows computation of the error at any PE in layer p using the errors of the PE's in layer $p+1$. The parameter that is propagated back from the output layer is defined as,

$$\delta_j^p = -\frac{\partial E}{\partial S_j^p} \tag{5.3}$$

where S_j^p is $\sum_i w_{ij}^p x_i^{p-1}$ and E is the sum squared error at the output layer given by

$$E = \frac{1}{2}\sum_j (t_j - a_j)^2 \tag{5.4}$$

where t_j is the target value and a_j is the actual value at the output layer.

Applying the chain rule to Eq. (5.3) and breaking the partial derivative into two factors

$$\delta_j^p = \frac{\partial E}{\partial x_j^p}\frac{\partial x_j^p}{\partial S_j^p}. \tag{5.5}$$

But

$$\frac{\partial x_j^p}{\partial S_j^p} = f\prime(S_j^p) \tag{5.6}$$

and

$$\frac{\partial E}{\partial x_j^p} = -(t_j - a_j). \tag{5.7}$$

Hence for the output layer

$$\delta_j^p = f\prime(S_j^p)(t_j - a_j) \tag{5.8}$$

and for the hidden layers,

$$\delta_j^p = f\prime(S_j^p)\sum_i \delta_i^{p+1} w_{ij}^{p+1} \tag{5.9}$$

$f\prime(x)$ depends on the activating function. For a sigmoid,

$$f\prime(x) = f(x)[1 - f(x)] \tag{5.10}$$

Thus the local error at each node can be computed. The connection weights are adjusted to minimize the global error. This is done by using the gradient descent rule and the change in weights is given by

$$\Delta w_{ij}^{p} = -\alpha \left(\frac{\partial E}{\partial w_{ij}^{p}} \right). \tag{5.11}$$

Using the chain rule again and splitting the partial derivative into two factors

$$\frac{\partial E}{\partial w_{ij}^{p}} = \frac{\partial E}{\partial S_{j}^{p}} \frac{\partial S_{j}^{p}}{\partial w_{ij}^{p}}. \tag{5.12}$$

From Eq. (5.11) and Eq. (5.3) we get

$$\Delta w_{ij}^{p} = \alpha \delta_{j}^{p} x_{i}^{p-1} \tag{5.13}$$

where α is defined as the learning rate and specifies the length of the adjustment made in the weights at each pass.

5.2.3 BP Network algorithm

The algorithm used to implement the network uses the generalized delta rule (Section 5.2.2) for training. Figure 5.1 shows the topology of the network. The algorithm [17] is outlined below.

1. Assign a random value r in the range $[+1, -1]$ to all the connection weights w_{ij}^{p} , and bias θ_{j}^{p} to all PE's.

2. Present the normalized input vector k to the first layer of the network and propagate it to the output layer as,

$$x_j^p = \frac{1}{1 + e^{-(\sum_i w_{ij}^p x_i^{p-1} + \theta_j^p)}} \tag{5.14}$$

after which each PE in the network will have an associated value x_j^p.

3. For each PE at the output layer the local errors between the desired value and the actual value is computed using,

$$\delta_j^p = x_j^p(1 - x_j^p)(t_j - x_j^p) \tag{5.15}$$

where p is the output layer.

4. For each PE in the hidden layers, starting at the layer below the output layer and ending at the layer above the input layer, the local errors are found using

$$\delta_j^p = x_j^p(1 - x_j^p) \sum_i \delta_i^{p+1} w_{ji}^{p+1} \tag{5.16}$$

5. Compute all the connection weight corrections as,

$$\Delta w_{ij}^p = \alpha \delta_j^p x_i^{p-1} \tag{5.17}$$

and the bias corrections as

$$\Delta \theta_j^p = \alpha \delta_j^p \tag{5.18}$$

6. Update all the connection weights by adding the weight corrections to the old weights.

7. Update all the PE bias values by adding the bias corrections to the previous bias values.

8. Repeat (step 2) until the error between the desired and the actual value of the output is sufficiently small.

5.3 Discussion of Results

5.3.1 Classification of Power Spectral Density functions

This application uses the network as a classifier, for power spectral density (PSD) functions from the Celanese fiber plant. The data used for training the BAM (Chapter 4) is also used to train the Backpropagation network. Unlike the BAM, which requires binary data, the BP network can accept analog data directly. A nonlinear scaling function is used on the PSD to enhance the peaks. The scaling function also normalizes the pattern which is a necessary form required by the BP algorithm.

The criterion used for discriminating the PSD's is the frequency of occurrence of the peaks. In order to highlight this information and to reduce the effect of low-level information a bit-mapping procedure, similar to the one described in Chapter 4, is used. The Y dimension is partitioned into r windows and each discrete frequency point falls within a window. This results in reducing the effect of low level points in the PSD, since most of the low level signals fall within the lowest window and are thus assigned a fixed value. Figure 5.3 shows the training

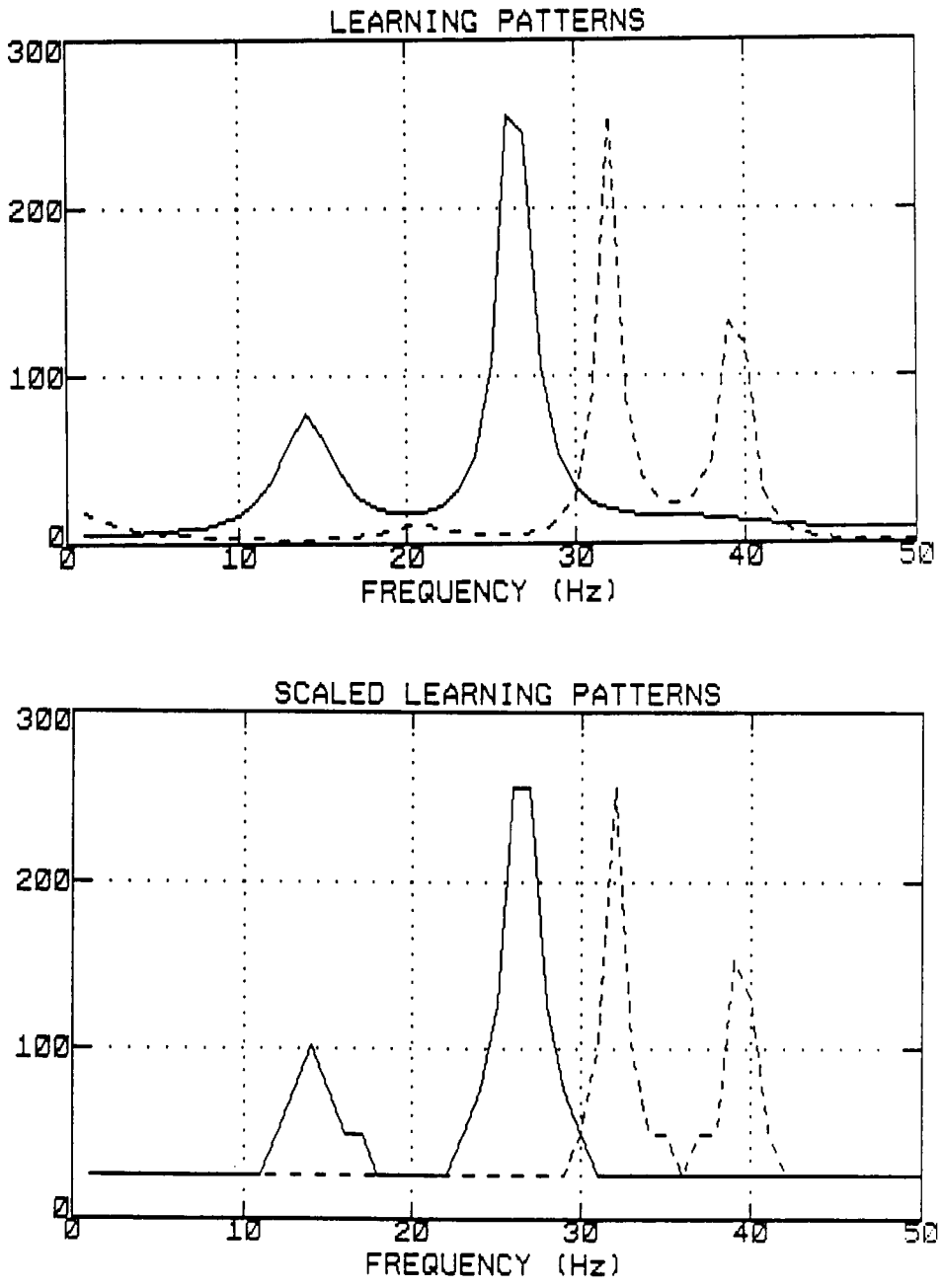


Figure 5.3: The patterns used to train the network before bit-mapping (top) and after bit-mapping (bottom).

patterns before and after bit-mapping.

Unlike the BAM, which produces the characteristic pattern of class to which the test pattern is assigned, the BP network is structured to produce a binary two bit code signifying the class membership. This structure has the significant advantage of reducing the size of the network, which in turn translates to savings in computer memory. A three layer network is used for training. The input layer has 50 PE's to match the 50 discrete points in the PSD. The hidden layer has 15 PE's and the output layer has 2 PE's. The output is coded so as to produce an output of [1 0] signifying the first class and [0 1] signifying the second class.

The BP algorithm almost never achieves target values in the extremities of its operating range of 0 to 1. The problem is normally overcome by setting the targets as [0.1 0.9] instead of [0 1]. But we have found that this target can be achieved by *progressive thresholding* which involves changing the value of β (Eq. 5.2) during learning to higher values. Figure 5.4 shows the convergence of the algorithm with the two training patterns. The X-axis represents the number of iterations and the Y axis shows the error between the desired output and the actual output. Our tests show that in the first few iterations the training patterns show a roughly symmetrical behavior around the X dimension. That is, the overall error for one pattern increases while the error for the other pattern decreases. But in a few iterations the connection weights arrange themselves so that the errors for both the training patterns decrease. Progressive thresholding reduces the training time when the target outputs are in the extremities of

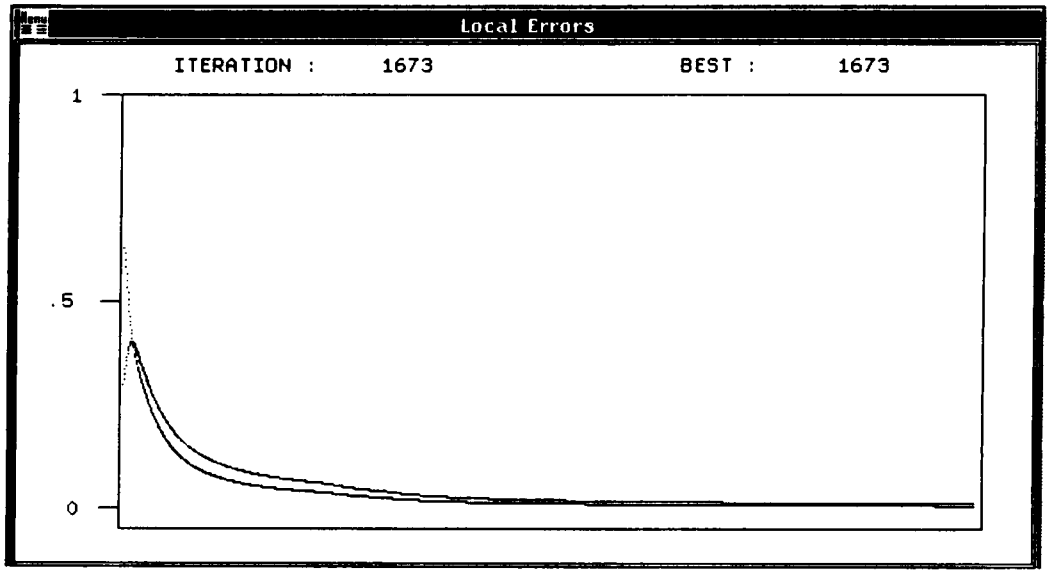


Figure 5.4: The plot showing the convergence of the BP algorithm. The Y-axis represents the error between the desired and the actual outputs of the network and the X-axis represents the iteration number. Each curve corresponds to a training pattern.

the BPN operating range. Figure 5.5 shows the convergence of the algorithm with progressive thresholding. The sudden reductions in the error correspond to changing β to higher values. This method allows the BPN to reach binary targets outputs.

On completion of the training the resultant network was tested with patterns not used for training. Figures 5.6 and 5.7 show the network outputs graphically. The rectangles represent PE's and the shading inside the rectangles represent their current values with the dark shade signifying a 1 and the light shade a 0. Intermediate values are represented by stripes of different thickness and orientation. Figure 5.6 shows a test pattern belonging to class 1 and the network output of [1 0] and Fig. 5.7 shows a test pattern belonging to class 2 with the network output being [0 1].

An important feature of the BP network is its ability to interpolate and extrapolate. This feature allows quantification of the goodness of classification by looking at the actual network output. For example, if the network produces an output of [0.7 0.2], it indicates that the test pattern is closer to class 1 (ideally [1 0]) but does not match the trained characteristic pattern of that class very well. The BP network correctly classified all the test patterns presented to it.

5.3.2 Application to Signal Validation

The second application of the BP network is to validate steady-state process variables from the ALCOA Hot Line rolling facility. The sensor outputs are

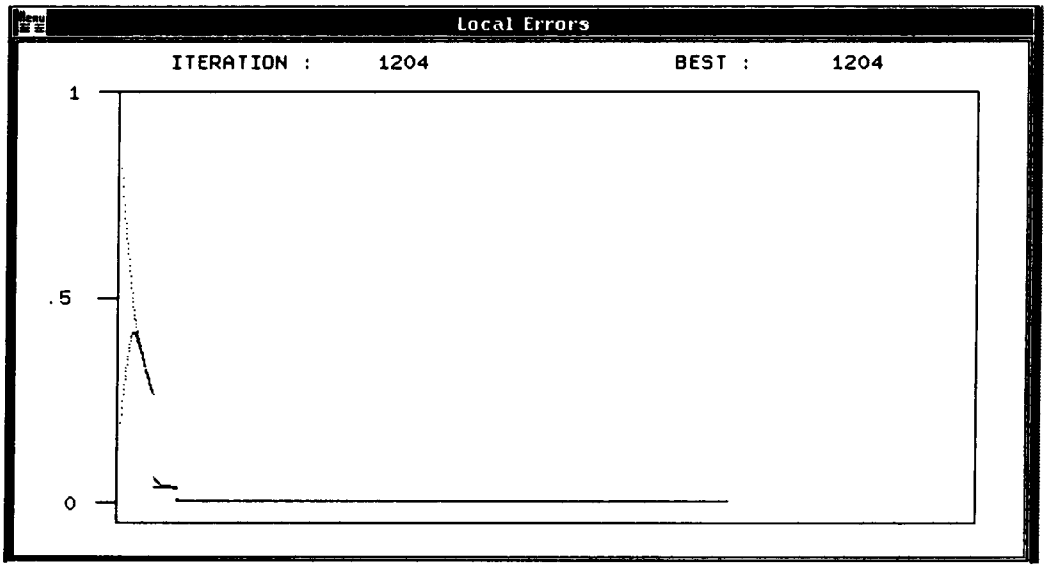


Figure 5.5: The plot showing the convergence of the BP algorithm with *progressive thresholding.*

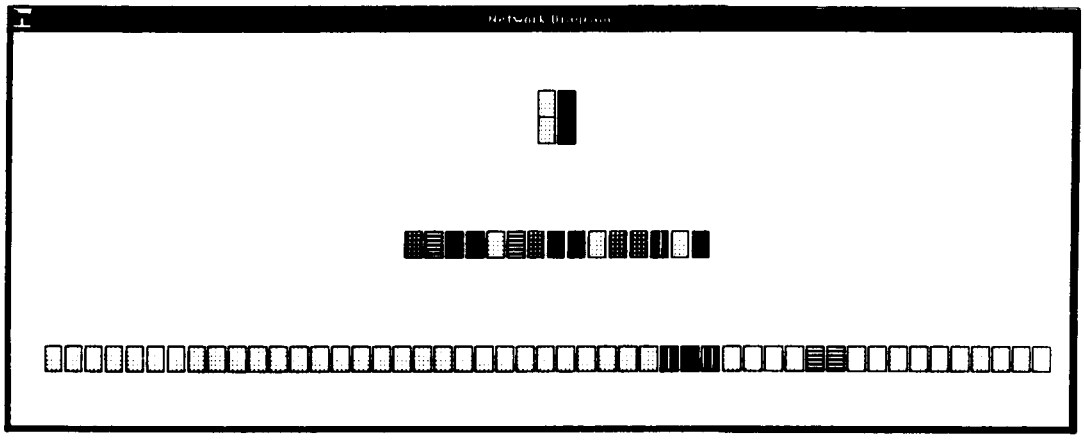


Figure 5.6: The network diagram showing the network output of '0 1' to a test pattern belonging to class 1.

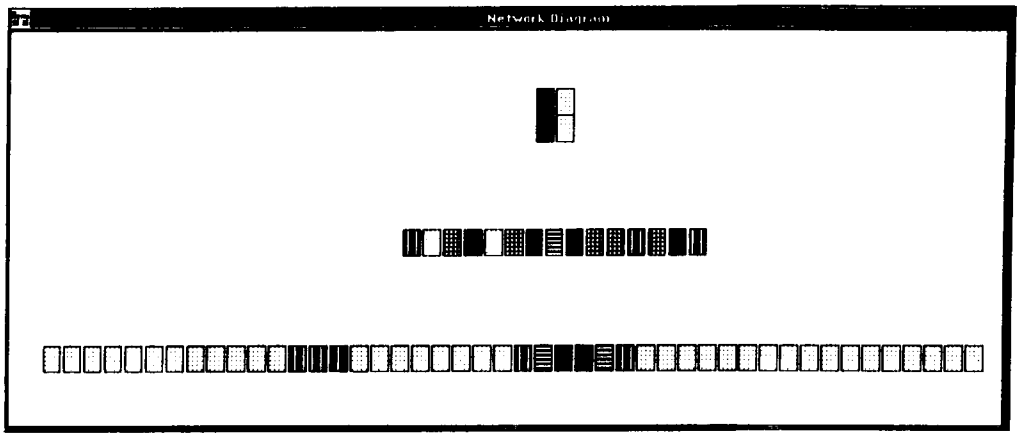


Figure 5.7: The network diagram showing the network output of '1 0' to a test pattern belonging to class 2.

validated by expressing the signal to be validated as a function of other signals. The formulation of the problem remains the same as the one discussed in chapters 2 and 3, with the BP network replacing clustering and empirical modeling. Keeping the same problem structure has the advantage of facilitating an accurate comparison of the two techniques.

The primary emphasis has been in training networks for predicting interstand 4-5 tension as a function of

- stand 4 roll speed,
- stand 5 roll speed,
- stand 4 roll force, and
- stand 5 roll force.

The selection of these signals is motivated by the physics of the system and by the extensive trials conducted with different combinations of signals prior to the development of the empirical models, discussed in Chapter 2.

Scaling of Input Signals

The BP algorithm uses an exponential scaling function (Eq. 5.2) as a threshold which causes the output of any PE to saturate at a value of 1. In order to utilize the active range of the exponential curve all the input signals must be normalized to lie within a range of $[0 \rightarrow 1]$. However, since the network cannot produce an output outside the $[0 \rightarrow 1]$ range, and since the application of signal

prediction dictates that the network should be able to extrapolate outside its learned domain, scaling is applied so that all the training patterns lie within the range $[0.2 \rightarrow 0.8]$. Thus any network output outside this range will represent an excursion from the trained domain, and the network is able to extrapolate beyond the domain of the training data.

Configuration of the Network

The network configuration has a direct effect on the behavior of the network. Two major factors that govern the network characteristics are, the number of hidden layers and the number of PE's in each hidden layer. Rumelhart [3] has shown that the number of presentations required (for an XOR test case) is given by

$$P = 280 - 33 \log_2 H \tag{5.19}$$

where P is the number of presentations required and H is the number of hidden PE's.

In general, increasing the number of PE's in the hidden layer causes the network to converge faster during training. Extensive testing has shown that if the number of training patterns is low then setting the number of hidden PE's equal to the number of training patterns results in good convergence. However, increasing the number of hidden PE's results in higher usage of computer memory and more CPU time per iteration.

A minimum of three layers are required to generate nonlinear decision bound-

aries to separate data which are not linearly separable. This makes at least one hidden layer necessary. It is generally accepted that a maximum of five layers are sufficient to separate patterns of arbitrary complexity. In the application described here three layers are found to be sufficient to give good results. The number of input nodes is dependent on the length of the input vector. Since four signals are used as input, the input layer has four nodes and the output layer has one node corresponding to the signal being predicted.

Training Methodology

The rate of convergence of the algorithm is dependent on many factors. Two major factors that govern learning are α (Eq. (5.13)) and β (Eq. (5.2)). While there is no way of determining apriori the best settings for α and β, changing these settings adaptively has given good results. The *BPN Development Module* (discussed in section 5.4) performs real time plotting of the error between the actual and the desired outputs, which allows the user to actively change α and β to facilitate faster convergence and observe its effect on the overall error. While increasing α in general causes the algorithm to converge faster, it may miss the best solution since α defines the length of the linear steps taken during gradient descent. We have found that starting with a low value of α and gradually increasing it during training yields good results.

The number of training cycles required is determined by the overall error after each cycle. When the overall error reaches an acceptable level (or a prede-

termined level) the training is stopped and the network is saved. Training the network beyond this only causes the connection weights to increase in magnitude while yielding no significant improvement in overall error.

Testing of Trained Networks

The determination of the termination point of the training based on the overall error within the training domain, may not always provide good results. The network may learn an incorrect mapping between the inputs and the output which would still produce an acceptable overall error but may not perform well with test patterns not previously used during training. This is overcome by testing the network with new patterns *while* training and evaluating its performance. The *BPN development module* allows the training to be interrupted and a new test set to be applied to the current network. The training is resumed if the results with the test set is unacceptable.

Figure 5.8 shows the results of application of the network for predicting interstand 4-5 tension after 100 cycles. At 50 cycles the network produces an average value of the training domain, but it improves progressively till around 100 cycles and shows no significant improvement on further training. The network is trained on the first 20 points and all the points beyond that are unused test patterns.

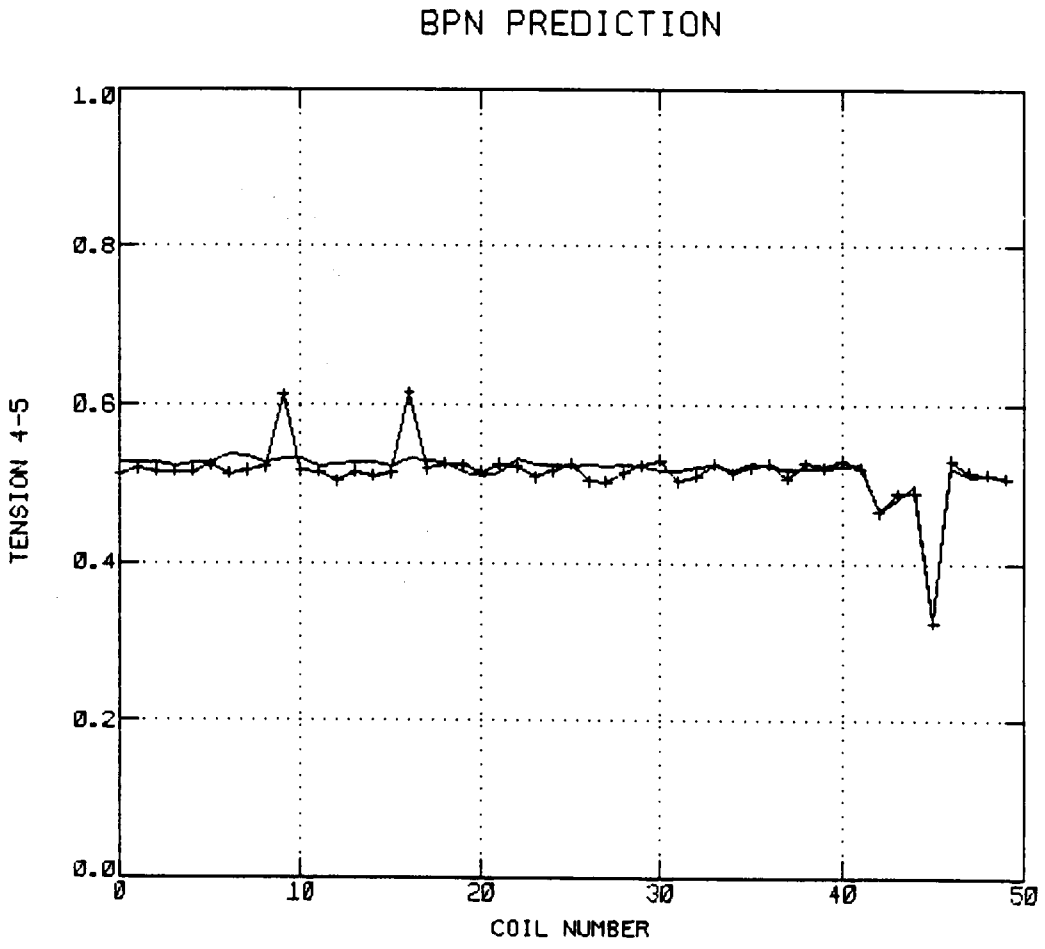


Figure 5.8: BP network prediction of interstand tension 4-5 after 100 training cycles

5.4 Description of Software

The *BPN development module* is written in VAX FORTRAN and implemented on a DEC 2000 workstation. The module is written to train networks by providing easy operator interface to influence the training. It is a complete development environment with pull down menus, mouse control, real-time plotting, and asynchronous interrupts to provide easy control over the training.

The program employs a data pointer system to store arrays efficiently. All the connection weights are stored in a 1-dimensional array and employs a pointer to simulate a 3-dimensional array. This structure allows efficient storage of the weight matrix even if the number of nodes in each layer is different from the others.

The structure of the network and all the other input parameters are provided in a file called the `bpnmode.dat`. The format of this file is shown in Fig. 5.9. The *Network Specifications* category defines the size and shape of the network. The network is constructed by specifying the number of layers and the number of nodes in each layer. The *training parameters* category contains the following specifications:

Number of training patterns defines the size of the training set. The BPN module allows a maximum of 200 training patterns.

Number of training iterations limits the maximum number of training cycles to a specified value.

```
+++++++++++++++++++++ PARAMETER DEFINITION FILE +++++++++++++++++++++++++++++++++
***** DO NOT CHANGE THE STARTING FIELDS OF THE SPECIFICATIONS ***************
--------------------- NETWORK SPECIFICATIONS ---------------------------------
 NUMBER OF LAYERS                  :3
 NODES/LAYER(STARTING AT I/P):-
 LAYER 1 :4
 LAYER 2 :60
 LAYER 3 :1
 LAYER 4 :0
 LAYER 5 :0
--------------------- TRAINING  PARAMETERS -----------------------------------
   NUMBER OF TRAINING PATTERNS    :40
 NUMBER OF TRAINING ITERATIONS    :100000000
          LEARNING COEFFICIENT    :.7
          INPUT PATTERN LENGTH    :4
         OUTPUT PATTERN LENGTH    :1
   FREQUENCY OF DISPLAY UPDATE    :300
   TYPE OF DISPLAY [1(N)/2(P)]    :3
  TRANSFER FUNCTION SHAPE[1-5]    :1.0
           DATA SCALING FACTOR    :1.0
------------------- FILE SPECIFICATIONS -------------------------------------
 INPUT FILE TYPE OPTION [1/2]     :2
 [1]FILENAME OF TRAINING PATTERN :8,9
 [2]FILENAME OF TRAINING PATTERN :tension.dat
```

Figure 5.9: The parameter definition file that supplies the network specifications to the BPN module

Learning coefficient The starting value of α is given here. This number can be changed during training.

Input pattern length specifies the length of the input pattern vectors. This value must be the same as the number of input nodes.

Output pattern length specifies the length of the desired output pattern. This value must be same as the number of nodes in the output layer.

Frequency of display update specifies the number of cycles between each display update. It has effect only when the network diagram (next item) is chosen as the display option.

Type of display allows the choice of three different display options. '1' specifies the network diagram, '2' the plot of the error between the desired and the actual outputs and '3' deactivates all displays. Mode 3 is the fastest mode. This option can be changed during training.

Transfer function shape specifies the starting value of β. It can be changed during training.

Data scaling factor All the input pattern vectors are divided by this number before being sent to the input layer. It allows internal scaling of the training patterns.

The final category in the parameter file is the *file specification.* There are two options allowed for the input data files. Option 1 requires each training pattern

to be in a separate file. An integer number is used to specify the filename, for example an '8' implies a file named `bpnlrn.008`. This type of file structuring is useful because it allows the ordering of the input patterns to be changed easily. Option 2 is provided for accommodating a large number of training patterns since it allows all the patterns to be in a single sequential file. The name of the input file is provided in the input field.

A *control panel* (Fig. 5.10) is used to modify the network parameters during execution. It is entirely mouse-driven and specific functions are selected by pointing the mouse to a box and clicking one of the three buttons provided on the mouse. The specific functions of each box are as follows.

Restart Clicking the left button of the mouse causes the training to restart from the beginning.

Adjust training rate Clicking the left button of the mouse causes a menu to appear with the various choices of training rates (Fig. 5.11).

Change shape Refers to changing the shape of the thresholding function by varying β. Clicking the middle button of the mouse makes a menu (Fig 5.11) appear. The last item in the menu changes the sign of β.

Reset plot It is used to clear the plotting window. Valid only when the display option is '2' (see next item).

Display Option Allows changing the display option among the network diagram, the error plot, and deactivation of all displays by clicking the left,

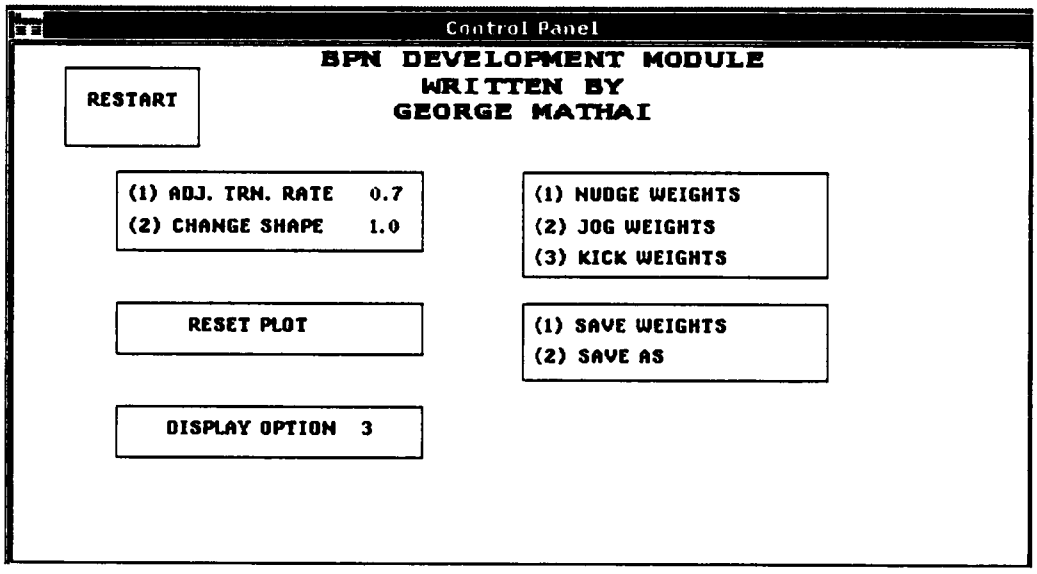


Figure 5.10: The control panel which is used in conjunction with a mouse to adjust training parameters.

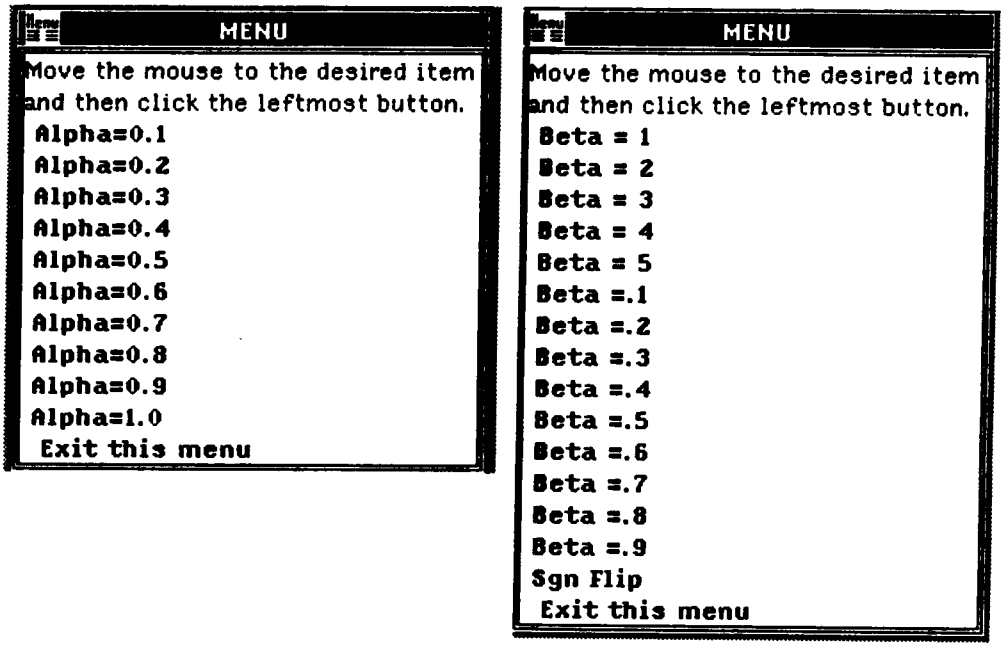


Figure 5.11: The menu on the left allows selection of various training rates and the menu on the right permits the selection of β.

middle and right button respectively.

Adjust weights Is useful in pulling the network out of a local error minima by adding a small random number to each connection weight. Nudge, jog and kick yield increasing magnitudes of random numbers.

Save Weights Causes a snapshot of the weights and bias to be written in a file called `bpnop.dat`.

Save as allows a data entry window to appear permitting a filename to be specified by the user. The connection weights and bias are then saved in the specified file.

CHAPTER 6

Summary and Conclusions

6.1 Summary

A study of different techniques applied to signal validation and process state classification is presented. Nonlinear empirical prediction models are developed for one or more critical signals as a function of other process variables in the system. The accuracy of the predictive models is enhanced by clustering the baseline data into different process states and representing each process state by a predictive model. This is due to the differing relationships between process variables from one process state to another, thus making a single mapping between different process variables inaccurate. It is effectively taken care of by characterizing different process states by their corresponding predictive models. The clusters are generated using the K-Means algorithm, the Single Linkage Cluster Analysis Algorithm (SLCA) and manually based on alloy type. The K-Means algorithm is sensitive to the ordering of the patterns and the number of clusters generated. Results from the application of SLCA is superior to the K-Means algorithm. The noise sensitivity of the SLCA is used advantageously to remove noisy patterns from the baseline data. Hierarchical clustering is used,

involving repeated application of the SLCA, to produce compact clusters of uniform membership. The final clusters are represented in a data base along with their corresponding predictive models. The data base is coordinated by the Signal Validation Module (SVM). The SVM is designed to coordinate the data base and to validate instrument outputs in an automated fashion by classifying the test patterns and applying the right model to generate a prediction.

The Bidirectional Associative Memory (BAM) network allows mapping between any two binary patterns. Multiple sets of associations can be superimposed in the associative memory. The BAM is used as a classifier of Power Spectral Density (PSD) functions. Its fault tolerance and error correcting properties make it capable of generating a successful recall even with incomplete or noisy patterns. It is a two-layer feedback network of interconnected nodes and its bidirectional property allows operation from input to output or vice-versa.

A detailed study of the performance of the BAM to classification of PSD functions is presented. The numerical aspects of signature encoding and the fault tolerant property of the BAM are evaluated. A bit-mapping scheme that enhances the discriminating features of the patterns and nullifies the irrelevant information is presented. The BAM energy function is used as a systematic method of rating its performance.

The BackPropagation Network (BPN) is used for signal validation and as a classifier for PSD functions. The BPN is a fully connected network containing three or more layers of neurons. It utilizes the generalized delta rule during

training to adjust the connection weights and processing element biases. Its flexibility and ability to learn arbitrary mapping makes it suitable for a wide variety of applications. The BPN trains off-line in a supervised mode with inputs and target outputs. Among its limitations are its long training time, sensitivity to internal parameters, and its little understood convergence conditions. But its intrinsic parallelism makes it very fault tolerant.

6.2 Conclusions

The SVM successfully detected transient anomalies in the interstand 4-5 tension and predicted its value in the same range as the other readings. Since the signal recovered in subsequent measurements, it was not declared as a sensor failure but as a transient fault or data acquisition error. The within-domain model prediction error for manual clustering is in the range of 3-5 % and for automated clustering is below 1 %. This represents a significant improvement in the prediction accuracy. A comparison of SVM and the manual clustering schemes with new test patterns showed the ability of the SVM to make accurate predictions outside the trained domain, while the performance of prediction models characterizing manual clusters tended to deteriorate sharply outside the trained domain.

Database clustering and empirical predictive models have proven to be efficient techniques in sensor fault monitoring. These methods can be applied for

on-line diagnosis in process industry. Its ability to validate nonredundant signals and process large amounts of data provides a valuable operator tool.

The results of the BAM are superior to classical pattern recognition methods. The trained BAM successfully classified all the test patterns. It performed successful recall even with very noisy test patterns exhibiting its robustness. The recall time is independent of the number of training patterns, and thus is very fast. One of the drawbacks of the BAM is that it requires binary pattern vectors. It is also expensive in terms of computer memory because of the large correlation matrices involved.

The BPN showed excellent performance as a classifier. A method called progressive thresholding is developed, which resulted in fast convergence of the algorithm during training. The network successfully classified all the test patterns presented to it. Unlike the BAM, the BPN can operate with analog patterns making it faster and more efficient after the training is completed. In terms of classification accuracy, both the BAM and BPN exhibited comparable performance. While the BAM assigns test patterns to the nearest known class (nearest neighbor), the BPN has the ability to interpolate and extrapolate from known states. This ability of the BPN also allows quantification of the closeness of the test pattern from the known states.

Training the BPN for signal validation takes a long time and requires considerable heuristics. But a well-trained network showed excellant performance accuracy. In terms of robustness and accuracy the BPN is superior to empirical

modeling and clustering. The BPN is also capable of handling a higher number of training patterns compared to the empirical modeling algorithm. The speed after the training period is comparable for both methods.

In conclusion, each of the techniques presented in this document has proven its usefulness in applications within the framework of signal validation.

6.3 Recommendations for Future Work

The SVM concept can be extended to validate signals from other industrial processes. Other neural network paradigms can be applied to signal validation for a comparative evaluation of performance. Multiple network configurations, which can be homogenous or heterogenous, should be implemented in a structured hierarchical format to improve prediction accuracy and to exploit the salient features of different paradigms. Further research is necessary to fully understand the response of the BPN to changes in threshold functions, learning rate, and the number of hidden nodes.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] G. Mathai and B. R. Upadhyaya, "Performance Analysis and Application of the Bidirectional Associative Memory to Industrial Spectral Signatures," *Proceedings of the Third IEEE International Conference on Neural Networks*, June 1989.

[2] B. Kosko, "Bidirectional Associative Memories," *IEEE Trans. on Man, Systems and Cybernitics*, vol. 18, pp. 49–59, February 1988.

[3] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing.* Cambridge, Mass: MIT Press, 1986.

[4] W. R. Taber *et al.*, "Recognition of ORCA Calls with a Neural Network," *International Workshop on Fuzzy System Applications*, July 1988.

[5] M. Watson, "A Neural Network Model for Phoneme Recognition Using the Generalized Delta Rule for Connection Strength Modification," *Proceedings of the First IEEE International Conference on Neural Networks*, 1987.

[6] H. Yang and C. Guest, "Performance of Backpropagation for Rotation Invariant Pattern Recognition," *Proceedings of the First IEEE International Conference on Neural Networks*, vol. IV-365, 1987.

[7] R. C. Gonzalez, D. N. Fry, and R. C. Kryter, "Results in Application of Pattern Recognition Methods to Nuclear Reactor Core Surveillance," *IEEE Trans. Nucl. Sci.*, vol. 21, pp. 750–756, 1974.

[8] R. C. Gonzalez and L. C. Howington, "Machine Recognition of Abnormal Behavior in Nuclear Reactors," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-7, NO. 10, pp. 717–727, October 1977.

[9] Z. Frei and B. R. Upadhyaya, "Empirical Modeling Using Steady-State Behavior of Linear and Nonlinear Systems with Application to Signal Validation," *Report prepared by the University of Tennessee for The U.S. Department of Energy*, vol. DOE/NE/37959-11, August 1987.

[10] A. Desrochers and S. Mohseni, "On Determining the Structure of Nonlinear Systems," *Int. J. Control*, vol. 40, pp. 923–938, 1984.

[11] J. C. Tou and R. C. Gonzalez, *Pattern Recognition Principles.* Reading, Mass.: Addison-Wesley, 1974.

[12] J. Gover and G. T. S. Ross, "Minimum Spanning Trees and Single Linkage Cluster Analysis," *Applied Statistics*, vol. 18, 1969.

[13] R. C. Prim, "Shortest Connection Networks and some Generalizations," *Bell Systems Technical Journal*, November 1957.

[14] T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification," *IEEE Trans. Information Theory*, vol. 13, pp. 21–27, January 1967.

[15] T. Kohonen, *Self Organization and Associative Memories.* New York: Springer-Verlag, 1984.

[16] M. Cohen and S. Grossberg, "Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks," *IEEE Trans. on Man, Systems and Cybernitics*, vol. 13, pp. 815–926, 1983.

[17] P. K. Simpson, *A review of Artificial Neural Systems 2.* San Diego, CA: General Dynamics, Electronics Division, 1988.

APPENDIX

An example output of the SLCA program with 46 input patterns. The output includes the scaling factors, cluster centers, variance table, and the cluster memberships.

DIMENTIONALITY OF DATA: 5

SRNO.	MEASUREMENTS				
1	25.910	5.230	6.460	121.169	98.726
2	26.030	5.210	6.450	121.827	98.423
3	25.920	5.240	6.450	121.218	97.491
4	26.000	5.220	6.360	122.028	97.873
5	26.130	5.210	6.470	122.240	97.157
6	26.120	5.210	6.420	123.778	97.393
7	25.670	5.210	6.490	122.456	98.581
8	25.990	5.230	6.490	120.868	100.119
9	26.190	5.180	6.560	120.774	100.519
10	26.200	5.260	6.510	119.555	97.873
11	25.770	5.220	6.410	124.945	98.781
12	26.230	5.190	6.390	122.658	95.295
13	27.080	5.240	6.410	123.019	97.421
14	27.210	5.250	6.470	123.178	101.689
15	25.900	5.230	6.370	123.183	95.516
16	26.070	5.180	6.530	119.442	97.208
17	25.970	5.180	6.430	126.758	98.879
18	25.720	5.180	6.320	122.545	94.142
19	26.180	5.260	6.400	120.474	95.507
20	27.350	5.230	6.420	125.249	95.274
21	28.230	5.190	6.330	124.866	91.981
22	25.770	5.180	6.280	121.585	94.779
23	25.700	5.250	6.450	123.208	94.927
24	25.840	5.240	6.390	128.319	98.510
25	25.820	5.270	6.490	128.237	100.172
26	25.960	5.220	6.410	128.457	97.461
27	26.220	5.210	6.340	129.553	96.691
28	25.800	5.190	6.490	127.694	101.363
29	25.870	5.250	6.470	127.834	95.701
30	25.870	5.230	6.600	126.309	102.370

! 31 !	26.140	5.270	6.390	130.193	94.894 !
! 32 !	26.100	5.240	6.320	126.383	96.203 !
! 33 !	25.800	5.270	6.450	131.032	95.145 !
! 34 !	25.560	5.180	6.460	125.955	100.960 !
! 35 !	26.150	5.200	6.310	125.996	93.981 !
! 36 !	25.770	5.180	6.380	132.920	96.580 !
! 37 !	26.160	5.190	6.350	134.386	98.567 !
! 38 !	25.350	5.240	6.330	130.209	93.347 !
! 39 !	26.320	5.270	6.290	134.805	94.417 !
! 40 !	25.940	5.190	6.240	139.924	94.932 !
! 41 !	25.890	5.240	6.400	134.663	98.749 !
! 42 !	26.080	5.190	6.410	126.872	93.685 !
! 43 !	25.690	5.260	6.320	125.091	93.788 !
! 44 !	26.150	5.220	6.390	124.009	92.983 !
! 45 !	26.010	5.220	6.510	136.187	100.318 !
! 46 !	25.950	5.140	6.530	135.905	103.916 !

PATTERN MEAN

26.082	5.219	6.416	126.260	97.180

SCALING FACTORS

4.84	24.19	19.68	1.00	1.30

SRNO.	SCALED MEASUREMENTS				
! 1 !	125.426	126.518	127.124	121.169	128.268 !
! 2 !	126.007	126.034	126.927	121.827	127.874 !
! 3 !	125.475	126.759	126.927	121.218	126.664 !
! 4 !	125.862	126.276	125.156	122.028	127.160 !
! 5 !	126.491	126.034	127.321	122.240	126.230 !
! 6 !	126.443	126.034	126.337	123.778	126.537 !
! 7 !	124.265	126.034	127.714	122.456	128.080 !
! 8 !	125.814	126.518	127.714	120.868	130.079 !
! 9 !	126.782	125.308	129.092	120.774	130.598 !
! 10 !	126.830	127.243	128.108	119.555	127.160 !

! 11 !	124.749	126.276	126.140	124.945	128.339 !
! 12 !	126.976	125.550	125.746	122.658	123.811 !
! 13 !	131.090	126.759	126.140	123.019	126.573 !
! 14 !	131.720	127.001	127.321	123.178	132.118 !
! 15 !	125.378	126.518	125.353	123.183	124.098 !
! 16 !	126.201	125.308	128.502	119.442	126.296 !
! 17 !	125.717	125.308	126.534	126.758	128.467 !
! 18 !	124.507	125.308	124.369	122.545	122.313 !
! 19 !	126.733	127.243	125.943	120.474	124.086 !
! 20 !	132.397	126.518	126.337	125.249	123.783 !
! 21 !	136.657	125.550	124.566	124.866	119.505 !
! 22 !	124.749	125.308	123.582	121.585	123.140 !
! 23 !	124.410	127.001	126.927	123.208	123.333 !
! 24 !	125.088	126.759	125.746	128.319	127.988 !
! 25 !	124.991	127.485	127.714	128.237	130.147 !
! 26 !	125.668	126.276	126.140	128.457	126.625 !
! 27 !	126.927	126.034	124.763	129.553	125.624 !
! 28 !	124.894	125.550	127.714	127.694	131.695 !
! 29 !	125.233	127.001	127.321	127.834	124.339 !
! 30 !	125.233	126.518	129.879	126.309	133.002 !
! 31 !	126.540	127.485	125.746	130.193	123.290 !
! 32 !	126.346	126.759	124.369	126.383	124.990 !
! 33 !	124.894	127.485	126.927	131.032	123.616 !
! 34 !	123.732	125.308	127.124	125.955	131.171 !
! 35 !	126.588	125.792	124.172	125.996	122.103 !
! 36 !	124.749	125.308	125.550	132.920	125.480 !
! 37 !	126.637	125.550	124.959	134.386	128.062 !
! 38 !	122.716	126.759	124.566	130.209	121.279 !
! 39 !	127.411	127.485	123.779	134.805	122.670 !
! 40 !	125.572	125.550	122.795	139.924	123.338 !
! 41 !	125.330	126.759	125.943	134.663	128.298 !
! 42 !	126.249	125.550	126.140	126.872	121.719 !
! 43 !	124.361	127.243	124.369	125.091	121.853 !
! 44 !	126.588	126.276	125.746	124.009	120.806 !
! 45 !	125.911	126.276	128.108	136.187	130.337 !
! 46 !	125.620	124.340	128.502	135.905	135.011 !

NUMBER OF CLUSTERS DESIRED:20

MINIMUM SPANNING TREE

2 -- 1
3 -- 2
5 -- 3
6 -- 5
7 -- 1
4 -- 2

8	1
9	8
10	3
16	10
11	6
17	11
24	17
26	24
27	26
29	26
15	6
12	15
23	15
18	15
22	18
19	12
25	24
28	25
34	28
31	27
33	31
30	28
43	18
35	43
42	35
44	35
32	35
36	33
37	36
41	37
45	41
38	33
13	6
20	13
46	45
39	31
14	13
40	39
21	20

SORTED NODE PAIRS	DISTANCES
2 -- 1	1.095
22 -- 18	1.511
26 -- 24	1.614
3 -- 2	1.627
5 -- 3	1.717
6 -- 5	1.852
7 -- 1	1.904

4 -- 2	1.941
8 -- 1	1.966
12 -- 15	2.000
23 -- 15	2.058
41 -- 37	2.067
9 -- 8	2.139
33 -- 31	2.216
42 -- 35	2.227
34 -- 28	2.248
16 -- 10	2.248
17 -- 11	2.308
27 -- 26	2.396
24 -- 17	2.406
28 -- 25	2.538
10 -- 3	2.545
18 -- 15	2.604
11 -- 6	2.753
29 -- 26	2.780
19 -- 12	2.795
35 -- 43	2.826
44 -- 35	2.888
15 -- 6	2.939
25 -- 24	3.013
31 -- 27	3.014
30 -- 28	3.060
32 -- 35	3.085
43 -- 18	3.235
45 -- 41	3.426
37 -- 36	3.576
36 -- 33	3.701
20 -- 13	3.816
38 -- 33	4.121
13 -- 6	4.769
46 -- 45	5.090
39 -- 31	5.127
14 -- 13	5.711
40 -- 39	5.895
21 -- 20	6.377

CLUSTER	CLUSTER CENTER					
! 1 !	125.822	126.219	126.916	123.601	127.431	!
! 2 !	125.459	126.155	125.320	122.275	123.463	!
! 3 !	125.983	126.155	125.451	134.524	128.180	!
! 4 !	125.717	127.485	126.337	130.612	123.453	!
! 5 !	126.419	125.671	125.156	126.434	121.911	!

6	124.539	126.114	127.518	127.295	131.004
7	124.361	127.243	124.369	125.091	121.853
8	126.588	126.276	125.746	124.009	120.806
9	125.233	126.518	129.879	126.309	133.002
10	126.346	126.759	124.369	126.383	124.990
11	125.911	126.276	128.108	136.187	130.337
12	124.749	125.308	125.550	132.920	125.480
13	132.397	126.518	126.337	125.249	123.783
14	131.090	126.759	126.140	123.019	126.573
15	122.716	126.759	124.566	130.209	121.279
16	125.620	124.340	128.502	135.905	135.011
17	127.411	127.485	123.779	134.805	122.670
18	131.720	127.001	127.321	123.178	132.118
19	125.572	125.550	122.795	139.924	123.338
20	136.657	125.550	124.566	124.866	119.505

VARIANCE TABLE

1	.5264	.2979	1.2451	10.6152	2.2500
2	1.0752	.6387	1.1807	.9375	.3926
3	.4287	.3662	.2402	.0195	.0137
4	.6768	.0000	.3496	.1777	.0273
5	.0283	.0146	.9668	.1914	.0381
6	.3271	.9502	.0762	.9473	.4102
7	.0000	.0000	.0000	.0000	.0000
8	.0000	.0000	.0000	.0000	.0000
9	.0000	.0000	.0000	.0000	.0000
10	.0000	.0000	.0000	.0000	.0000
11	.0000	.0000	.0000	.0000	.0000
12	.0000	.0000	.0000	.0000	.0000
13	.0000	.0000	.0000	.0000	.0000
14	.0000	.0000	.0000	.0000	.0000
15	.0000	.0000	.0000	.0000	.0000
16	.0000	.0000	.0000	.0000	.0000
17	.0000	.0000	.0000	.0000	.0000
18	.0000	.0000	.0000	.0000	.0000
19	.0000	.0000	.0000	.0000	.0000
20	.0000	.0000	.0000	.0000	.0000

CLUSTERS FORMED

CLUSTER= 1

2	1	3	5	6	7
4	8	9	10	11	16
17	24	26	27	29	

CLUSTER= 2

22	18	15	12	23	19

CLUSTER= 3

41	37

CLUSTER= 4

33	31

CLUSTER= 5

42	35

CLUSTER= 6

34	28	25

CLUSTER= 7

43

CLUSTER= 8

44

CLUSTER= 9

30

CLUSTER=10

32

CLUSTER=11

45

CLUSTER=12

36

CLUSTER=13

20

CLUSTER=14

13

CLUSTER=15

38

CLUSTER=16

46

CLUSTER=17

39

CLUSTER=18

14

CLUSTER=19

40

CLUSTER=20

21

VITA

George Mathai was born on August 7, 1963, in Cochin, India. He graduated from Sardar Patel University, Gujarat, India in December 1985 with a Bachelor of Engineering (Electrical) degree. After graduating he worked as a design engineer, for a period of eight months, at Hindustan Brown Boveri Ltd., Baroda, India. He graduated from the The University of Tennessee, Knoxville, in May 1989 with a Master of Science degree in Electrical Engineering. While at the University of Tennessee, he worked as a research assistant. His major research area was the application of pattern recognition and neural computing to signal validation. The author is a member of Eta Kappa Nu Honor Society.