Application of Multivariate Statistics to Fermentation Database Mining

by

Roy T. Kamimura

MSCEP, Massachusetts Institute of Technology,
June 1992

BSChE, University of California, Berkeley,
June 1990

Submitted to the Department of Chemical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in Chemical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1997

Author _____
Department of Chemical Engineering
May 19, 1997

Certified by ___
Greg Stephanopoulos
Professor of Chemical Engineering
Thesis Supervisor

Accepted by __
Robert E. Cohen
St. Laurent Professor of Chemical Engineering
Chairman, Committee for Graduate Students

Application of Multivariate Statistics to Fermentation Database Mining

by

Roy T. Kamimura

Submitted to the Department of Chemical Engineering
on May 19, 1997 in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in
Chemical Engineering

ABSTRACT

Ideally during the course of a fermentation, an on-line measure estimating the performance of the current operation is desired. It is beneficial to determine as soon as possible whether the current run is of acceptable quality. Unfortunately, due to the poor mechanistic understanding of most biological systems complicated further by the lack of appropriate on-line sensors and the long lag time associated with off-line assays, attempts at direct on-line evaluation of bioprocess behavior have been hindered. The data that is available often lacks sufficient detail to be used directly and after a cursory evaluation is stored away. It is hoped that this historical database of process measurements may still retain some useful information. The hypothesis is that the underlying mechanisms are exemplified in the varied measurement profiles that define characteristic fingerprints of different types of process behavior. Processes that have similar outcomes should have a characteristic fingerprint. Hence, by correlating these fingerprints to the process behavior, process classification can be performed.

The aim of this thesis is to provide a systematic approach to identifying and modeling these patterns in a historical database. The first step involves mean hypothesis testing with the goal of isolating which process measurements as well as time windows have the ability to discriminate among different types of process behavior. Next, a novel cluster analysis technique is used to assess the data homogeneity. When developing a model of a class, it is important to understand the type of variability present in the data. An assessment must be made to determine if all the lots in a class behave similar to one another and to identify which ones are atypical. Lots representative of the class are to be used in the training of the model. The third stage is actual modeling of the different classes using the results of the previous two. Discriminating variables and time windows are used in the selection of model inputs while model parameters are determined by a training set selected by cluster analysis.

After training, the behavior of a current run is then compared to these models and a classification assigned. This methodology has been applied to several case studies involving industrial data and has been able to provide early process classification.

Thesis Supervisor: Gregory Stephanopoulos
Title: Professor of Chemical Engineering

To My Family

and

Yutaka Yakabe

Acknowledgments

Professor Gregory Stephanopoulos, for his guidance, patience, and support during my stay here at MIT. I would like to thank him for exhibiting patience and enthusiasm for my research as well as having the uncanny ability to capture big picture. And for making sure that I always stayed focus on the issues at hand. To members of my committee, Professor Paul Barton, Professor Charles Cooney, and Professor George Stephanopoulos for their inputs, comments, and guidance.

Members of the MIT Consortium for Fermentation Diagnosis and Control whose financial support and industrial support made this work possible. Special thanks to Dr. Joseph Alford of Eli Lilly for his input and comments.

Members of the research group, Troy Simpson, Robert Balcarcel, Mattheos Koffas, Michael Marsh, Kurt Yanagimachi, Marc Shelikoff, and Dr. Aristos Aristidou, for their constructive criticism and input. Special thanks to Cathryn Shaw-Reid for the memories at Practice School as well as her pleasant company.

Mathworks for creating Matlab thereby greatly aiding my research while at the same time making me lose sleep.

Mt. Dew, Iced Mocha Blast, and other caffeinated beverages without whose support burning the midnight oil would have been impractical.

Friends at MIT and beyond (if there truly is such a place). "The Fellas", Henry Isakari, Adam Lim, Kwai and Wai Lau, and George Wong as well as Vicky and Vivian Yap for keeping me up-to-date on daily events in California as well as the rest of the world. Cherry Ogata, "Pengwyn," for livening up my stay here. Suman Banerjee for his unique prospective on life at MIT. Elaine Aufiero-Peters for a happy smile and a compassionate ear. Dr. Christine Ng for the help with the Internet and Athena tricks.

Alex Diaz for maintaining the focus at MIT. Dr. Martin Reinecke whose prospective on life and governments made for great conversation. Maria Klapa - my successor in ruling the office- for driving me nuts as well as keeping the office in order with her spunk and laughter. Dr. Silvio Bicciato, "Partner in Crime," Dr. Urs Saner, and Professor Hiroshi Shimizu whose insight, advice, and help contributed greatly to helping me finish my thesis. Betty Irawan for being a good friend. Hiroshi Saito and Sal Salim, "Escapees from the Insane Asylum", for the late-night discussions, food, movies, and other escapist outlets. "The why are we here at MIT" group of Angelo Kandas, Dave Oda, and Harsono Simka for movies, poker, dinner, lunch, philosophizing, games, and other attempts at keeping the world sane.

Parents for their love and support. And finally to my wife, Maki, who despite not being an MIT student was willing to spend late nights at the Institute to keep me company. This work could not have been finished without her love, encouragement, and support.

Table of Contents

List of Figures

List of Tables

Chapter 1:

Introduction

1.1 Thesis Motivation

Due to their complex and time-dependent nature, biotechnological processes are inherently difficult to model and control. Incomplete understanding of microbial physiology coupled with issues of system observability have hindered attempts to mechanistically model most bioprocesses. The lack of on-line sensors to measure key metabolic compounds and the long lag time associated with existing off-line assays have further impaired sophisticated process control development and optimization. Thus, bioprocesses have experienced problems in process scale-up, suboptimal operations, and variable product quality, yield, and productivity (Royce, 1993).

To address these issues, a large number of fermentation variables are routinely monitored during the course of a run. Unfortunately, much of the data obtained lacks sufficient detail to be used in a mechanistic model and after a cursory evaluation most of the information is stored away. It is possible, however, that much of this historical data is still informative. Different types of process behavior, for example, high, medium, low product concentration, may have a characteristic "fingerprint" reflected in the measurement profiles. These fingerprints can be spikes, process trends, or any other feature that is a persistent or outstanding pattern in the data. By identifying which fingerprints are characteristics of certain classes, on-line process classification can be attempted. A comparison is made between the profiles of the existing run to those of existing classes and a search for a match initiated. Once a match has been achieved, the run is assigned a classification of its anticipated outcome.

Several methodologies involving artificial intelligence (AI) and multivariate statistical (MS) techniques have been used to model these fingerprints - data patterns -

(Massart, D.L., Vandeginste, B. G. M., et. al ,1988; Warnes, Glassey, et al, 1996). Knowledge-based systems (KBS) have incorporated qualitative information about processes as well as experience and knowledge from human sources (Halme, 1989). Artificial neural networks and principal components analysis have modeled the varying correlational structure among the variables for different process behaviors (Raju and Cooney, 1992; Montague and Morris, 1994; Wold, 1976). Decision trees have segregated lots on the basis of information theory (Quinlan, 1986; Saraiva and Stephanopoulos, 1992). Unfortunately, most of these techniques have focused primarily on modeling the data with minimal concern to the quality of the data presented.

The issue of data quality has become more pressing as data modeling has been facilitated by the growing computational power of computers for performing complex calculations (Royce, 1993). Numerically intensive calculations involving large volumes of process data can now be accomplished in a relatively short period of time. There are several inherent dangers, however, to sending all this information to the computer and having the algorithms sort through the ocean of data without providing some guidelines as to what the user considers important. Relationships identified as significant by the algorithms may not be apparent to the user. This is especially crucial for modeling tools such as artificial neural networks (ANN's). ANN's are well-known for their ability to capture complex nonlinear behavior, but they are also known for generating relationships between inputs and outputs whether they are physically feasible or not. This in itself is not a fault of the algorithm since its objective is to relate outputs to inputs. The error lies in forcing the algorithm to derive a relationship which is not grounded on physical, chemical, or biological principles. Is this the fault of the data, algorithm, or user? The key here is not to simply model what is available but what is important. Another danger not considered is the issue of too much irrelevant information. Excessive data, as opposed to redundancy, if it is not pertinent to the modeling objectives can degrade algorithm performance. For example, in attempting to model variable 1's profile, the reconstruction of variable 6, which has a strong correlation to the model objective say yield, may be compromised. Since variable 6 is not accurately modeled, the correlation may not be uncovered and this knowledge is

lost. Another data quality issue is to consider the nature of model inputs. For instance, in linear regression there are two inherent assumptions: 1) the variables that are measured are relevant to the modeling objective; 2) and that they are independent. Failure to check these assumptions can cause the resulting model to be incorrect. The lack of independence can produce instabilities in the regression coefficients. Since mechanistic understanding of bioprocesses is often incomplete, it is possible that not all the variables measured are important. In either case, linear regression may have been sufficient to model the data but its performance is marred by using improper data inputs. Selection of the training set is another arena where understanding the data quality plays a significant role. Atypical samples should be isolated and not used for model training. These are lots which belong to one class (same process outcome - e.g. high yield) but have characteristics in common with other classes - i.e., low lots having profiles that make them resemble a high lot. If care is not taken and those lots are included in the training set, model performance can again be impaired.

Hence there is a tremendous need to understand the quality of the data prior to modeling if an accurate conclusion is to be drawn from the algorithm's performance. Are all the variables measured relevant to the modeling objective? Are the class descriptors one assigns appropriate? For example, a run may be assigned a high label but how is this label assigned? - the result of a titer measurement performed at the end of the run? ; is it based on average productivity?, etc. The label definitions can affect whether it is indeed possible to perform on-line classification from the data that is available. If this is not possible from the data present, then one needs to be aware of this. Another consideration is if a run has profiles representative of substandard quality but the endpoint assay says it is standard which one does one believe? Following an old adage, if it walks, squawks, and swims like a duck but someone says it's a swan who does one believe? One needs to be aware of how such samples can affect the modeling.

It is only after the limitations and assumptions of the data are known can modeling begin, see Figure 1.1. After all, models should bring to light the knowledge stored in the data.

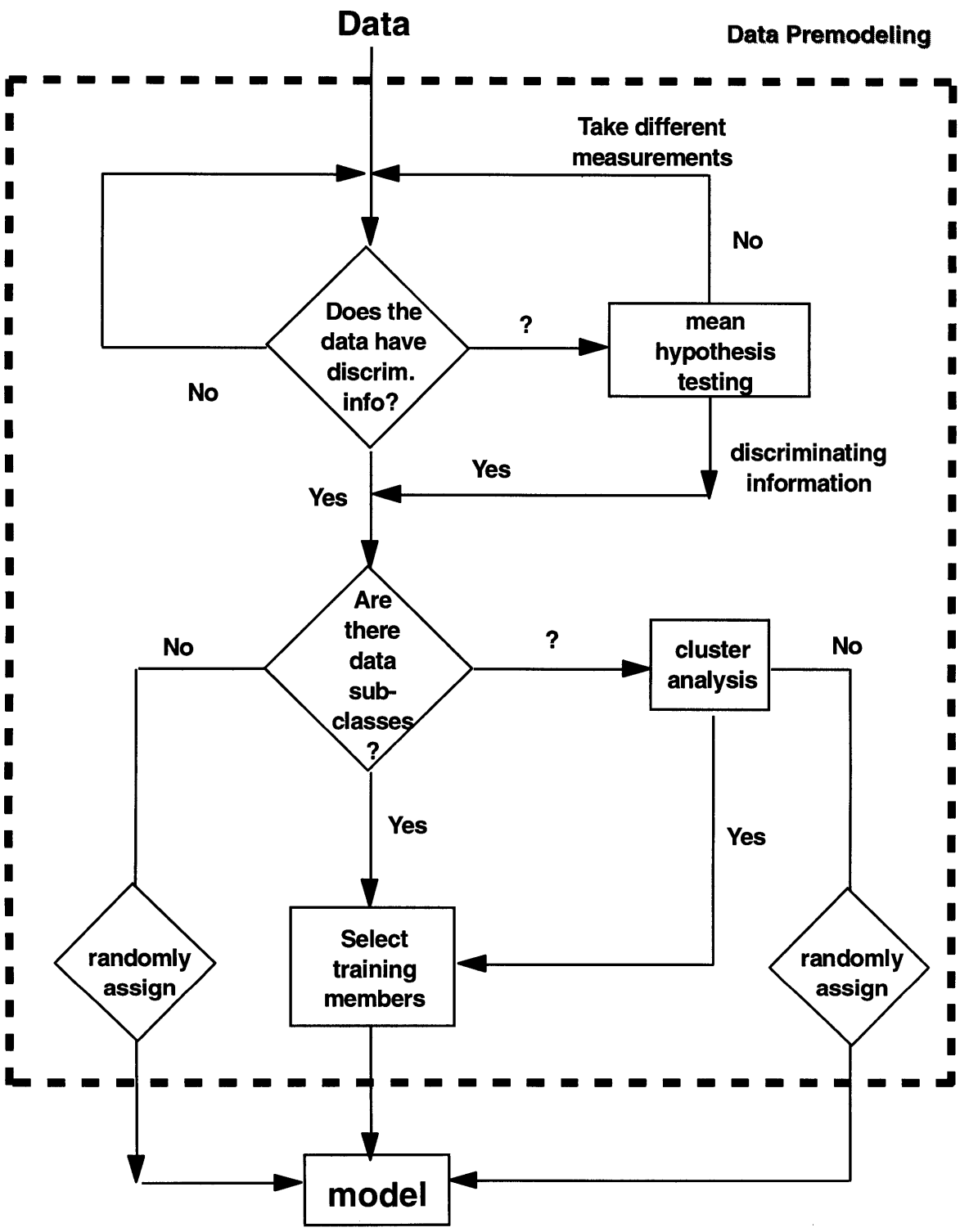


Figure 1.1 Overview of thesis methodology.

1.2 Thesis Objectives

The objectives of this thesis are summarized as follows: 1) identification of discriminating variables and time windows useful for detecting class differences; 2) development of a systematic approach for designing training sets for models; and 3) early process performance classification via models developed from historical records.

1.2.1 Thesis Overview

The aim of this research is to provide a systematic approach to performing process classification by first assessing the quality of data available and then utilizing models to capture characteristic patterns in a historical database of previous runs. The underlying hypothesis is that different types of process behavior (classes- e.g., high, medium, low titer) have a fingerprint reflected in the measurement profiles and by correlating this fingerprint to the different classes one can perform process diagnosis/classification. The first step in this approach involves mean hypothesis testing with the goal of isolating which process measurements as well as time windows have the ability to discriminate among different types of process behavior. Next, a novel modification of cluster analysis is used to assess the data homogeneity. When developing class models, it is important to understand the type of variability present in the data. Training a model requires that the limits of the information available be known. This is not simply limited to identifying outliers. If there are samples that have properties of one group but the label of another, these need to be isolated and considered separately. Furthermore, it is essential when creating the training set that it reflects the typical variations observed in the data, not aberrant situations. This is not possible if one is not aware of how well-behaved the data is. The third stage is modeling the different classes using as inputs the results of the previous two - discriminating variables/time windows and a carefully selected training set. The behavior of a current run is then compared to these models and a classification assigned. This methodology has been applied to case studies involving industrial data and has been able to provide early process classification.

1.3 Thesis Novelty/Impact

As shown below, the methodology developed in this thesis is novel in the following aspects:

- The notion of analyzing data quality before modeling is addressed - most pre-modeling schemes either mean-center the data or normalize the variables to simplify analysis but they do not focus on the nature of the data presented.

- A systematic procedure for identifying variables and time windows capable of discriminating among different data classes is developed. Currently, many discriminating variables are identified on the basis of the algorithm used to model the data. This is inappropriate as the results are influenced by model bias. What is needed is a model-independent approach which is presented here.

- Many statistical methods are designed to analyze discrete data, not time series - in particular not multiple time series, which is common in most chemical processes. The techniques developed here address the latter.

- A novel method for classification of runs is presented, looking at the impact of process measurements on classification.

While this thesis is based on case studies of industrial fermentations considering the generality of the points listed above, the algorithms presented here are generic enough to be applied to any chemical process where the data is in the form of multiple time series. In fact, these techniques are applicable to any field of study where the objective is to discriminate among different classes of data.

1.4 Thesis Organization:

This thesis is organized into 5 chapters. This first chapter provides the motivations behind this work and an overview of the research objectives. Chapter 2 discusses the role of mean hypothesis testing in variable and time window selection. In the third chapter, cluster analysis will be used to assess data homogeneity. Chapter 4 introduces the models based on the results of the previous 2 chapters and amalgamates all into one cohesive approach for process classification. Chapter 5 is a summary of all the results and their significance as well as directions for future work.

1.5 References

Halme, A. (1989)

"Expert system approach to recognize the state of fermentation and to diagnose faults in bioreactors," *Computer Applications in Fermentation Technology,* ed. by N.F. Thornhill, N.M. Fish, and B. I. Fox.

Montague, G. and Morris, J (1994)

"Neural-network contributions in biotechnology," *TIBTECH,* vol 12, p 312-324

Massart, D.L., Vandeginste, B. G. M., et. al (1988)

Chemometrics: a textbook, Elsevier Science Publishers, New York.

Quilan, J. R. (1986)

"Induction of decision trees," *Machine Learning,* 1, 1, p 81-106

Raju, G. K. and Cooney, C (1992)

"Using neural networks for interpretation of bioprocess data," In *Proceedings of the IFAC Modelling and Control of Biotechnical Processes,"* p 159-162

Royce, Patrick N. (1993),

"A discussion of recent developments in fermentation monitoring and control from a practical perspective," *Critical Reviews in Biotechnology*, 13(2); 117-149

Saraiva, P., and Stephanopoulos, G., (1992)
"Continuous process improvement through inductive and analog learning," *AIChE J.*, 38, 2, p 161-183

Stephanopoulos, G. N., and C. Tsiveriotis, (1989),
" Toward a systematic method for the generalization of fermentation data," Computer Applications in Fermentation Technology: Modeling and Control of Biological Processes. ed. by N. M. Fish, R. I. Fox, and N. F. Thornhill, Society of Chemical Industry- Elsevier Science Publishers, LTD.

Warnes, M.R, Glassey, J., et. al (1996)
"On Data-Based Modelling Techniques for Fermentation Processes," *Process Biochemistry,* vol 31, no 2, p147-155

Wold, S. (1976)
"Pattern recognition by means of disjoint principal components models," *Pattern Recognition,* vol. 8, p 127-139

Chapter 2:

Mean Hypothesis Testing (MHT)

2.1 Background

Many modeling algorithms attempt to correlate all available measurements to an objective such as yield, product concentration, or in this case process outcome. The implication is that all the data is relevant to the model but this is not always the case. Some measurements exist for control considerations and others simply to monitor the process from a regulatory standpoint. Hence, not all of these variables are useful in discriminating between different types of process behavior. It is also important to realize that some variables may be discriminating only during a particular time period, a "time window," in the process. Most modeling techniques do not address these points (Kell and Sonnleitner, 1996).

Another problem with current modeling methods is that they determine discriminating variables in the context of specific models and thus are subject to any bias the models themselves have (Coomans, D., Massart, D.L., and Broeckaert, 1981; Wold, 1976). These bias are often in how the algorithms capture the interrelationships among the different variables. For example, Wold uses principal components analysis (PCA) to identify discriminating variables; a comparison is made between the variance generated by fitting a model (model 1) to data of a different class (class 2) and the variance generated by fitting the model (model 1) to the class for which it was designed (class 1). Variables used in model 1 that maximize the difference in variances are viewed as discriminating. While this is a very useful technique, it is dependent on the model's ability to characterize the data accurately. PCA being a linear technique may not be useful for modeling highly nonlinear systems as in most bioprocesses and so comparing the variances is questionable. What is needed is a methodology that

focuses on the data structure itself and let the measurements "speak" for themselves. The raw variables themselves should be focus of analysis not a model's approximation.

The approach presented in this thesis is mean hypothesis testing (MHT). This is a basic statistical tool used to identify differences between populations of samples utilizing the mean as a measure of the population's bulk behavior. In this chapter, the focus is on how to determine which process measurements behave differently between different process outcomes, specifically high (good) and low (bad) yield production runs, by observing how distinct their mean values are. Variables with dissimilar mean behaviors in the high and low classes will be considered discriminating. In turn, these discriminators will then be used to characterize an unknown run as high or low. Unfortunately, mean hypothesis testing (MHT) is traditionally designed to handle discrete data rather than time series which are typical of chemical and biotechnological processes. To address this situation, a key assumption underlying this test must be addressed - that the data must consist of independent samples. In a time series, the value at each time point is dependent on its values from the previous time points; hence, one cannot compare time points within a lot. The solution is to recast the data into a form where the means that are compared are the variable's class-average values, taken at each point in time. It is important to stress that these values are **different from time-average means** which are calculated over the course of one run. The means used here are calculated over the membership of a class for each time point. The rationale is that each run within a class is independent of the other runs and as long as the comparison is made between lots at similar time points, the independent sampling rule is not violated. This novel approach allows the processing of multiple time series data by conventional multivariate mean hypothesis testing.

2.2 Objective

The goal is the systematic identification of discriminating process variables and their respective time windows in the context of process classification. The mean value(s) of variable(s) for different classes at each time point is compared and a determination is made as to whether different process behaviors can be differentiated on the basis of such measurements.

2.3 Theory

To be able to discriminate among different groups, the first step is to identify what and where the differences are. Since the population mean is a measure of the bulk behavior of a group, it is a summary of the group's behavior. Hence, it is a convenient starting point to begin the analysis. Here, the populations are the different process outcomes, segregated as high and low classes, and the samples are the individual production runs, lots. The class means of the process variables are compared to determine if their overall behavior is statistically different. Large discrepancies in the means indicate that the variables behave dissimilarity in the two groups and so these variables are viewed as class discriminators. The means themselves are calculated by extracting values from the same time points of each lot in the class, see Figure 2.1. For each time point, t, a class mean is generated over the class membership. It is this mean that is then compared between the classes.

Mathematically, the representation is as follows:

$$\mu_{ij}(t) = \frac{1}{n_i}\sum_{k=1}^{n_i}\left(x_{ij}(t)\right)_k \tag{1}$$

$$\vec{\mu}_1(t) = [\mu_{11}(t)\ \mu_{12}(t)\ \mu_{13}(t)\ \ldots\ \mu_{1c}(t)]^T \tag{2}$$

$$\vec{\mu}_2(t) = [\mu_{21}(t)\ \mu_{22}(t)\ \mu_{23}(t)\ \ldots\ \mu_{2c}(t)]^T \tag{3}$$

$\vec{\mu}_i(t)$: class average vector of class i at time t

$x_{ij}(t)$: variable j of class i at time (t)

c: number of variables

n_i: number of members of class i

k: kth lot of class i, ranges from 1 to n_i

A hypothesis test is performed to determine if the 2 means are statistically equivalent or not. If equivalent, this suggests that the variable (in univariate case) or the vector of variables (in multivariate case) under investigation are statistically indistinguishable for the 2 classes at least for that time point. The test is repeated for

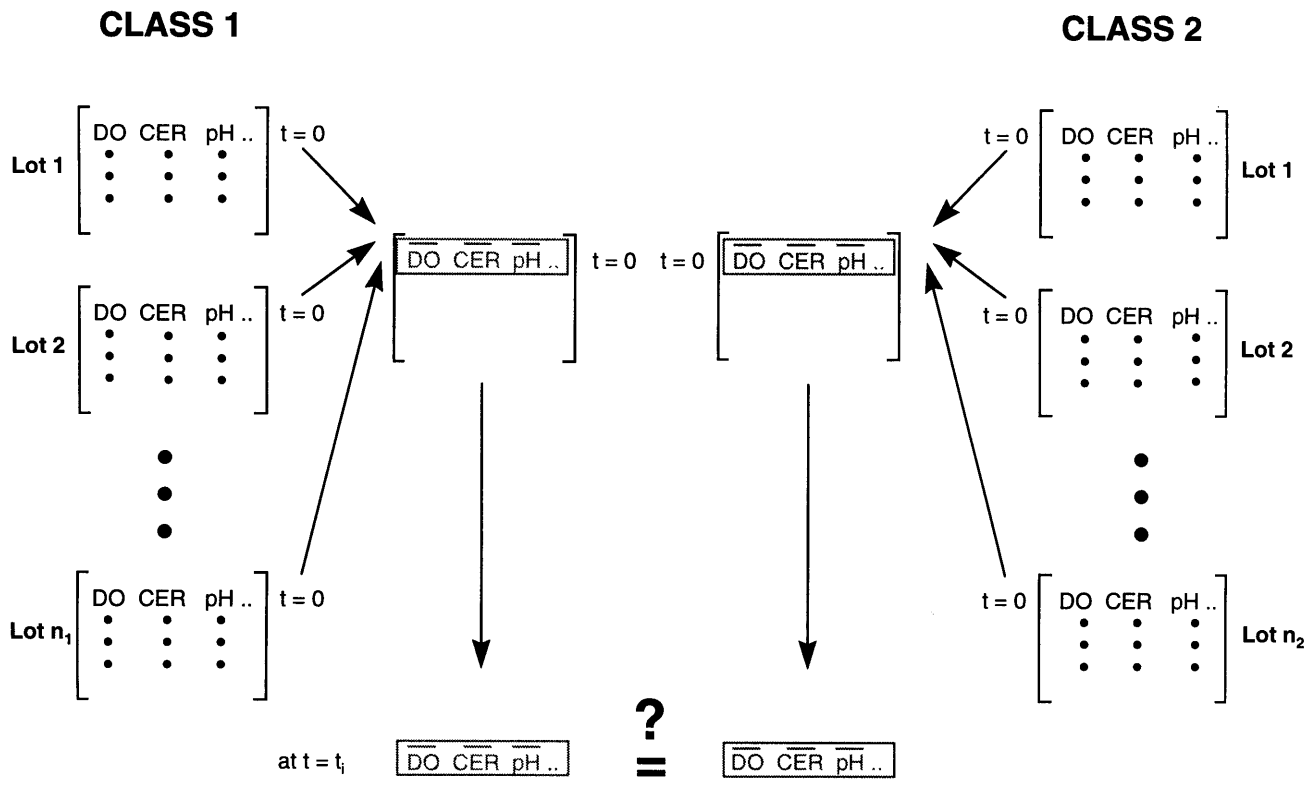


Figure 2.1 Concept of Mean Hypothesis Testing. Data from similar time points of each lot is extracted. The average of these points is then calculated and compared for hypothesis testing.

each subsequent time point generating a profile of how the 2 classes differ over time with respect to one another. Figure 2.2 illustrates conceptually the univariate case while Figure 2.3 shows the bivariate situation. The subplots on the figures denote the probability distributions of the variable values about the mean. For the univariate case it is a Gaussian shape. The elliptical shapes of the bivariate plot is the two-dimensional equivalent to a Gaussian curve for pair-wise data.

Once the influence of time has been corrected for, the standard techniques for mean hypothesis testing for multivariate systems can be applied with the following 2 considerations in mind: 1) the number of lots making up each class is typically less than 30, so the uni/multivariate t-statistic is employed rather than z-test which is designed for large sample sizes; and 2) the standard deviation/covariance matrix for each class is not be assumed to be the same nor constant. The second point is not covered in most statistical texts for the multivariate case but is a common situation for most process data. Failure to consider these two points can lead to erroneous interpretations of the test results.

The following sections is a review of the statistical theory of mean hypothesis testing in the context of this research. For a more thorough coverage, the reader is referred to Mardia, Kent, and Bibby 1979.

2.3.1 Single Variables - Univariate Case

For each time point, mean hypothesis testing determines which of 2 competing suppositions about the mean is likely to be correct, equal or not equal. The basis for selection depends on how likely the observation is due to random chance or is statistically significant. In this case, one would like to know whether the mean(s) of a variable or set of variables from 2 different classes is different enough that the variable(s) can be used as a basis of discrimination. This is achieved by creating 2 hypotheses, the null (H_o) and the alternative (H_a). Let μ_1 and μ_2 represent average values of the same variable taken from 2 different classes, denoted as 1 and 2. The null hypothesis states there is no difference in the means and this is translated as $\mu_1 - \mu_2 = 0$. The alternative is that the difference is significant and not a random

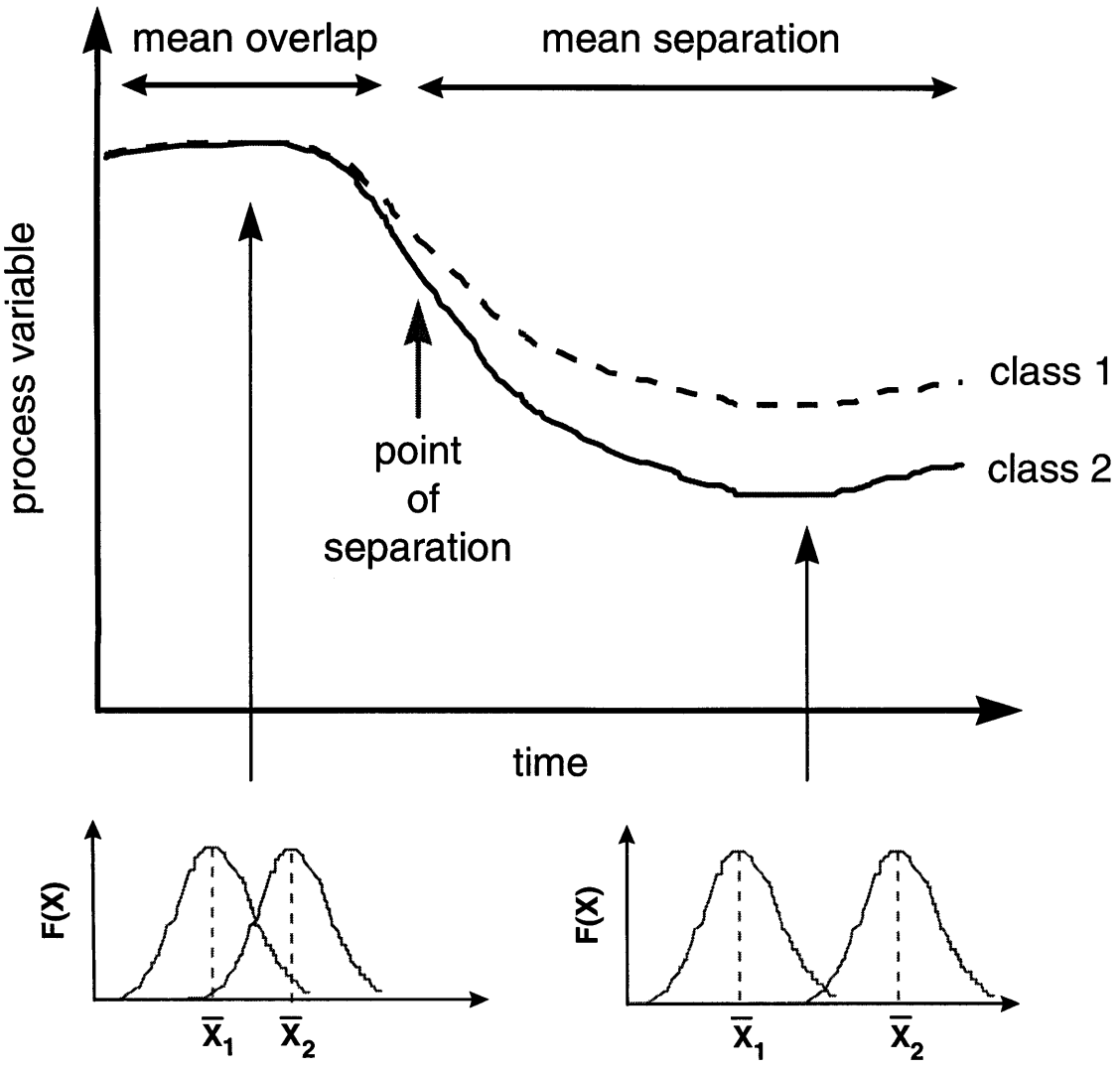


Figure 2.2 In the univariate case, the class means are assumed to have a Gaussian distribution as seen in the 2 smaller graphs. Mean overlap is when the separation in the means is covered by the standard deviations of the classes.

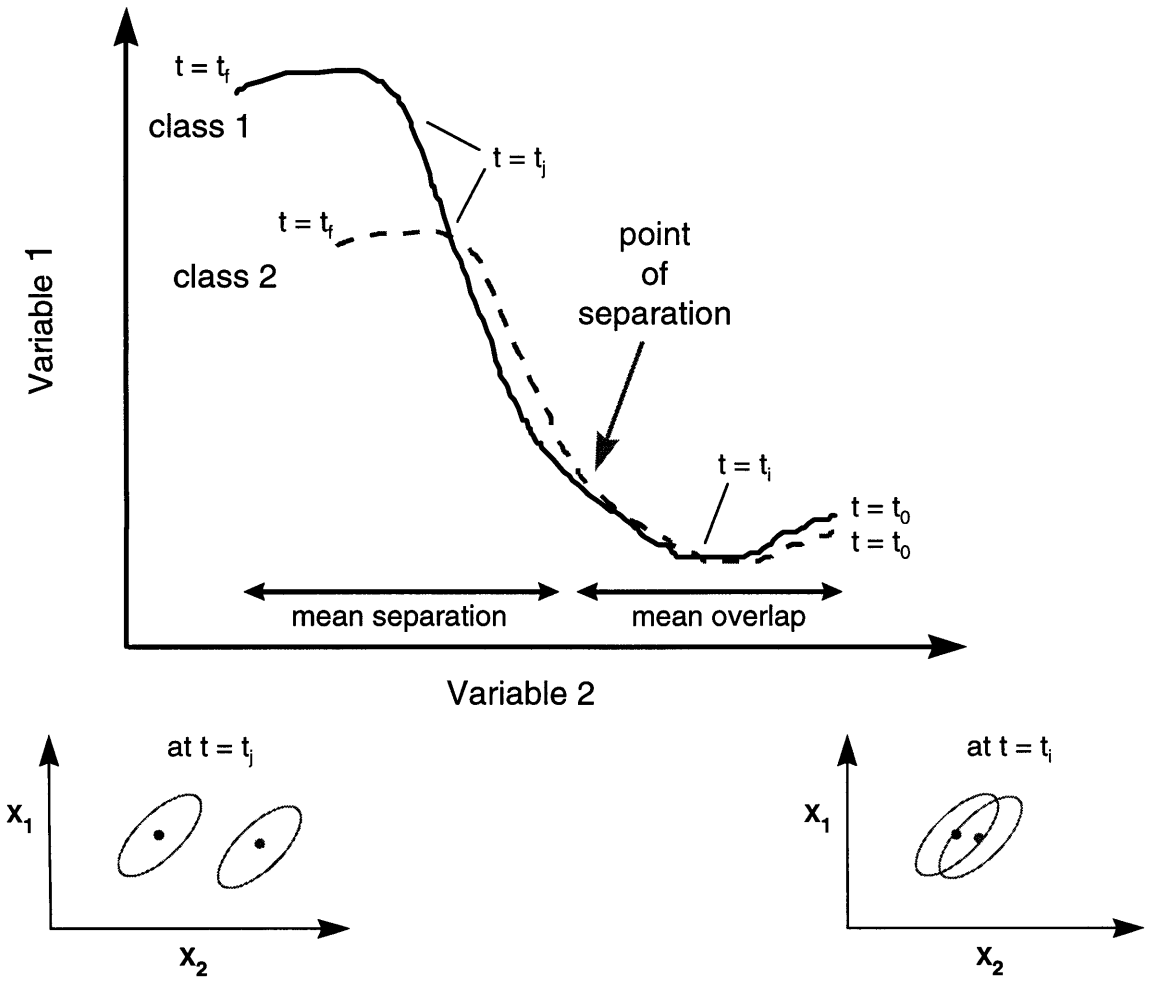


Figure 2.3 In the case of pairwise data, a bivariate Gaussian distribution of the means is assumed, which is elliptical in shape.

occurrence. To determine which hypothesis is valid, a test-statistic from the data is generated and then compared to a probability distribution. If the statistic is found to exceed the statistical threshold set by the user, then the null hypothesis is rejected and the means are taken to be different. For sample sizes (class membership < 30), the t-statistic is used as the test-statistic. When this measure exceeds the tabulated t-value, see Table 2.1, (determined by significance level set by the user and the number of degrees of freedom), the null hypothesis of equal means is rejected. It is important to note that this test, depending on the significance level set, will not reject the null hypothesis even if the means are numerically quite different but the standard deviations are sufficiently large. In that particular case, the test interprets the observed difference in the means to be statistically questionable since the data exhibits large variability.

Mathematically, the test is represented as follows:

$$H_o : \mu_1 - \mu_2 = 0 \tag{4}$$

$$H_a : \mu_1 - \mu_2 \neq 0 \tag{5}$$

test statistic:
$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \tag{6}$$

rejection region: $|t| > t_{\alpha/2,\nu}$ (significance level = 1-α) (7)

degrees of freedom: $\upsilon = n_1 + n_2 - 2$ for equal class size

where:

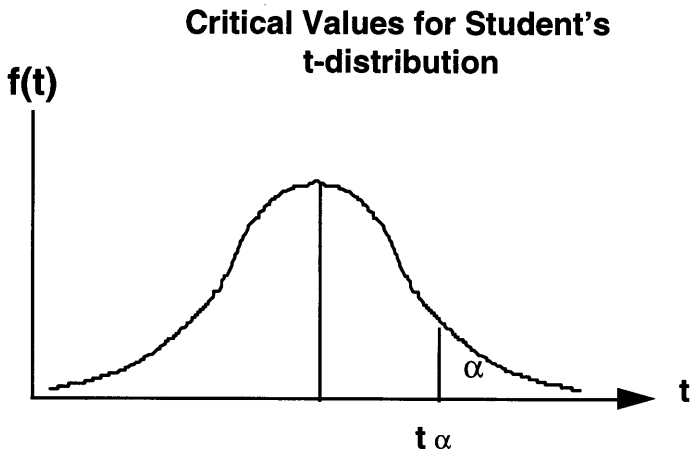
μ_i: sample mean of variable of class i (i =1,2)

n_i: number of members of class i (i = 1,2)

s_i^2: standard deviation of variable of class i (i= 1,2)

υ: degrees of freedom - this parameter is used to take into different sample sizes and number of variables under consideration

Table 2.1 Statistical table for t-values adapted from Mendenhall and Sincich, 1992



v	$t_{.100}$	$t_{.050}$	$t_{.025}$	$t_{.010}$	$t_{.005}$	$t_{.001}$	$t_{.005}$
1	3.078	6.314	12.706	31.821	63.657	318.31	636.62
2	1.886	2.920	4.303	6.965	9.925	22.326	31.598
3	1.638	2.353	3.182	4.541	5.841	10.213	12.924
4	1.533	2.132	2.776	3.747	4.604	7.173	8.610
5	1.476	2.015	2.571	3.365	4.032	5.893	6.869
6	1.440	1.943	2.447	3.143	3.707	5.208	5.959
7	1.415	1.895	2.365	2.998	3.499	4.785	5.408
8	1.397	1.860	2.306	2.896	3.355	4.501	5.041
9	1.383	1.833	2.262	2.821	3.250	4.297	4.781
10	1.372	1.812	2.228	2.764	3.169	4.144	4.587
11	1.363	1.796	2.201	2.718	3.106	4.025	4.437
12	1.356	1.782	2.179	2.681	3.055	3.930	4.318
13	1.350	1.771	2.160	2.650	3.012	3.852	4.221
14	1.345	1.761	2.145	2.624	2.977	3.787	4.140

The significance level is generally set at 95%, indicating that there is a 5% (α) chance that the test will erroneously reject the null hypothesis when it is actually true. Since the variation exhibited by 2 classes can vary considerable, the more general case of unequal standard deviations is presented above.

2.3.2 Combinations of Variables - Multivariate Case

Conceptually, this situation is similar to viewing data in a state-space representation. How the variables relate to one another is compared to see if their behavior varies between classes. Do variables, x_i and x_j, vary the same way in class 1 as they do in class 2 for the time point t? Mathematically, the multivariate form is similar to the univariate case with adjustments for the multivariate nature of the input: a the single values replaced by a vector of means, the standard deviation by the covariance matrix, and the degrees of freedom adjusted by the additional number of variables. As in the univariate case the more realistic case of unequal population covariance matrices is considered as opposed to the common assumption of equal covariance matrices which is presented in most statistical texts. One major difference from the univariate form, however, is that multivariate t-statistic is now compared with the F-distribution rather than the student's t-distribution see Table 2.2 for the F-distribution. For each time point t, MHT is mathematically represented below (for full description, refer to Mardia, Kent, and Bibby 1979):

$$H_o : \vec{\mu}_1 - \vec{\mu}_2 = \vec{\mathbf{d}} = \vec{\mathbf{0}} \tag{8}$$

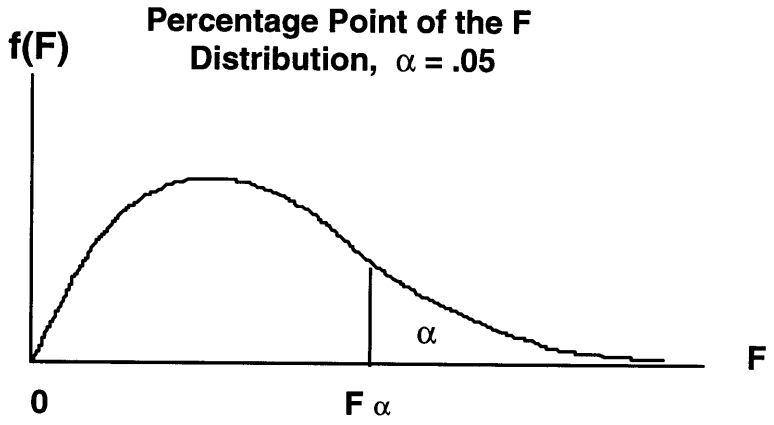
$$H_a : \vec{\mu}_1 - \vec{\mu}_2 = \vec{\mathbf{d}} \neq \vec{\mathbf{0}} \tag{9}$$

test statistic (Union Intersection method) : $$UIT = \left(\frac{f^* - c + 1}{f^* c} \right) \vec{\mathbf{d}}^T \mathbf{U}^{-1} \vec{\mathbf{d}} \tag{10}$$

rejection region: $$UIT > F_{\alpha, \upsilon_1, \upsilon_2} \tag{11}$$

degrees of freedom: $\upsilon_1 = c, \upsilon_2 = f^* - c + 1$

Table 2.2 Statistical table for F-values adapted from Mendenhall and Sincich, 1992



v1	1	2	3	4	5	6	7	8	9
v2									
1	161.4	199.5	215.7	224.6	230.2	234.0	236.8	238.9	240.5
2	18.51	19.00	19.16	19.25	19.30	19.33	19.35	19.37	19.38
3	10.13	9.55	9.28	9.12	9.01	8.94	8.89	8.85	8.81
4	7.71	6.94	6.59	6.39	6.26	6.16	6.09	6.04	6.00
5	6.61	5.79	5.41	5.19	5.05	4.95	4.88	4.82	4.77
6	5.99	5.14	4.76	4.53	4.39	4.28	4.21	4.15	4.10
7	5.59	4.74	4.35	4.12	3.97	3.87	3.79	3.73	3.68
8	5.32	4.46	4.07	3.84	3.69	3.58	3.50	3.44	3.39
9	5.12	4.26	3.86	3.63	3.48	3.37	3.29	3.23	3.18
10	4.96	4.10	3.71	3.48	3.33	3.22	3.14	3.07	3.02
11	4.84	3.98	3.59	3.36	3.20	3.09	3.01	2.95	2.90
12	4.75	3.89	3.49	3.26	3.11	3.00	2.91	2.85	2.80
13	4.67	3.81	3.41	3.18	3.03	2.92	2.83	2.77	2.71
14	4.60	3.74	3.34	3.11	2.96	2.85	2.76	2.70	2.65

where:

$\vec{\mu}_i$: c x 1 vector of means of variables of class i (i =1,2)

$\vec{d} = \vec{\mu}_1 - \vec{\mu}_2$: c x 1 vector of difference of means (12)

$$\mathbf{U} = \mathbf{U}_1 + \mathbf{U}_2 \quad (13)$$

$$\mathbf{U}_i = \frac{\mathbf{S}_i}{n_i - 1} \quad (14)$$

$$\mathbf{S}_{ui} = \frac{n_i}{n_i - 1}\mathbf{S}_i \quad (15)$$

$$\frac{1}{f^*} = \frac{1}{n_1 - 1}\left(\frac{\vec{d}^T\mathbf{U}^{-1}\mathbf{U}_1\mathbf{U}^{-1}\vec{d}}{\vec{d}^T\mathbf{U}^{-1}\vec{d}}\right)^2 + \frac{1}{n_2 - 1}\left(\frac{\vec{d}^T\mathbf{U}^{-1}\mathbf{U}_2\mathbf{U}^{-1}\vec{d}}{\vec{d}^T\mathbf{U}^{-1}\vec{d}}\right)^2 \quad (16)$$

$\mathbf{S}_{ui}$: denotes the unbiased estimate of the covariance matrix of class i

c: number of variables

n_i: number of members of class i (i = 1,2)

f^*: adjusted degrees of freedom - this parameter is used to take into different sample sizes and number of variables under consideration

$\vec{d}^T\mathbf{U}^{-1}\vec{d}$: this term is often referred to as the Hotelling's multiple - sample T^2 statistic.

For each time point, a vector representing the difference of the means, $\vec{d}$, is compared to the zero vector taking into account the variance in the data about the mean. For the null hypothesis to be rejected, indicating a discrimination point, the UIT statistic must exceed the tabulated F-value. The complicated nature in which the degrees of freedom is calculated is to consider the effect of different covariance structures of the 2 populations. If both populations have different means but large covariance values, depending on the significance level set, MHT will view the means to be equivalent and denote the time point as non-discriminating.

Note: The univariate case is included simply as a point of reference and the equations are not used in this thesis. The actual algorithm uses the multivariate

equations but can be used to explore single variables by simply selecting the vector to have only one member.

2.3.3 Approach - Mean Hypothesis Testing (MHT)

First, the data is preprocessed to align all the time points. This action is to compensate for the differing starting points in addition to standardizing the run lengths. Lots will only be compared where comparable time points exist. An assumption is made that any time shifts in the process data are a pattern. The reasoning is that it is not possible to distinguish *a priori* from the data provided whether the shift is due to an operator action or a change in the reaction mechanism. After alignment, the data is then partitioned and rearranged into a form amenable to mean hypothesis testing. It is important to stress that comparing the class means from the same time points allows the standard multivariate tests to be applicable.

For each class and selected variable(s), a vector of mean(s) is calculated for each time point as well as the corresponding covariance matrix. These 2 parameters are then used to construct the Hotelling's T^2 statistic, $\vec{\mathbf{d}}^{T}\mathbf{U}^{-1}\vec{\mathbf{d}}$,which after modified to account for the degrees of freedom creates the UIT-statistic. This last statistic is then compared to the F-value. For a selected significance level, generally 95%, the decision to accept or reject the null hypothesis is made. Rejection of the null implies that the selected variables are useful in class discrimination. Figure 2.4 illustrates the entire approach.

As an initial starting point for the analysis, the discriminating power of each individual variable by itself should be explored. Those measurements found to be nondiscriminating should be removed from further analysis. This is to avoid the effect of discriminators masking the nondiscriminators when considering pairs, triplets, and higher variable groupings. Another benefit of considering only discriminators is the subsequent reduction in the number of variable combinations that need to be evaluated. While the possibility exists that combinations of nondiscriminating variables can provide discriminating information, the likelihood is remote. The situation is different from that of statistically designed experiments where individual factors may not

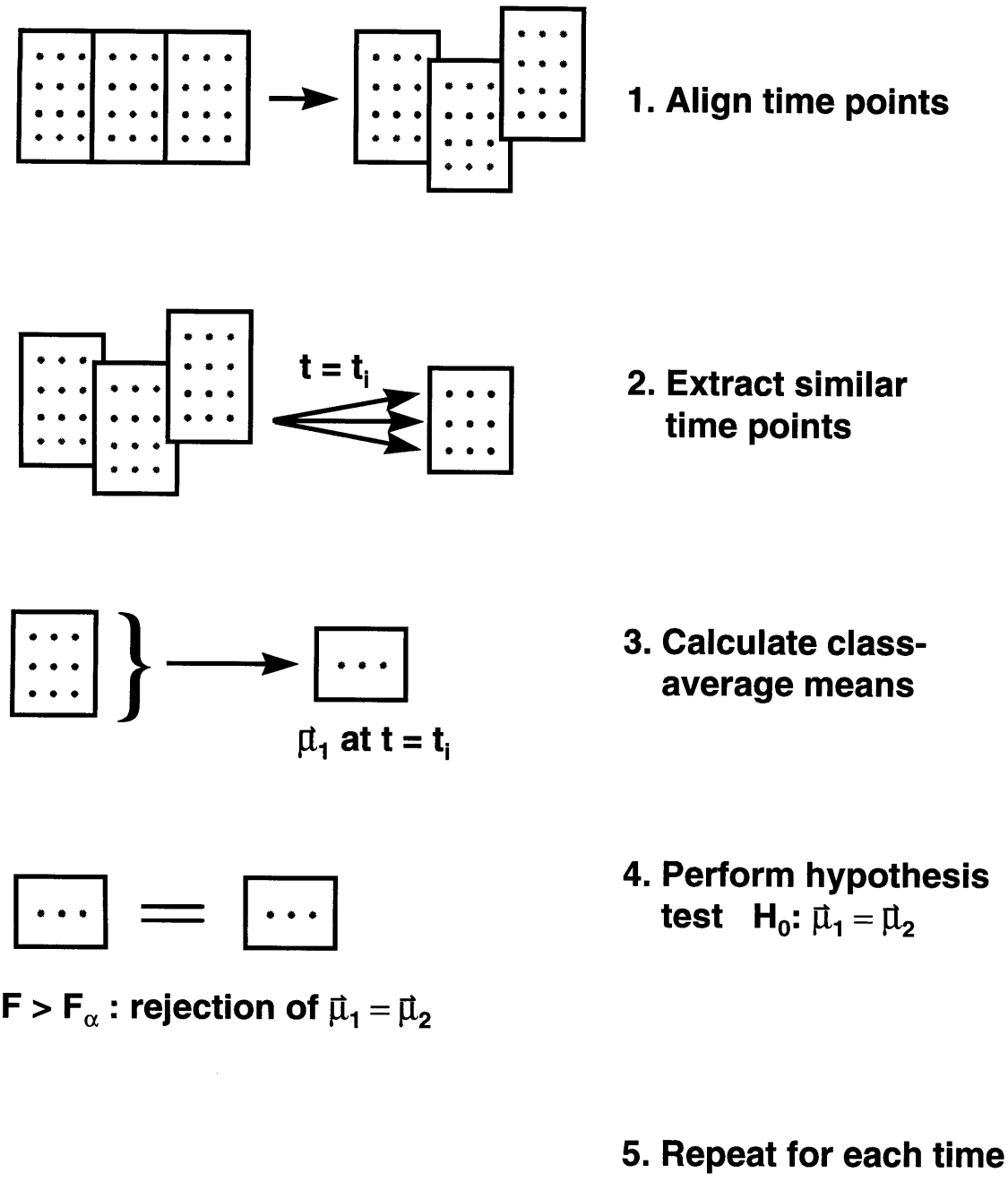


Figure 2.4 Overview of Mean Hypothesis Testing approach.

be discriminating but the combination is. The reason is that in this analysis any interactions by the variables is already present in the data itself. Nothing is added so what is observed is the discriminating ability of the variable in the presence of other variables.

If the discriminators identified in the first screen are satisfactory, the search can be terminated. Either all or a subset of the discriminators identified can be selected to continue the analysis to either cluster analysis or go directly into modeling. The advantage to concluding the analysis at this early point is the fast computation and ease of interpretation when considering only individual variables. If a more rigorous search is required, for example, to identify either a earlier time window of discrimination or a more stable region, then combinations of several variables can be considered. Looking at combinations of variables is not the same as examining the time windows predicted by each member of the combination. The reason is these combinations contain information about variable interactions and, as a result, can choose different time windows other than those selected by the individual variables. The drawback, however, lies in the number of combinations that must be evaluated which is ${}_nC_k$ where n is the total number of variables and k the variable subset. As an example, to explore all 7 variable groupings from a total of 14 variables involves searching through 3,452 combinations. This number does not include evaluating the single to 6 member groups. Another disadvantage is that the effect of variable interactions is not apparent when looking at the individual variables by themselves. The window of discrimination identified by a 7 variable set may differ from windows identified by each of the 7 variables separately.

2.4 Results

The analysis is performed by programs written in Matlab 4.2c on a Silicon Graphics R5000 workstation. The relevant codes are listed in the Appendix 2.7.1.

The data used in these case studies are provided by industrial participants of the MIT Consortium for Fermentation Diagnosis and Control. In all cases, the classes are predefined as high or low by the industrial source. The basis is product yield; but the numerical ranges for these classifications were not provided. For confidentiality

reasons, the units and identities of all variables have been removed and the data normalized between 0 and 1. All lots are standardized to the same length to simplify analysis.

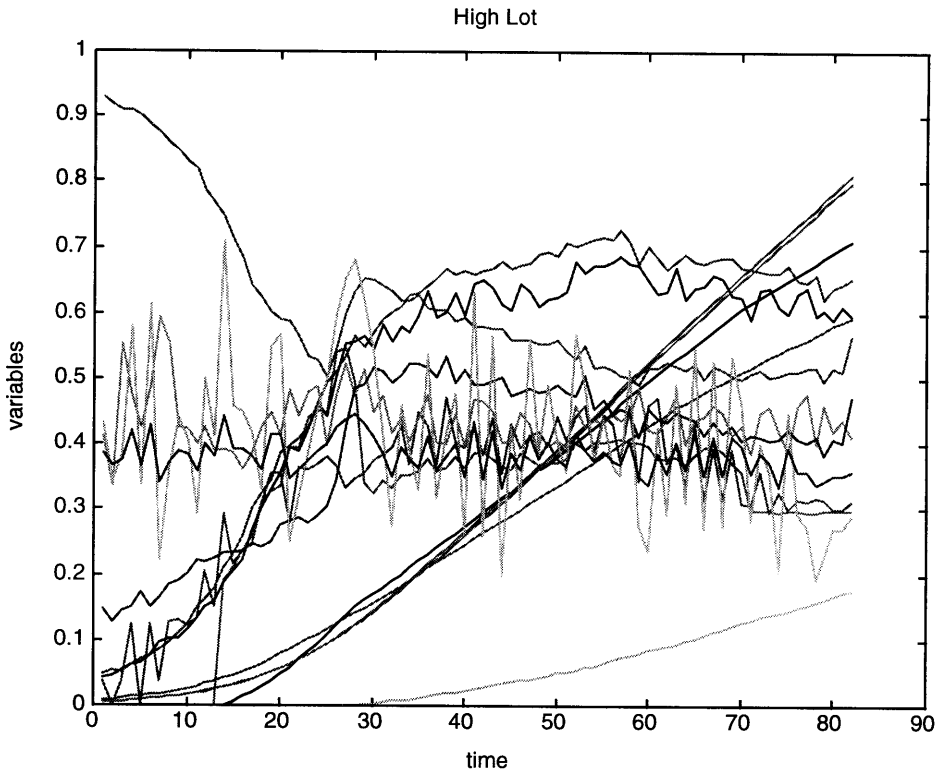
2.4.1 Case Study 1

Type:	industrial fermentation
Mode:	fed-batch
Run length:	82 time points
Number of variables:	17 (only variables 4-17 used however)
Number of classes:	2
Lots/class:	22 for high and 23 for low

2.4.1.1 Single Variable Effects

Figure 2.5 (a)-(b) displays a typical lot for each class. For this analysis, all 45 lots are used with 22 in the high class and 23 in the low. After data preprocessing , the first step in the analysis is to consider the discriminating influence of individual variables. Table 2.3 shows a typical result focusing on variable 4, a discriminator. The first column displays the observed value and the second column the statistical F-value that must be exceeded for the means to be not equivalent. The third column simply lists the **time point** when the null hypothesis of equal means is rejected, and hence, a discrimination point since the classes show a statistically significant difference. For simplicity, **n.d. denotes no discrimination**. The fourth column is simply a ratio of the first to the second column to provide an measure of the discriminating power - the greater this value is the larger the discrepancy between the observed and tabulated F-value. The disparity between the 2 F-values translates to a large difference in the means, which in turn suggests that the variable under inspection is a strong discriminator. Variables that fail the hypothesis test are considered to have equivalent means. This in turn designates them as nondiscriminating, since statistically on average, they exhibit similar behavior in both classes.

(a)



(b)

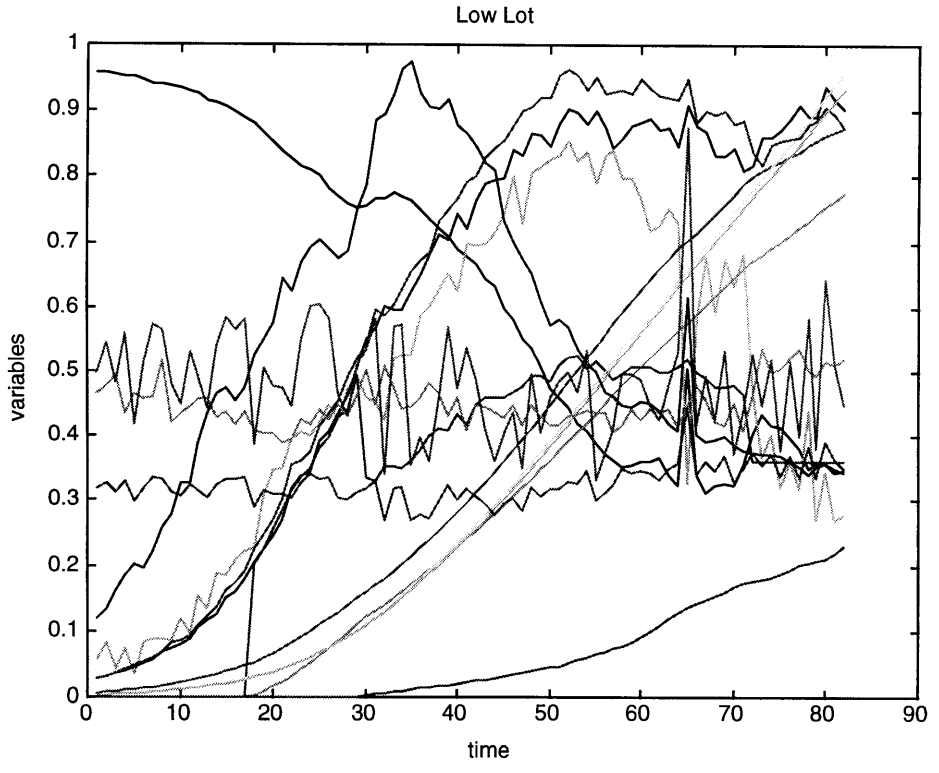


Figure 2.5 (a) High lot; (b) Low lot.

Table 2.3 Sample MHT output for variable 4

Calculated F-value	Tabulated F-value	Time of Discrimination	F-Ratio
0.04	4.07	n.d.	0.01
0.09	4.08	n.d.	0.21
0.01	4.08	n.d.	0
0.01	4.09	n.d.	0
0.03	4.09	n.d.	0.01
0.03	4.09	n.d.	0.01
•	•	•	•
•	•	•	•
•	•	•	•
4.65	4.08	32	1.14
9.00	4.11	33	2.20
10.40	4.11	34	2.53
11.38	4.11	35	2.77
13.33	4.11	36	3.24
•	•	•	•
•	•	•	•
•	•	•	•
53.68	4.07	80	13.2
65.09	4.07	81	16.0
69.16	4.07	82	17.0

As seen in Table 2.3, the first 31 points of variable 4 are the same in both classes and so the ability to differentiate classes is poor in this time region. From 32 to 82, a mean difference is detected. This can be verified by looking at Figure 2.6, where it is observed roughly after time point 30 that the 2 class means follow different trajectories. This result verifies that there are time windows where the variable's ability to discriminate is far greater than in other time periods. This information needs to be taken into account when modeling. There is no foreseeable need to consider the entire time course if only a fraction of it is meaningful.

In contrast, variable 17 is the behavior of a nondiscriminator. As seen in Table 2.4, there is no time period where the means are different. In fact, looking at column 4, with the exception of a few points in the beginning, the calculated F-value is often much less than the tabulated criterion. The corresponding time profile is in Figure 2.7. From the figure, it is interesting to note that visually there appears to be 2 unequivalent means but because of the scatter in the data MHT views the means to be similar enough to each other to not discount the null hypothesis.

The results of the remaining variables are summarized in Table 2.5 with the corresponding profiles in Figures 2.8 (a)-(i) for discriminators and Figures 2.9 (a)-(c) for the nondiscriminators - profiles for variables 4 and 17 have been presented so will not be listed in Figure 2.6 and Figure 2.7. As seen in the table, the remaining discriminators, variables 5-12 and 15, behave similarly to variable 4 in that once a class difference appears it remains persistent to roughly the end of the run at time point 82; the time window starting point, however, varies from variable to variable. Only variable 7 differs significantly from the other discriminators in that its time window ends at time point 70. Looking at Figures 2.8 (a)-(i), after an initial period of overlap between the classes, the difference in the means becomes noticeable after time point 40. This suggests that if the modeling is limited to using single variables as model inputs any time period before 40 is unlikely to be able to distinguish between the 2 classes.

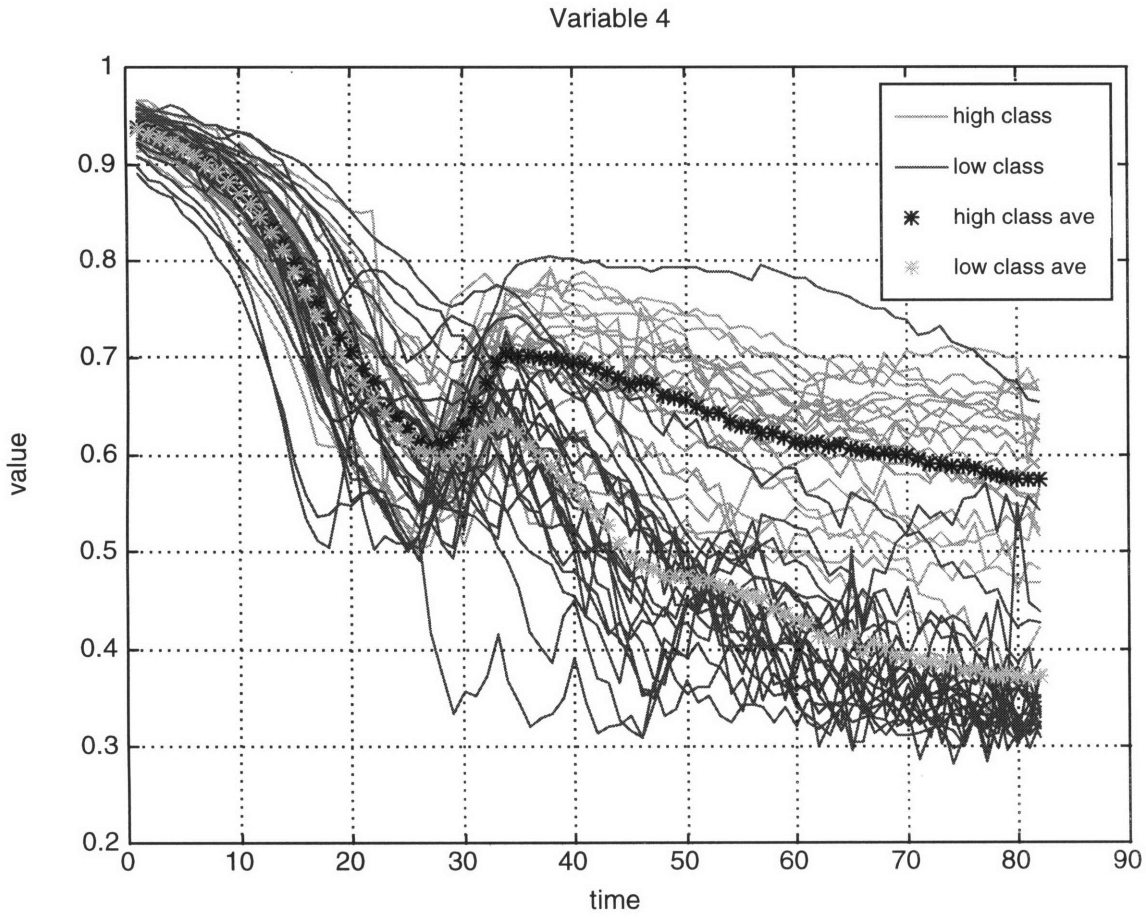


Figure 2.6 Time profiles of discriminator, Variable 4.

Table 2.4 Sample MHT output for variable 17

Calculated F-value	Observed F-value	Time of Discrimination	F-Ratio
1.11	4.11	n.d.	0.27
2.23	4.08	n.d.	0.55
3.16	4.07	n.d.	0.78
3.01	4.08	n.d.	0.74
1.63	4.07	n.d.	0.40
3.18	4.11	n.d.	0.77
•	•	•	•
•	•	•	•
•	•	•	•
1.67	4.10	n.d.	0.41
2.00	4.11	n.d.	0.49
0.91	4.12	n.d.	0.22
0.88	4.12	n.d.	0.21
1.18	4.14	n.d.	0.28
0.52	4.17	n.d.	0.13
•	•	•	•
•	•	•	•
•	•	•	•
1.43	4.13	n.d.	0.35
1.40	4.17	n.d.	0.34
1.53	4.20	n.d.	0.36
0.30	4.18	n.d.	0.07
0.39	4.20	n.d.	0.09
1.12	4.18	n.d.	0.27

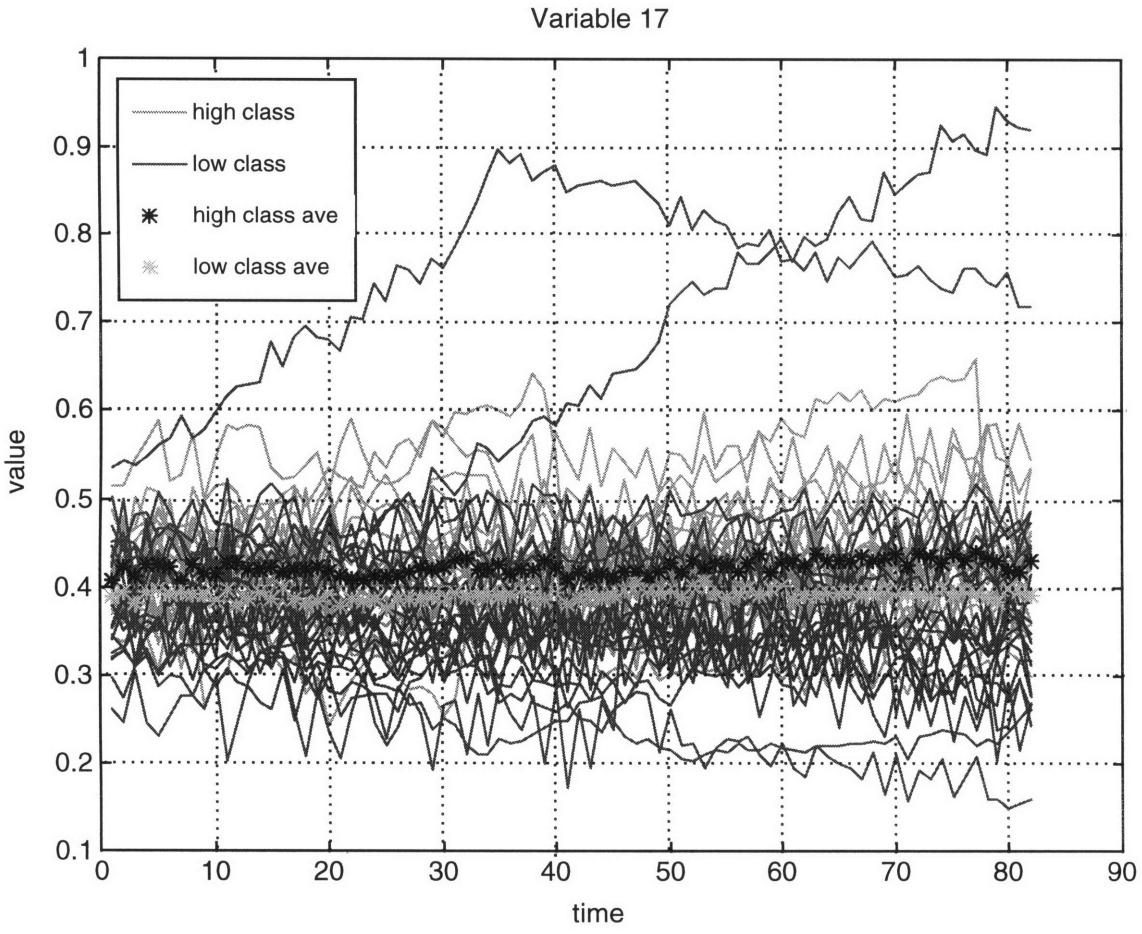


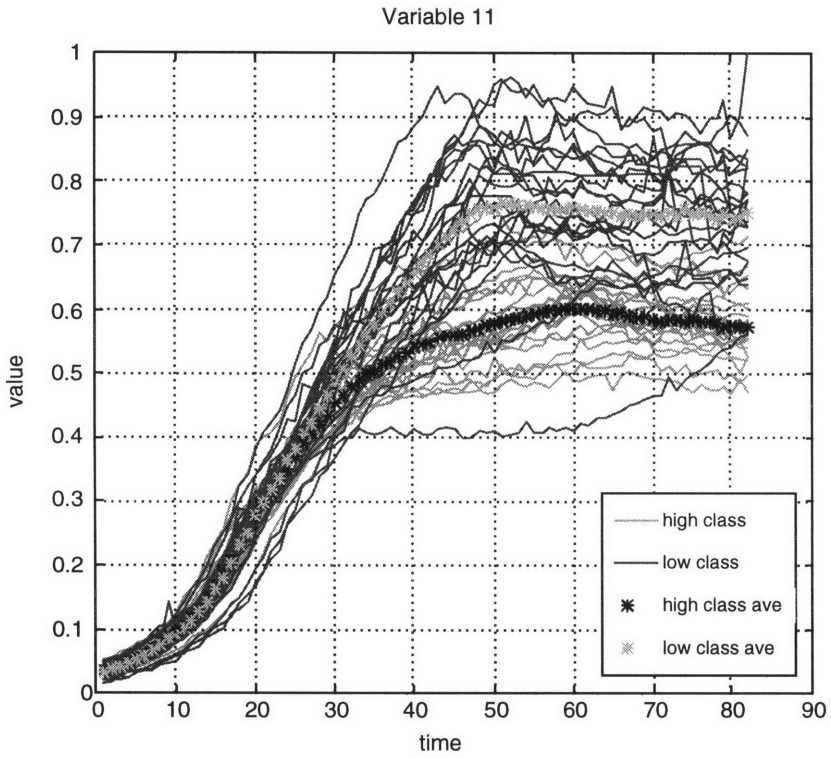
Figure 2.7 Time profiles of nondiscriminator, Variable 17.

Table 2.5 Univariate MHT results

Variable	Discriminating Time Window	Average F-ratio for all times*
Discriminators		
4	32-82	7.52
11	31-82	5.65
9	32-82	4.7
5	33-82	3.33
12	42-82	2.79
10	42-82	2.77
15	40-82	2.67
8	43-82	2.6
6	45-82	2.0
7	37-70	1.78
Nondiscriminators		
17	n.d.	0.33
13	n.d.	0.30
14	n.d.	0.27
16	n.d.	0.23

* Note: the F-ratio must exceed 1 to be discriminating but as mentioned previously this column is the average F-ratio taken over all the time points. The average is used because it takes into account periods where the variable is not discriminating. For example, a variable that has a F-ratio of 20 but for only one time point and is close to 0 for the other times may not be considered as discriminating as a one whose F-ratio is 10 but goes for 10 time points and the average reflects this disparity. For the same time window, the former would be ~ 2 and the latter 10.

(a)



(b)

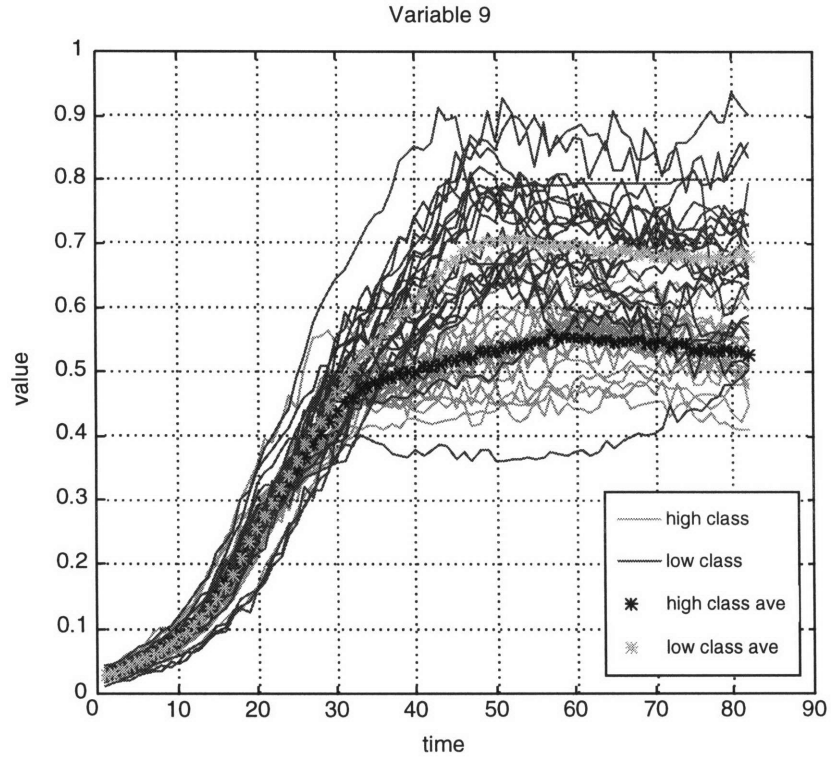
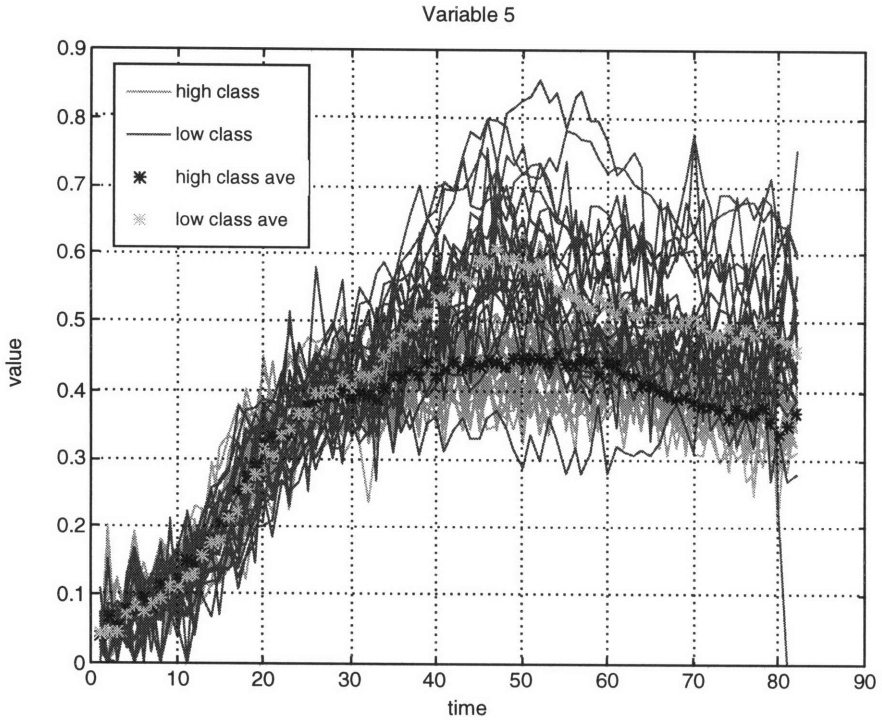


Figure 2.8 (a)-(b) Time profiles of discriminators.

(c)



(d)

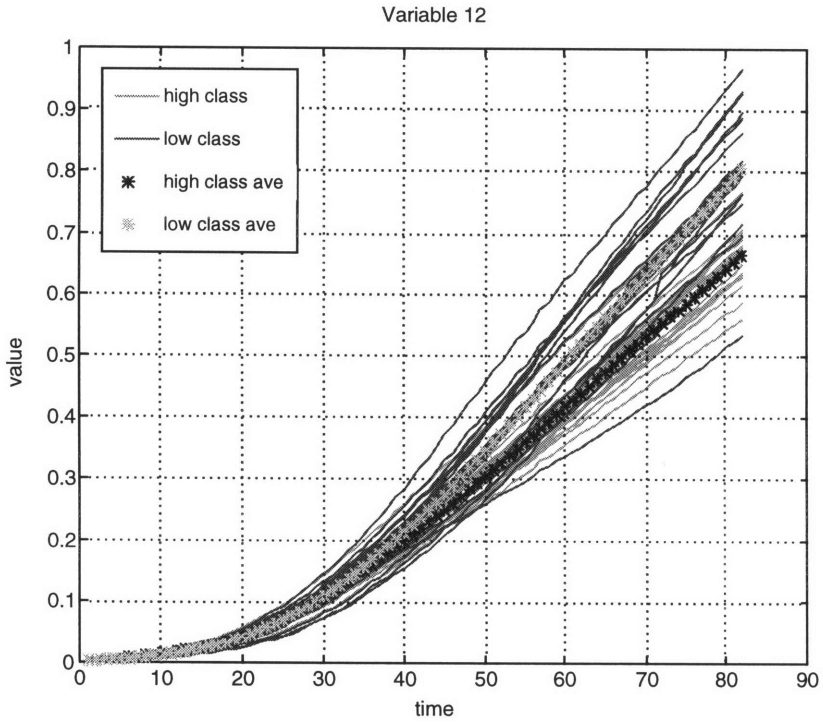
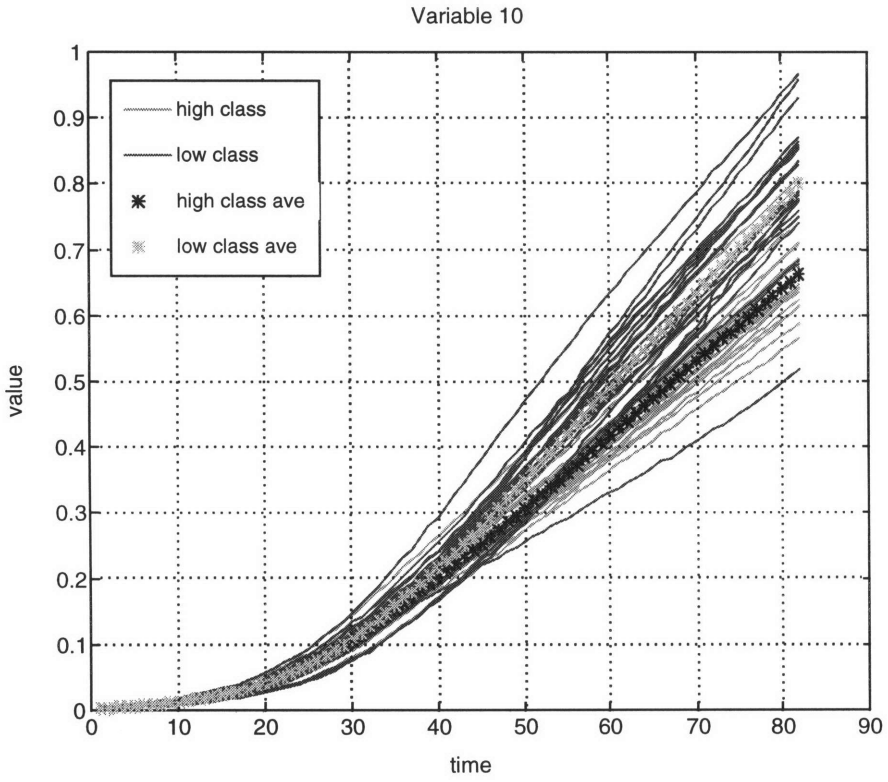


Figure 2.8 (c)-(d) Time profiles of discriminators.

(e)



(f)

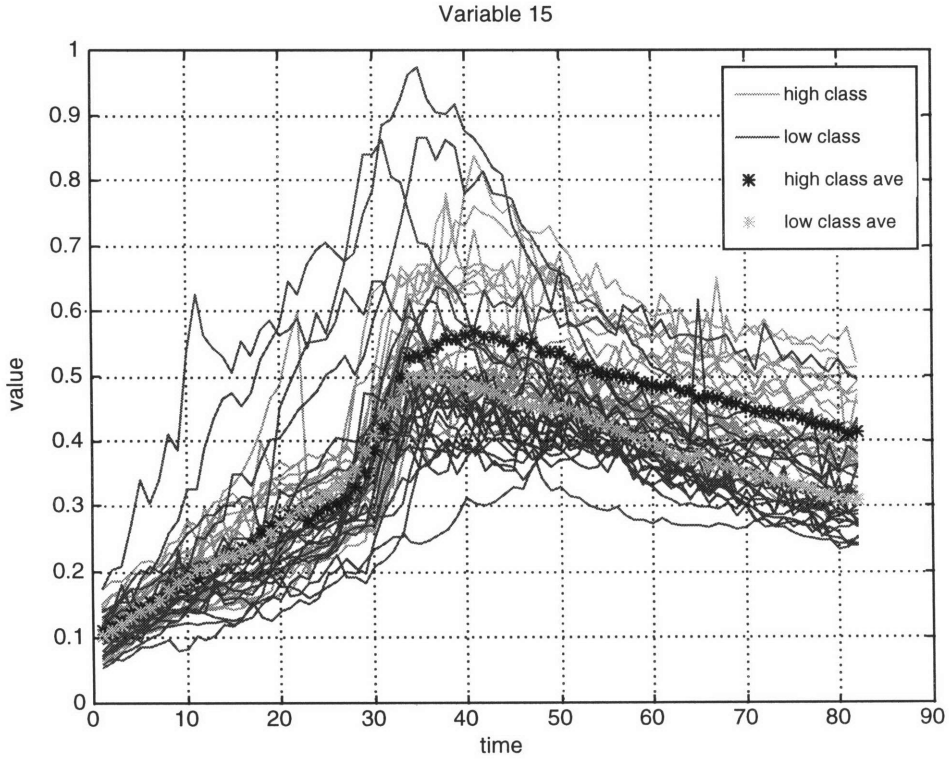
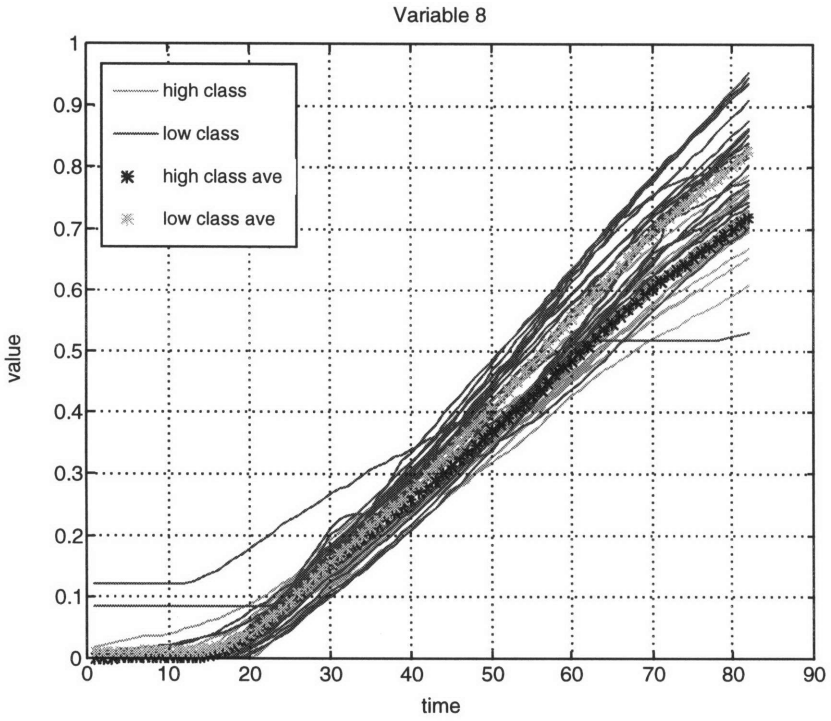


Figure 2.8 (e)-(f) Time profiles of discriminators.

(g)



(h)

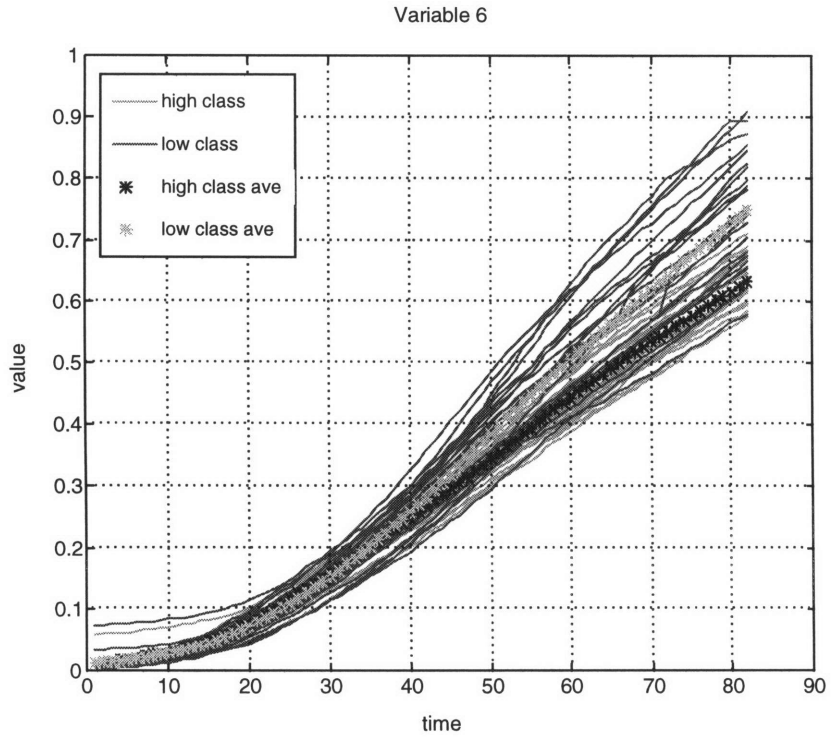


Figure 2.8 (g)-(h) Time profiles of discriminators.

(i)

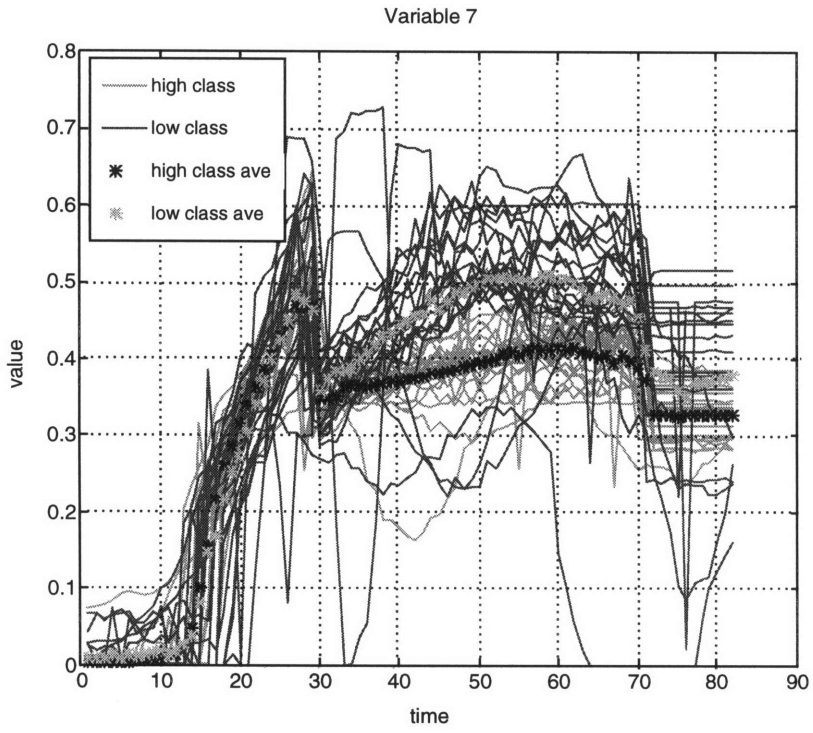


Figure 2.8 (i) Time profiles of discriminators.

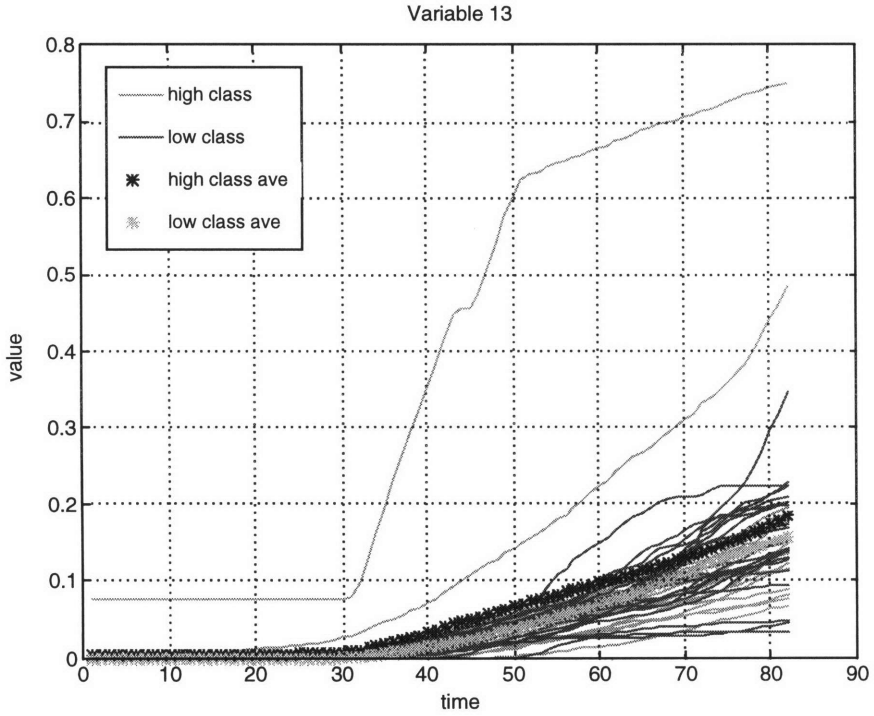
If an earlier time period is needed, two possibilities exist: 1) a more thorough analysis can be performed to include variable combinations, which will be discussed later in this section; or 2) other more sensitive measurements need to be made if earlier detection is the goal. The second point is important because this suggests that the current measurement set may not be sufficient for modeling purposes.

In contrast, the other nondiscriminators, variables 13, 14, and 16, appear to behave the same in both high and low classes. There appears to be no time windows where there is a significant difference and this is seen visually in Figures 2.9 (a)-(c). These findings suggest that it is highly improbable that including these variables in a model will increase sensitivity to class differences. In view of the noisiness of some of these measurements, variables 16 and 17 in particular, overall model performance may be compromised if reconstructing these variables accurately is part of the algorithm.

2.4.1.2 Combination of Variables Effect

Unfortunately, it is not possible to generalize from single variable (univariate) results to combinations of variables (multivariate). The time windows predicted by the combination can differ from the ones observed by each member of the combination. The reason is that the effect of variable interactions is introduced when combinations of variables are considered. As a result, class differences can appear at different time periods. This observation has been confirmed when using the same data as previously but now focusing on variable combinations; in some cases, an earlier time window of discrimination appears - starting roughly from the late 20's, see Table 2.6. For this particular data set, the time window appears earlier than those observed by the individual variables themselves but this is not always guaranteed. For a thorough analysis, all combinations of variables have to be examined but the drawback lies in the sheer number of groupings to be explored. In this case study, the analysis stops arbitrarily at 4-variable combinations. Even so, this requires examining 1,001 combinations when 14 variables are present. This number does not include the pair and triplet-wise combinations, which if considered pushes the total number to 1,455. Table 2.6 summarizes only the top 3 results. This ranking is based on the highest average F-ratio from each variable grouping, beginning with pairs and ending with

(a)



(b)

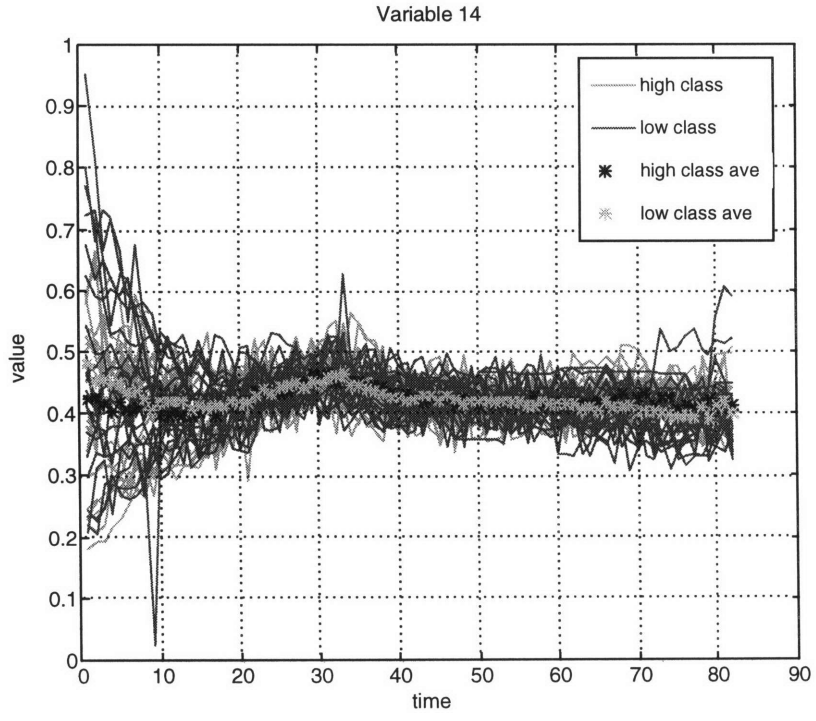


Figure 2.9 (a)-(b) Time profiles of nondiscriminators.

(c)

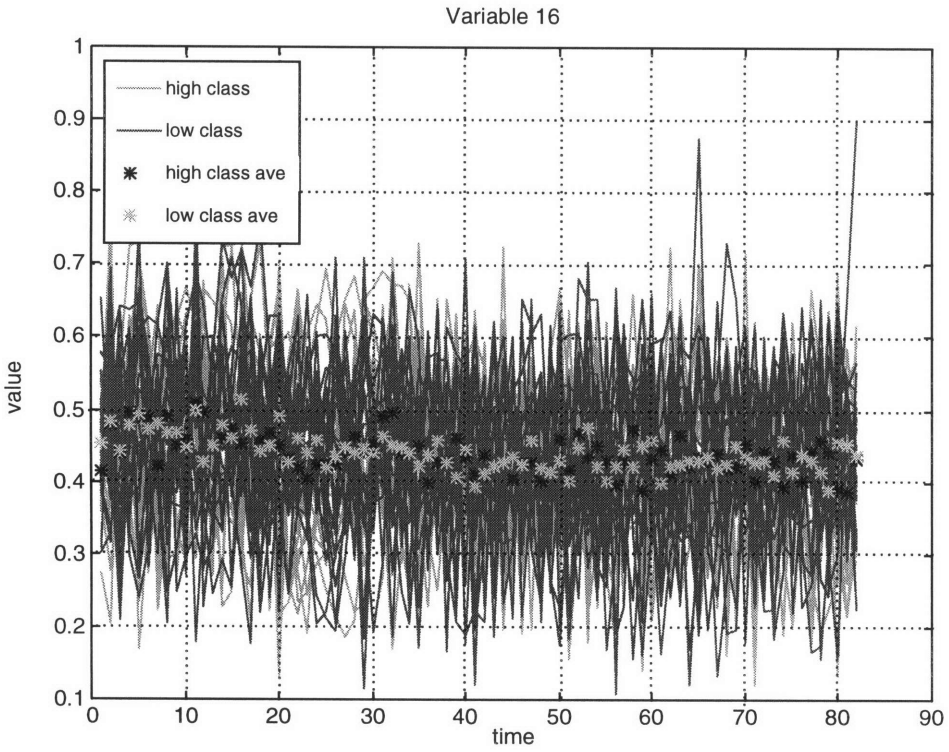


Figure 2.9 (c) Time profiles of nodiscriminators.

Table 2.6 Summary of MHT multivariate results with Case 1 data.

Variable Combinations	Discriminating Time Window	Average F-ratio for all times
4,7	32-82	5.15
4,5	33-82	5.10
4,11	27-82	5.05
4,7,15	29-82	4.18
4,7,11	31-82	4.18
4,5,7	33-82	4.15
7,8,11,12	24-82	3.97
4,8,11,12	28-82	3.82
4,7,11,12	28-32	3.69

quadruples. Though not presented here, it is important to note that for variable combinations if the nondiscriminating variables are not deleted from the group under inspection it becomes difficult to interpret the results. A triplet containing a discriminating pair and one nondiscriminator may predict time windows similar to the one proposed by looking only at the discriminating pair. Hence, there is no inherent advantage value to using this particular triplet over the pair.

2.4.1.3 Comparison with Another Classification Method - dbminer

To see how MHT performs relative to another discrimination methodology, the same data was analyzed using dbminer© - a software tool developed by Professor Greg Stephanopoulos' group at MIT for database analysis (Stephanopoulos, et. al 1997). One of dbminer's many functions is the ability to identify discriminating variables in the form of decision trees. Decision trees determine which variable(s) and variable attributes are the most important in terms of their correlation with process outcome (Quinlan, 1986; Saraiva and Stephanopoulos, 1992). At each tree node, the samples in the group being considered are divided into smaller groups, which represent branches of the tree. The decision to choose which variable and the particular value of that variable to form a node is due in part to considering the information content of every possible partition that can be generated. The information content is determined by using Shannon's entropy formula, equation 17. The information content after a group has been split is shown in equation 18. For each variable, the difference in information content before and after the split is calculated and the conditions, variable attribute and value, which maximize the change in information content is selected as the node in the decision tree.

Mathematically, Shannon's entropy formula and the accompanying change in information content after a group split is shown below, adapted from Bakshi, 1992.

$$I(N_{C_1}, N_{C_2}, ..., N_{C_n}) = -\sum_{i=1}^{n} \frac{N_{C_i}}{N} \log_2\left(\frac{N_{C_i}}{N}\right) \qquad (17)$$

$$E(G_1, G_2) = \frac{N_{G_1}}{N} I(N_{C_1,G_1}, N_{C_2,G_1}, \ldots, N_{C_i,G_1}) + \frac{N_{G_2}}{N} I(N_{C_1,G_2}, N_{C_2,G_2}, \ldots, N_{C_i,G_2}) \quad (18)$$

where:

I:	information content
N_{C_i} :	number of samples in class i
N:	total number of samples from all classes
N_{C_i,G_j} :	number of samples of class C_i in group G_j
E:	information content after splitting into groups, G_1 and G_2

Dbminer searches through all the selected variables to first isolate the best separation as dictated by the largest change in information content; this becomes the top node. The procedure is then repeated on the remaining samples in the branches but this time considering only those variables not selected in previous nodes. This process is continued until each branch consists of members of the same class.

Figure 2.10 displays the decision tree results from dbminer using all 14 variables. The way to interpret the flowchart is that each node is represented by a box. The numbers below H/L are the number of high and low lots, respectively. Variable name lists the variable under inspection and the value listed is the breakpoint where lots that had values less than the breakpoint fall into the left branch with the remainder going to the right. Ideally, the best classifier is one where all the high lots (22) and all the low lots (23) fall into 2 different branches, but as seen in Figure 2.10 this is not the case.

What is interesting to note, however, is that dbminer achieves the best classification with variable 4, consistent the results from the MHT. The program finds at best that it can group all 22 of the high lots into the right branch with 21 of 23 low lots in the left. The tree is stating that all of the high lots in the right branch have the property that their variable 4 has a value greater than 0.5178 at time point 49. However, 2 of the 23 low lots also share this characteristic and cannot be distinguished on the basis of variable 4 alone. To separate the remaining 2 low lots from the 22 high, variable 7 is needed and this is shown in the second node on the right side of the tree. All the high

lots have values greater than or equal to 0.32 at time point 55 for variable 7 whereas the 2 low lots do not and are therefore segregated to the left branch. It should be emphasized that the decision tree places the conditions (variable/value/time point) that gives the best class separation at the top node and eliminates that variable for further consideration when it generates the rest of the tree.

Table 2.7 summarizes dbminer results if only one variable is considered at a time - in effect the discrimination power of each measurement. Variables that are strong in discrimination, 4, 7, 9, and 11, tend to have large classification percentages (>90%) in both branches indicating good separation. The nondiscriminators, variables 13, 14, 16, and 17, appear to be poor at classifying one of the classes. Recall, that for excellent separation each branch should contain only members of 1 class and none of the other.

The point of interest is that MHT's single variable (univariate) results are comparable with dbminer which follows a different approach towards evaluating the discrimination power of variables. Dbminer suggests that variables 4, 7, 9, and 11 are strong discriminators. MHT proposes the essentially the same though it does not rank variable 7 to be as strong a discriminator as in the decision tree. This may be explained by the fact the ranking in MHT is based on the average F-value for all times which includes discriminating as well as nondiscriminating periods (see note under Table 2.5 for detailed explanation). With the case of variables 13, 14, 16, and 17, all of which MHT predicted as nondiscriminating, the decision tree reports similar findings judging from poor class separation that is achieved when using those variables. There are some "discrepancies," however. The decision tree does not rank variable 6 as a discriminator considering it classified 9 out of 23 lows in the same branch as all 22 highs. MHT lists variable 6 as a discriminator, however, but as a weak one. Another slight discrepancy is the difference in time points of discrimination - variable 4 being

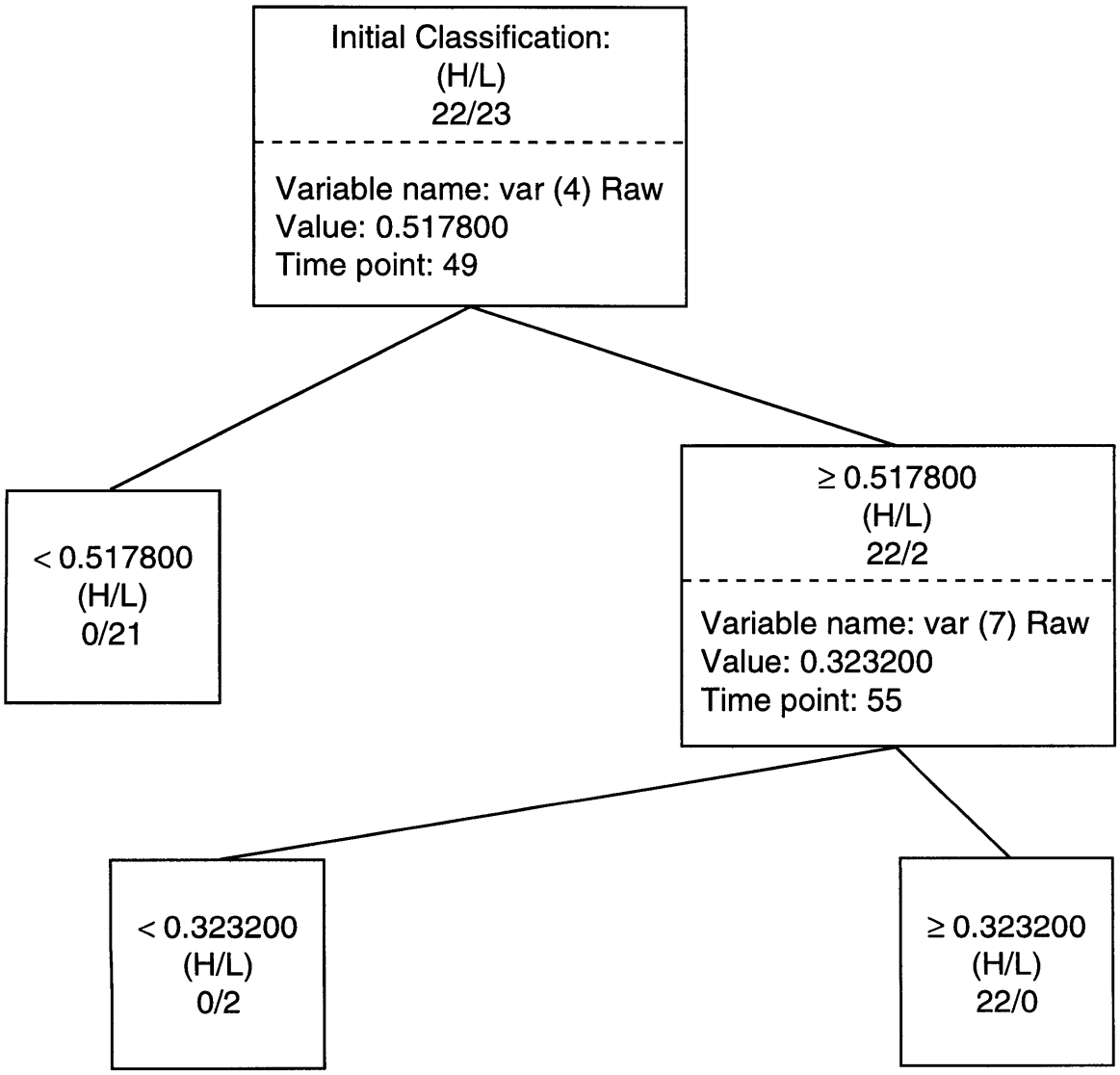


Figure 2.10 Decision Tree for case study 1 using 14 variables.

Table 2.7 dbminer results with Case 1 data using 1 variable and focusing on the first node of the decision tree.

Variable	Time point of Segregation	Left Branch (H/L)	Right Branch (H/L)	% Classification Based on Separation (H/L)
4	49	0/21	22/2	100/91
5	69	22/5	0/18	100/78
6	73	22/9	0/14	100/61
7	62	21/2	1/21	95/91
8	78	21/6	1/17	95/74
9	48	21/2	1/21	95/91
10	74	21/4	1/19	95/83
11	78	21/2	1/21	95/91
12	79	20/3	2/20	91/87
13	24	13/22	9/1	59/97
14	78	0/8	22/15	100/65
15	57	1/18	21/5	95/78
16	21	8/0	14/23	64/100
17	11	0/10	22/13	100/57

discriminating at time 32 in MHT and 49 in dbminer - reflects dbminer's search for the best separation whereas MHT identifies all time points where the classes are different. Overall, these disagreements are minor and the main results still suggest that both MHT and decision trees support each other's findings for the case of single variables.

Unfortunately, the decision tree approach cannot handle variable combinations greater than pairs. So further comparisons cannot be made for the combinations of variables (multivariate) case. For the case of pairs, taking the ratio of the individual variables can replace pairwise information but may cause problems if one of the 2 variables is 0.

2.4.1.4 Summary of Findings

In summary, the following conclusions are drawn from the analysis of this case study:

1) From the measurements collected, both high and low classes behave similar to one another early in the run but diverge after time point 31 for the single variable case and 24 for the variable combinations. Hence, it is extremely unlikely that modeling the first 20 points is unlikely to provide any discrimination.
2) Not all the measurements are useful. Variables 13, 14, 16, and 17 behave similar enough in both classes to be indistinguishable from a statistical standpoint.
3) Variables 4,5 9, and 11 by themselves and variable combinations involving 4,7, 11 and 12 appear to be good discriminators.
4) Class differences, once they appear, tend to be stable and persist to the end.
5) MHT's selection of discriminating and nondiscriminating variables compare favorably with those from decision trees which are based on a different criteria using Shannon's entropy formula.

2.4.2 Case Study 2

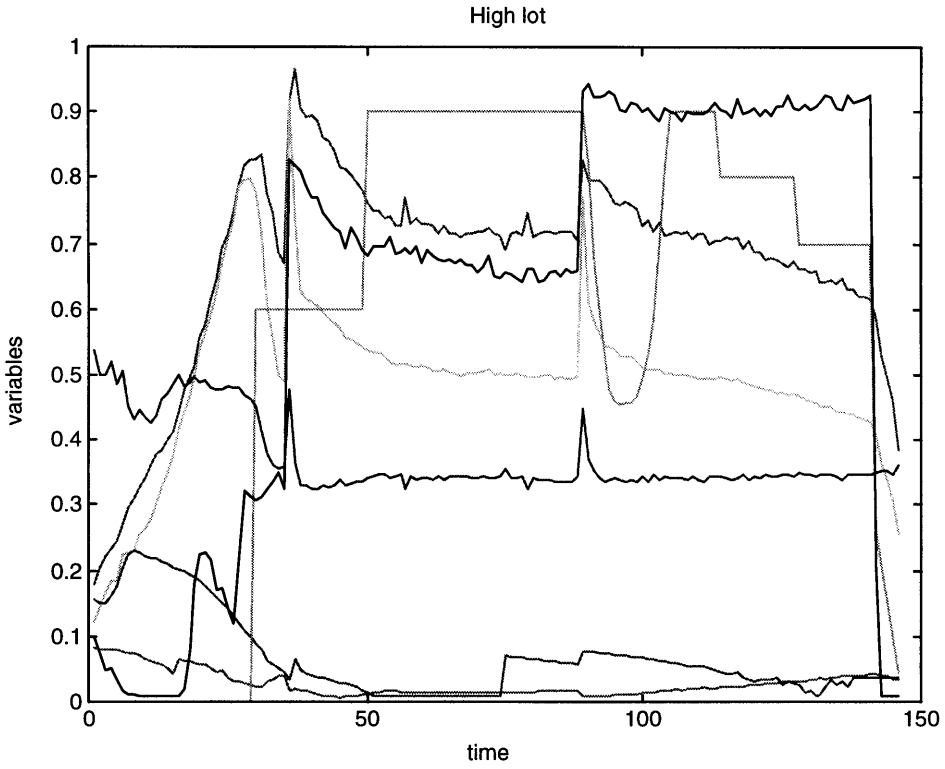
Type: industrial fermentation

Mode:	fed-batch
Run length:	146 time points
Number of variables:	6 (labeled variables 3 to 9, variable 7 ignored)
Number of classes:	2
Lots/class:	10 for high and 6 for low

2.4.2.1 Single Variable Effects

MHT is now applied to a different data set. The search for discriminators begins with applying MHT to the individual variables. Figure 2.11 (a)-(b) displays a typical lot from each classes. For this analysis, all 16 lots are used and the results are shown in Table 2.8. Again as in the first case study, it is observed that some variables are discriminating, 3, 4 , and 5, while others, variables 6, 8, and 9, are not; also, the entire time profile is again found not to be discriminating but rather certain sections of it. One noticeable difference from the previous case study, however, is the observation of a time window which appears early in the process and then disappears, time points 25 to 66 for variable 4 and 40 to 66 for variable 5. This strongly indicates that there is a short window in time where early discrimination is possible and it is this region that should be modeled if discrimination is the goal. By contrast, in the first case study roughly any time point after 40 was discriminating for most variables. Another difference is that variable 3 has a period of discrimination that differs from the other discriminators, variables 4 and 5, producing a time window 94 to 111. Such results suggest the creation of 2 separate models to cover the different time periods if the analysis is restricted to just considering individual variables.

a)



b)

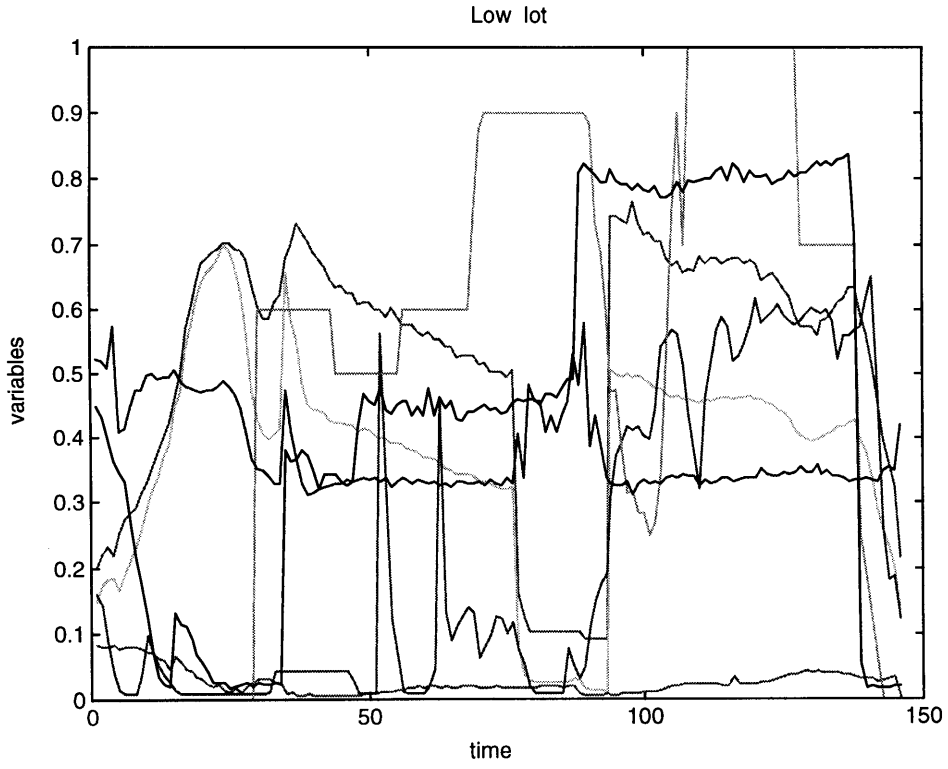


Figure 2.11 a) Typical Case study 2 High lot; b) Typical Case study 2 Low lot.

Table 2.8 Univariate MHT results with Case 2 data

Variable	Discriminating Time Windows	Average F-ratio for all times
Discriminators		
4	25-66	1.46 (3.19) *
5	39-66	0.75 (1.96) *
3	94-111	0.64 (1.60) *
Nondiscriminators		
9	n.d.†	0.38
8	n.d.†	0.33
6	n.d.†	0.19

* Note: the F-ratio must exceed 1 to be discriminating but, as mentioned previously, this column is the average F-ratio taken over all the time points, discriminating and nondiscriminating. Because the window of discrimination for these data is small relative to the length of the entire run, the F-ratio values tend to be lower than expected. The numbers in parentheses reflect the average F-ratio over the discriminating time period, which as expected is larger than 1.0.

† There are time points where discrimination was achieved, however, these consisted mostly of isolated points. For simplicity n.d. is used.

2.4.2.2 Combination of Variables Effect

The results of evaluating combinations of variables are summarized in Table 2.9. Because only 3 of the variables appear to be discriminating the number of combinations to consider are lower than in the first case study. The nondiscriminating variables have been removed from consideration. As seen in the single variable analysis, it is observed again that there are certain windows in time where discrimination is possible.

2.4.2.3 Comparison with dbminer

The single variable results from MHT are also compared with the decision trees obtained from dbminer, see Table 2.10. Again, agreement is observed between the 2 different techniques in determining the discriminators. Total class separation is achieved in the first node using either variable 4 or 5 and good separation with variable 3, all variables MHT suggested as discriminating. With variables 6, 8, and 9, however, there appears initially to be some discrepancy. While these 3 variables are not as discriminating as 3, 4, and 5, the decision tree suggests that good separation should still be possible in contrast to MHT which lists them as nondiscriminating. One possible reason the 2 algorithms disagree on the selection of nondiscriminators is that decision trees are designed to find the best separation possible with the implicit assumption that there is an inherent difference in the data whether it truly exists or not. Hence, they are not well-suited to identifying nondiscriminators in contrast to MHT which is not constrained to find the best but to simply determine if a difference exists or not. Another possibility is the different way MHT and decision trees interpret their results. In MHT, the emphasis is not simply to find 1 or 2 time points where a class difference is detected but to find a stable region where this difference can be observed.

Table 2.9 Summary of MHT multivariate results with Case 2 data.

Variable Combinations	Discriminating Time Windows	Average F-ratio for all times
3,4	23-66, 90-111	1.34
4,5	24-66, 91-111	1.13
3,5	39-62, 77-93	0.791
3,4,5	23-66, 77-85, 89-102	1.39

Table 2.10 dbminer results with Case 2 data using 1 variable and focusing on the first node of the decision tree.

Variable	Time point of Segregation	Left Branch (H/L)	Right Branch (H/L)	% Classification Based on Separation (H/L)
3	14	9/0	1/6	90/100
4	27	0/6	10/0	100/100
5	40	0/6	10/0	100/100
6	3	1/5	9/1	90/83
8	17	10/1	0/5	100/83
9	61	8/0	2/6	80/100

Additionally, for MHT it is not sufficient that a difference simply exist, the difference must be statistically significant. As a result, MHT can be viewed as a more conservative approach to selecting discriminators. By comparison in decision trees, the classification is based on whether the lot exceeds a particular value at **one** time point. There are **isolated** time points for variable 8 and 9 where MHT has detected class differences, but as shown in Table 2.8 the time window is listed as n.d. for simplicity's sake. Incidentally, for variable 8, time point 17 is listed by MHT as discriminating in the actual, more detailed listing. As for variable 9, at the decision tree's selection of time point 61, the observed F-value (2.97) approaches the tabulated F-value (4.6) more than at other time points but this still fails the statistical test and is listed as nondiscriminating by MHT. MHT found no discriminating time points for variable 6 but this is partially reflected in the decision tree by variable 6's relatively poor classification ability.

2.4.2.4 Summary of Findings

Aside from identifying discriminating variables, (3, 4, and 5) and time windows (44-66 and 94-111), other additional findings from this case study are summarized below:

1) Time windows are not as stable as in the previous case study and more than one discriminating time period exists. Hence, modeling the entire time-profile may be counterproductive if only sections of the data are important. The discriminating regions should be the focus of the modeling effort.
2) Good agreement between dbminer and MHT on identifying variable discriminators.

2.5 Conclusion/Discussion

Mean hypothesis testing is an important pre-modeling step if insight into the nature of the data that is to be modeled is to be obtained. MHT allows the modeling effort to focus on sections of the data that are relevant to discrimination- specifically, time windows and variables. From the analysis of the case studies presented, several observations are drawn:

1) Time windows are important. It is not necessary to model the entire time course, just the discriminating regions. This can greatly reduce the time needed to fit models to the data.
2) Not all variables are discriminating. The emphasis should be placed modeling the discriminators rather than all the available data. In combination with 1, it is important to note that discriminating variables may have only windows in time where they are effective.
3) Variables identified by MHT are in agreement with those discovered by decision trees, thus providing independent confirmation of the methodology. Visual inspection of the variables themselves also confirm the discriminating nature of MHT's selections.
4) Variable combinations can provide more insight than looking at variables by themselves. Hence, MHT is able to consider more possibilities than decision trees, which can analyze only up to pairwise combinations.

While the emphasis has been on pre-modeling applications, MHT can also play a vital role as a tool for developing rules for expert systems along the lines for decision trees. Information about time windows and discriminators can be combined with a histogram analysis of the range of values displayed by the discriminators. The histograms can reveal the range of values displayed by members of a class at a particular time window for a specified variable. This information can be converted into a rule similar to information provided by decision trees- specifying that x out of y lots could be segregated if its value was above or below the value z.

2.6 References:

Bakshi, B. (1992)

Multi-resolution methods for modeling, analysis, and control of chemical process operation, Sc.D. thesis, Massachusetts Institute of Technology, Department of Chemical Engineering

Coomans, D, Massart, D.L., and Broeckaert (1981)
"Potential methods in pattern recognition. Part 4: A combination of ALLOC and statistical linear discriminant analysis," *Anal.Chim.Acta*, 133, p 215

Kell, D. B. and Sonnleitner, B (1995)
"GMP - Good Modelling Practice: an essential component of Good Manufacturing Practice", *TIBTECH,* vol 13, p 481-492

Mardia, K. V., Kent, J.T., and Bibby, J. M. (1979)
Multivariate Analysis. Academic Press. San Diego

Quinlan, J. R. (1986)
"Induction of decision tress," *Machine Intell.*, vol 11(7), p 710-732

Saraiva, P. and Stephanopoulos, G (1992)
"Continuous process improvement through inductive and analogical learning," *AIChEJ*, 38(2), p 161

Stephanopoulos, G. Locher, G, Duff, M, Kamimura, R, and Stephanopoulos, G (1997)
"Fermentation Database Mining by Pattern Recognition," *Biotech Bioeng,* vol vol 53, 5, p 443- 452.

2.7 Appendix

Data Sorting:

function [strtpt,strtcrt,LotInfo,newData,LotSize2,LotInfoA,rejcts] = DPrep(Data,lbl,time,cls,nouse,perc)

```
% Date: 8/14/96, Roy Kamimura, copyright (c) by MIT
% DEFINITION:
% Standardizes lots to start at the same time point as well as have the same size.
% Recommend using Tsort to identify missing time points in the database prior to
% using this function.
```

```
% Inputs
% perc = 0.05 usually
% nouse = lots that are to be deleted for other reasons
% cls = classifications assigned to lots in order of lots received

% Output of DatSum
% 1 - relative lot number
% 2 - lot size
% 3 - initial point
% 4 - final point
% 5 - lot label (not class)

LotI = DatSum(Data,lbl);
LotI = [LotI cls];
nLots1 = row(LotI);
LotInfo = [];

if (col(nouse) == 0),
  nouse = zeros(1,nLots1);
end

for ia = 1:nLots1,
   aa = col(find(nouse == ia));
   if (aa == 0),
      LotInfo = [LotInfo; LotI(ia,:)];
   end
end


InitLsize = min(LotInfo(:,2));
nLots = row(LotInfo);


addData = [];

ed = 0;

% Sort data into one matrix containing only times.  Columns
% of this matrix are the time elements.

for i = 1:nLots,
   strt = LotInfo(i,3);
   ed1 = strt + InitLsize - 1;
   addData = [addData Data(strt:ed1,time)];
end


% Find the max time point for starting

strtcrt = max(addData(1,:));

for j = 1:col(addData),
   strtpt(1,j) = min(find(addData(:,j) >= strtcrt));
```

```
end

% Find the new starting point of all lots

LotI3 = LotInfo(:,3) + strtpt' - ones(row(LotInfo),1);

% Revise lot size to consider new starting point

Lotsizes = LotInfo(:,4) - LotI3 + 1;


LtSrt = sort(Lotsizes);

for jj = 1:nLots,
   tst = LtSrt(jj,1);
   chk = row(find(LtSrt <= tst))/nLots;
   if (chk >= perc),
     LotSize2 = LtSrt(jj,1);
     break
   end
end

newData = [];

% Now need to reject lots that do not meet minimum lot length

noRejIndx = find(Lotsizes >= LotSize2);

LkIndx = zeros(nLots,1);

LkIndx(noRejIndx,1) = ones(row(noRejIndx),1);

for k = 1:nLots,
   if LkIndx(k,1) == 1,
     indx1 = LotI3(k,1);
     indx2 = LotI3(k,1) + LotSize2 - 1;
     newData = [newData;Data(indx1:indx2,:)];
   end
end

rejcts = [LotInfo(find(LkIndx == 0),5) LotInfo(find(LkIndx == 0),6)];

LotInfoA = DatSum(newData,lbl);

LotInfoA = [LotInfoA LotInfo(noRejIndx,6)];
```

function [newD] = DExt(D,opt,begmrk,endmrk,mrkpost,LotS)

```
% DATE: 9/23/96, Roy Kamimura, copyright (c) by MIT

% DEFINITION:
% Extract out from all the matrices a particular
% subset in question and lump them all into one matrix.
```

```
% INPUTS:
% D = data matrix containing all values.
% opt = specify whether using position or actual values
%       0 = position, 1 = actual values
% begmrk = starting point
% endmrk = end point
% mrkpost = column denoting marker
% LotS = standard data lot size

if opt == 1,
  BegMrk = find(D(:,mrkpost) == begmrk);
  EndMrk = find(D(:,mrkpost) == endmrk);
  MrkInd = [BegMrk EndMrk];
end

if opt == 0,
  offst = begmrk - 1;
  for i = 1:row(D)/LotS,
     StrtMrk(i,1) = 1 + (i-1)*LotS;
     intrvl = endmrk - begmrk;
     BegMrk(i,1) = StrtMrk(i,1) + offst;
     EndMrk(i,1) = BegMrk(i,1) + intrvl;
     MrkInd = [BegMrk EndMrk];
  end
end

newD = [];

for j = 1:row(D)/LotS,
   addD = [D(MrkInd(j,1):MrkInd(j,2),:)];
   newD = [newD;addD];
end
```

function [T,Tavail,gapsize,rejcts] = Tsort(D,lbl,t,crt)

```
% Date: 7/31/96, Roy Kamimura, copyright (c) by MIT
% DEFINTION:
% Create master time list from all data.  Identify which lots are
% missing which time points.

% INPUTS:
% D = matrix containing all the lots
% t = column denoting time variable
% lbl = column denoting lbl variable
% crt = number of missing points acceptable

% OUTPUTS:
% T = all available time points possible
% Tavail = matrix where row corresponds to value of time point
%   in T and col denotes relative lot location that has that
%   time point (1 = present, 0 = not present)
% gapsize  = summary on gaps in data
```

```
%   1st row = size of largest gap
%   2nd row = number of gaps, regardless of size


% ASSUMPTIONS:
% Time points do not differ in value from one lot to the next
% though some entries might be missing

% Sort all available time values in ascending order
allTval = sort(D(:,t));

% Keep only 1 value of each time point

T = allTval(1,1);

for i = 2:row(allTval),
  old = allTval(i-1,1);
  new = allTval(i,1);
  if old ~= new,
    T = [T;new];
  end
end

% T now contains all the possible time points
% Key is to determine what time points are present in each lot

Lot = DatSum(D,lbl);

% Lot(:,1) =  lot number (position of lot relative to all the lots)
% Lot(:,2) =  lot size
% Lot(:,3) =  initial point of lot
% Lot(:,4) =  final point of lot
% Lot(:,5) =  lot label

% Tavail denotes the matrix that lists in col 1 all the time values
% possible and each subsequent column denotes the availability of those
% values for each lot. 0 == value not present; 1 value present

Tavail = zeros(row(T),row(Lot));

for j = 1:row(Lot),
  Dt = D(Lot(j,3):Lot(j,4),t);
  for k = 1:row(T),
    chk = find(Dt == T(k,1));
    if row(chk) ~= 0,
      Tavail(k,j) = 1;
    end
  end
end

% Now calculate the size of the gaps and their frequency.
% First row denotes max gap observed.
% Second row dentoes total number of missing entries (could have
% many small gaps instead of one large gap)
```

```
% Third row picks larger of the 2.

for l = 1:col(Tavail),
  nomiss = find(Tavail(:,l));
  gaps = diff(nomiss) - ones(row(diff(nomiss)),1);
  gapsize(1,l) = max(gaps);
  gapsize(2,l) = sum(gaps);
  gapsize(3,l) = max(gapsize(1,l),gapsize(2,l));
  clear nomiss
end

% Rejcts is the lots with unacceptably high number of missing
% points.  Crt sets number of missing points acceptable.

rejcts = find(gapsize(3,:)>crt);
```

Multivariate mean hypothesis testing

function [F] = multiT2(H,L,Dsize)

```
% Date: 11/14/95, Roy Kamimura, copyright (c) by MIT
% DEFINITION:
% For 2 groups of data, H and L, this program determines whether their means
% are statistically different.  The test is taken from p 144 of
% Mardia, Kent, and Bibby (1994/1979) and assumes that the covariances of
% the 2 groups are different from one another.

% F(:,1) = calculated F-value
% F(:,2) = statistical F-value (value exceeded to reject Ho where
% Ho is means are equal)
% F(:,3) = time point where means are different

[rh, ch] = size(H);
[rl, cl] = size(L);

nH = rh/Dsize;
nL = rl/Dsize;
n = nH + nL;
fH = nH - 1;
fL = nL - 1;

F = zeros(Dsize,3);
%F(:,2) = ones(Dsize,1)*finv(0.95,ch,n - ch + 1);

for i = 1:Dsize,

  [H1,Hm,HS] = VsortT(H,Dsize,i);
  [L2,Lm,LS] = VsortT(L,Dsize,i);

  X1 = Hm';
  X2 = Lm';
  dif = X1-X2;
```

```
  U1 = HS/fH;
  U2 = LS/fL;
  U = U1 + U2;

  T2 = dif'*inv(U)*dif;

  A1 = (dif'*inv(U)*U1*inv(U)*dif/(T2))^2;
  A2 = (dif'*inv(U)*U2*inv(U)*dif/(T2))^2;

  ff = round(1/( (A1/fH) + (A2/fL)));

  F(i,1) = (T2)*(ff - ch + 1)/(ff*ch);
  F(i,2) = finv(0.95,ch,ff - ch + 1);
end

for j = 1:Dsize,

  if (F(j,1) > F(j,2)),
    F(j,3) = j;
  end

end
```

Chapter 3:

Cluster Analysis (CA)

3.1 Background

Two common assumptions made in modeling are: 1) the members of a data class/group are well-behaved (consistent) and are similar to one another; and 2) the definition of the classes is reflected in the behavior of the measurements and the data structure. In complex dynamic systems such as bioprocesses, where detailed understanding is not often available, it is important to test these assumptions. For example, the discovery of the *Crabtree effect* demonstrates how the first assumption is violated. In this particular case, the conditions that are linked to the 'good' class, high cell density, happen to lie between the conditions that produce a 'bad' class, low cell density. For high cell density, it is important that the rate at which glucose is fed to the yeast cells be carefully monitored. It is obvious that too low of a feedrate can decrease cell concentration since the cells are starving for lack of raw materials. It was not initially apparent, however, that feeding too much glucose also lowers cell density until the Crabtree effect was discovered. The excess glucose alters the yeast's metabolism to produce ethanol, which inhibits cell growth and the cell yield on glucose (Bailey and Ollis, 1986). Thus, 2 different conditions can produce the same result, low cell density. Hence, trying to create a model to differentiate between the high and low classes may be problematic since the range of glucose feedrates that can create a low class run encompasses the range of values that also can produce good class lot. This is a case where members of one class, the low, are not consistent since low glucose feedrates can adversely lower cell concentration just as easily as high glucose levels. To model the low class effectively, it is important to recognize that it is not homogeneous and that subclasses exist.

Violation of the second assumption is closely related to an issue influenced by mean hypothesis testing from the standpoint that data itself may be inadequate to reflect differences among the classes as they are defined. As a hypothetical example, defining the basis for a class to be productivity may cause the measurements to segregate into well-defined groups whereas if the yield is used, the class differences may not be as apparent. In either case, the data is still the same but its usefulness in discriminating differences in yield is lower than its ability to detect changes in productivity.

Cluster analysis (CA) is a useful tool to explore the data structure and can be used to identify if either of the 2 previously mentioned assumptions are violated. It works by grouping together data that share similar structures- values, shapes, profiles, etc. Thus, the data can be grouped according to how similar they are to one another rather than on the basis of a predefined class. The emphasis in this thesis will be to assess class homogeneity, specifically addressing the issue of whether there are subclasses in the high and low classes. Knowledge of the data's homogeneity can lead to a more systematic approach to developing training sets for models as opposed to the current approach of random selection of samples. It is important to recognize that the key assumption underlying random selection is that the data is homogeneous. In such cases, picking lots at random for the training set is valid because each lot behaves just like any other lot so random selection is able to provide a representative sampling of the data. As will be seen later in the chapter, the data is not often homogeneous and the training set members should be selected carefully if the model is to reflect the variation in the data.

Unfortunately, one of the drawbacks to CA is that it historically has been used on samples that are discrete in nature rather than the continuous data found in historical process data. While Guthke, R, and Roβmann, R. 1991 did analyze a fermentation process using CA, the analysis was limited to discrete variables that characterized various aspects of the process, such as yield. Attempts to extend CA to time series have been done (Shaw and King, 1992; Simutis, et. al, 1995), but these efforts focused on data where only a single time series was involved. Most historical process data

consist of multiple time series, so these approaches cannot be applied since information about interactions across different time series is lost. As mentioned previously, CA deals normally with discrete data where each sample can be represented by a vector of variables which carries all the information about the sample. When analyzing historical data bases, the sample is now a production run consisting of multiple time profiles of several variables. The issue then becomes one of how does a method designed for vectors be adapted when the data are matrices.

One possible solution investigated in this chapter is to use principal components analysis (PCA) to compress the matrix into a single vector. PCA is a multivariate statistical technique that is often used to perform data compression. Here it is used to reduce the number of variables to one. If **D** is taken to represent the data of one lot, a rxc matrix where r is the total number of time points in a run and c the number of variables, in the approach outlined in this chapter **D** will be replaced by a rx1 column vector of the first principal component. The rationale is that in PCA the first principal component is constructed so that it accounts for the largest amount of the total variation in the data. Each row entry of this first principal component vector represents the variation in the data at that point in time as captured by PCA. The original matrix of multiple time series is now replaced by a single time series of the first principal component. In turn, this time series is viewed as a pattern vector for that particular lot. Since the information of each lot can now be replaced by a vector, it is possible to use conventional cluster analysis techniques. In this thesis, agglomerative clustering is used. Pattern vectors (1 per lot) with similar profiles are then clustered into different groups. It is important to realize that the groups determined by cluster analysis do not take into account the class assignments of the individual lots; CA simply assigns lots that behave similar to each other into the same cluster. Hence, it is possible to have a high lot in the same group as a low lot. This result means that these 2 lots, despite being from different classes, have a stronger similarity to each other than to other lots of the same class. Thus, by analyzing the which lots went into which resulting clusters and noting what their original class assignment was, in this case high and low, it is possible to analyze the homogeneity of the data. If the data is truly homogeneous, the makeup of the groups determined by cluster analysis should resemble the class

memberships; all high lots should cluster into one group which is separate from one containing all the low lots. If not, then it is important to account for the data heterogeneity when generating the training sets for the model.

3.2 Objective

Assess the homogeneity of class membership - identify lots with similar characteristics as well as highlight those lots that display atypical behavior. In this case, atypical is not just limited to outliers but also includes lots that appear similar to members of other classes. Once the homogeneity is understood, selection of lots for the training sets of the model can begin.

3.3 Theory

Since the algorithm used in this analysis is a hybrid of both cluster analysis and principal components analysis, a brief review of both of these general methods is presented followed by the discussion of the algorithm itself.

3.3.1 Cluster Analysis - General

The main use of cluster analysis is to identify a smaller number of groups (clusters) such that elements residing in a particular group are, in some sense, more similar to each other than to elements belonging to other groups (Dillion and Goldstein, 1984), see Figure 3.1. Several different methods for cluster analysis exist in statistics (Kaufman and Rousseeuw, 1990; Massart and Kaufman, 1983). The approaches can be divided into two general categories: hierarchical and nonhierarchial. In the former, the clusters are generated so that the smaller clusters are contained in the larger ones. In the latter, the algorithms search for a set number of clusters, k, such that the members of the same cluster are close to each other and different clusters are well-separated. Since these groupings are generated simultaneously, the resulting classification is nonhierarchial.

Although the nonhierarchial approach appears desirable, it does possess some disadvantages which can limit its utility. In general these methods are mathematically complex and iterative in nature. By iterative, the cluster memberships are not constant

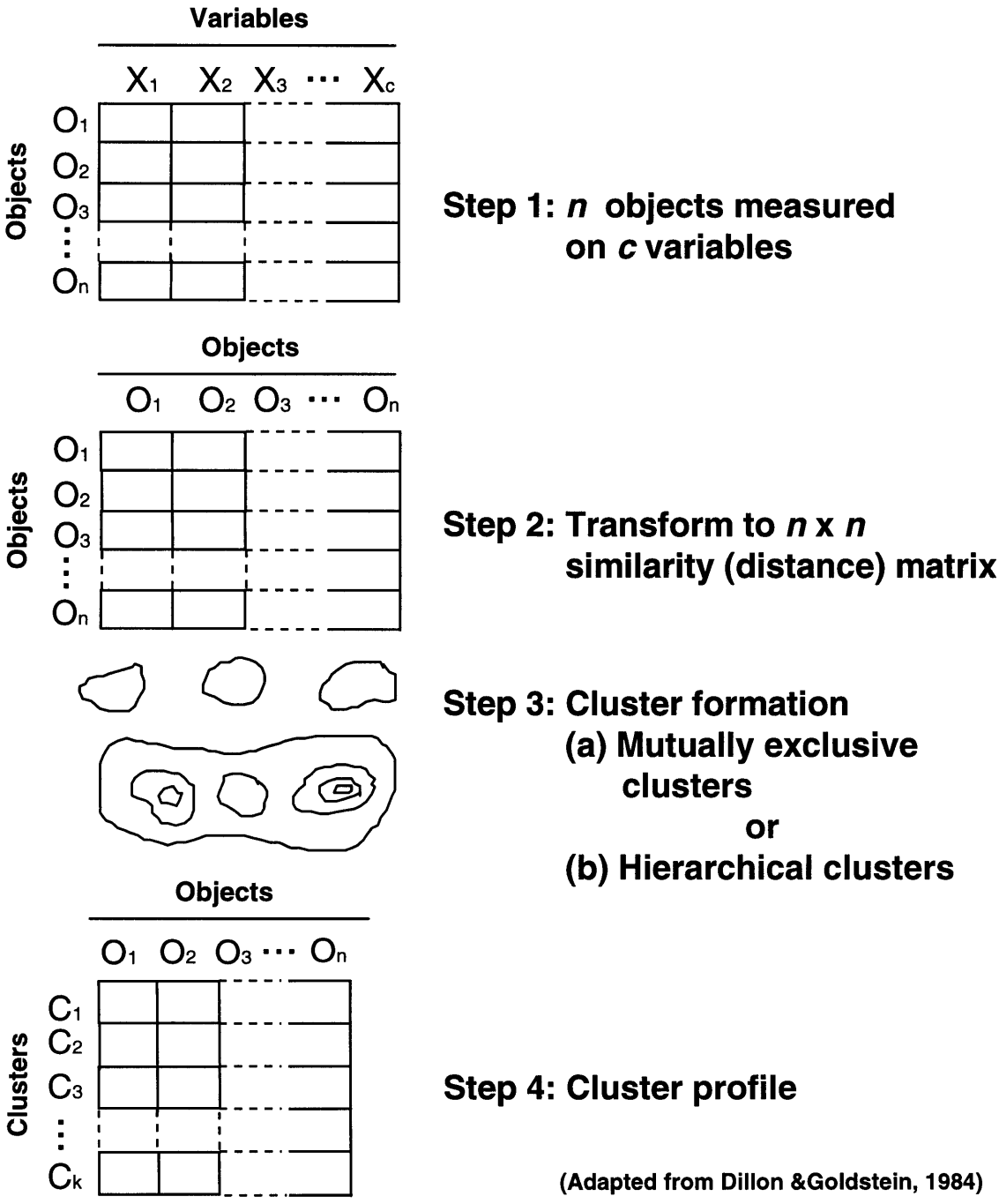


Figure 3.1 An overview of cluster analysis.

and varies with starting conditions. For example, since k clusters are assumed to exist, the algorithms select k starting points, 'seeds', to begin the clustering. Depending on the which lots are used to seed the clusters, the resulting clusters can have different membership. To obtain consistent results, the memberships from several starting points need to be examined and compared; a search has to be performed to determine which lots have remained together over all the different starting conditions. Lots that tend to associate with certain lots regardless of starting conditions are likely to be very similar to each other and this would be consistent cluster. This search can be a time-consuming task. Another disadvantage is that some clustering techniques impose a shape on the clusters. For example, the algorithm might assume the clusters to be circular in shape when in reality they may be closer to elliptical.

Hierarchical methods, on the other hand, are relatively simple in implementation and interpretation. They perform a series of successive combinations or divisions of the samples. A characteristic of these algorithms is that once a sample is allocated to a cluster the association is irreversible. The sample cannot be considered for membership in other clusters nor can it be removed from the cluster it was assigned to. Thus, there is no need to repeat the analysis several times as opposed to nonhierachial methods. Two types of hierarchical methods exist: agglomerative and divisive. In the former, the samples undergo a series of fusions until a preset number of clusters are generated. In the latter, the data is partitioned into finer and finer subdivisions.

The agglomerative approach using the nearest-neighbor method is implemented in this thesis. This method utilizes a minimum-distance rule that first identifies the two objects having the shortest distance to one another and then combines them to form the first cluster. For this thesis, the Euclidean distance is used as the measure of nearness, equation 1. At the next step, one of two actions are possible: a third object will join the previous two or the two closest unclustered samples are joined to form another cluster. The decision depends on whether the distance from one of the unclustered object to the first cluster is shorter than the distance between the two closest unclustered objects, see Figure 3.2. The process continues until a predetermined number of clusters, k, is reached.

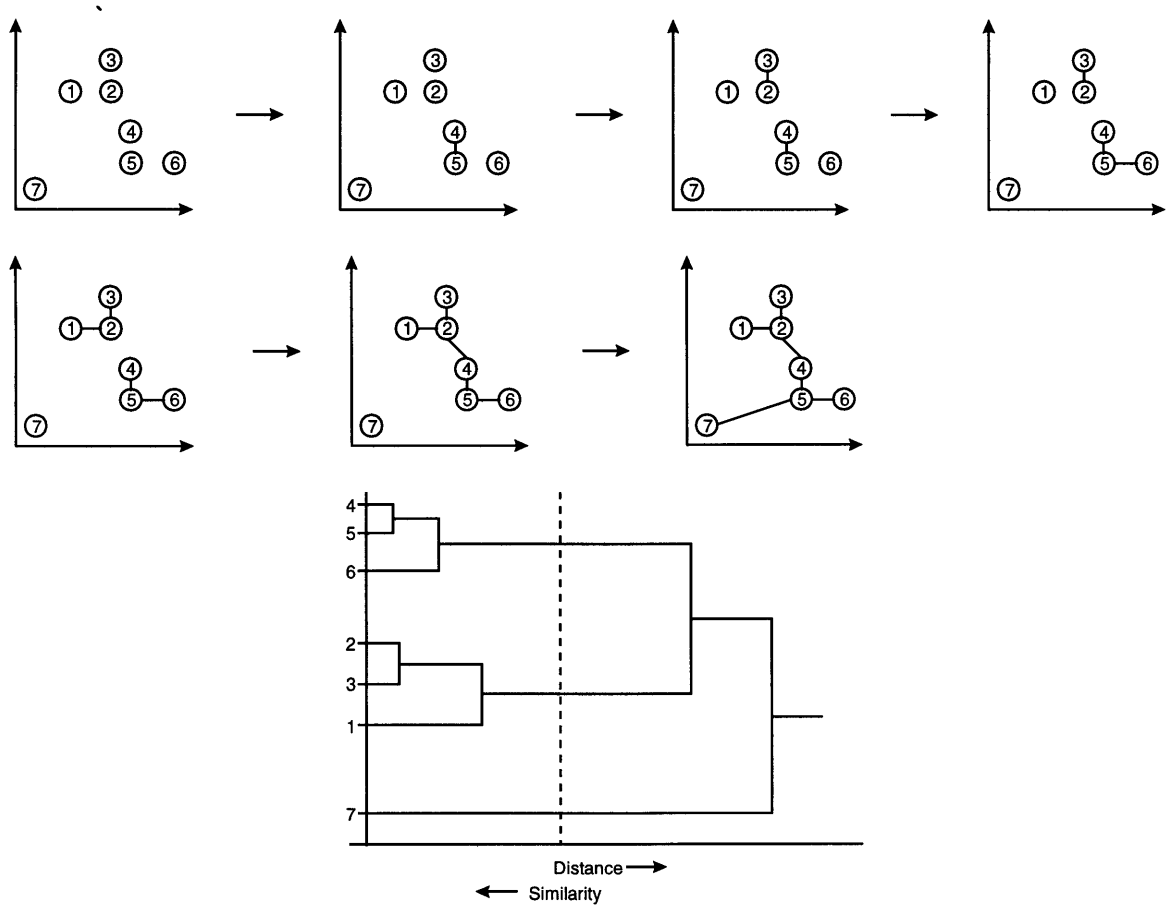


Figure 3.2 Agglomerative clustering involves grouping similar objects into the same group. (Adapted from Sharaf, Illman, Kowalski, 1986)

$$d_{ij} = \left(\sum_{k=1}^{c} (x_{ik} - x_{jk})^2 \right)^{1/2} \quad (1)$$

where:

d_{ij}: distance between samples i and j

x_{ik}: kth variable of sample i

To illustrate the nearest-neighbor clustering method used in this thesis, an example from Dillon and Goldstein, 1984 is presented. Consider a matrix of inter-sample distances, $D^{(1)}$, where the element in the ith row and jth column denotes the distance, d_{ij}, between the samples i and j as calculated by the Euclidean distance. Each sample is denoted by a letter.

$$D^{(1)} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} 0.0 & 1.0 & 5.0 & 6.0 & 8.0 \\ 1.0 & 0.0 & 3.0 & 8.0 & 7.0 \\ 5.0 & 3.0 & 0.0 & 4.0 & 6.0 \\ 6.0 & 8.0 & 4.0 & 0.0 & 2.0 \\ 8.0 & 7.0 & 6.0 & 2.0 & 0.0 \end{pmatrix} \end{array}$$

The first step is to identify the samples that are closest to one another to form the first cluster. In this case, A and B are joined since d_{AB} is the smallest element in $D^{(1)}$. After this cluster is formed, the next step is to compute the distance of the remaining samples to this cluster and to each other. It is important to note that in the nearest neighbor method the element of the cluster closest to the sample in question is used to calculate the group-individual distance.

$$d_{(AB)C} = \min\{d_{AC}, d_{BC}\} = d_{BC} = 3.0$$

$$d_{(AB)D} = \min\{d_{AD}, d_{BD}\} = d_{AD} = 6.0$$

$$d_{(AB)E} = \min\{d_{AE}, d_{BE}\} = d_{BE} = 7.0$$

A new matrix of distances, $D^{(2)}$, is constructed with the interindividual and group-individual distances:

$$D^{(2)} = \begin{array}{c} \\ AB \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{cccc} AB & C & D & E \end{array} \\ \begin{pmatrix} 0.0 & 3.0 & 6.0 & 7.0 \\ 3.0 & 0.0 & 4.0 & 6.0 \\ 6.0 & 4.0 & 0.0 & 2.0 \\ 7.0 & 6.0 & 2.0 & 0.0 \end{pmatrix} \end{array}$$

The smallest entry in this matrix is $d_{DE} = 2.0$. Since this is smaller than any of the group-individual distances, the decision is to join D and E to form a second cluster rather than joining any other individual to the AB cluster.

Again, with this new cluster, a new distance matrix, $D^{(3)}$, is generated from the following distances:

$$d_{(AB)C} = 3.0$$

$$d_{(AB)(DE)} = \min\{d_{AD}, d_{AE}, d_{BD}, d_{BE}\} = d_{AD} = 6.0$$

$$d_{(DE)C} = \min\{d_{CD}, d_{CE}\} = d_{CD} = 4.0$$

$$D^{(3)} = \begin{array}{c} \\ AB \\ C \\ DE \end{array} \begin{array}{c} \begin{array}{ccc} AB & C & DE \end{array} \\ \begin{pmatrix} 0.0 & 3.0 & 6.0 \\ 3.0 & 0.0 & 4.0 \\ 6.0 & 4.0 & 0.0 \end{pmatrix} \end{array}$$

Since the smallest distance is now $d_{(AB)C}$, the sample C is fused with cluster AB. For agglomerative clustering, this procedure is continued until a set number of clusters is reached or in the absence of such a limit all the clusters are lumped into 1 giant cluster containing all the samples. Unfortunately, in CA, there are no hard rules for determining the number of clusters *a priori*.

As seen in the example, it is relatively straightforward to use CA if the data is discrete, but as mentioned previously, the historical process data that will be analyzed is typically in the form of multiple time series, the nature of which complicates

conventional cluster analysis. This distinction can be viewed in that CA works normally on 1xc pattern vectors where c is the number of variables describing a particular sample. But since the focus of this thesis is to analyze which lots are similar, each sample is now a matrix, which is rxc in size, where r denotes the number of time points and c the number of variables. The goal is to condense the information of a matrix into a vector which can then be subject to conventional CA. The approach uses principal components analysis to compress the information of several variables into smaller set of principal components. Using only the first principal component, which captures the dominant trends in the data, a rx1 vector can be now used as the CA input. For n lots, n vectors (rx1) will be compared for similarity and clustered into g groups.

3.3.2 Principal Components Analysis (PCA) - General

A brief discussion of principal components analysis (PCA) is covered in this section; for more details on the theory and utility of PCA, the reader is referred to Jolliffe, 1986 and Morrison, 1990. If the data is redundant, PCA achieves data compaction by projecting the data to a reduced dimensional space defined by a new set of variables known as principal components (PC's). Mathematically, PCA partitions the variance in the data among the PC's, so that the first *few* PC's retain most of the variation present in *all* of the original variables (Jolliffe, 1986). The higher numbered PC's are typically assumed to model the noise in the system and are often ignored. Each PC is constructed to be orthogonal to its predecessors so each is an independent variable. The net result of PCA is that much of the information contained in the experimental space can be represented by a lower dimensional subspace if redundancy is present.

Geometrically, PC's can be viewed as a set of mutually orthogonal axes which span the variable space. The first PC is the axis that attempts to capture much variance in the data by passing through the greatest concentration of data points (Malinowski, 1987). The second PC is an axis orthogonal to the first and attempts to model as much as possible the variance not accounted for by the first PC. Each succeeding PC is orthogonal to its predecessors and tries to account for what is left of the remaining variability, see Figure 3.3.

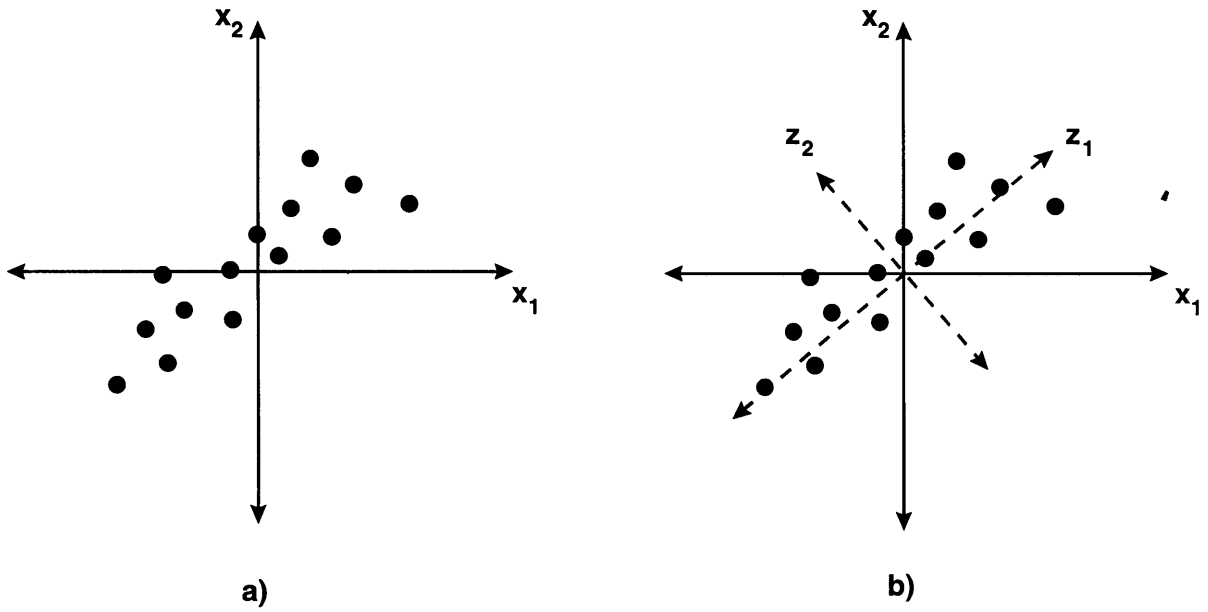


Figure 3.3 a) represents the original data (x_1, x_2); b) is the same data but the dashed lines represent the PC axes that attempt to pass the greatest concentration of data points. Here, the number of variables, c, is 2, but d (the true dimension) can be taken as 1.

As seen in the figure, the data compaction by PCA is quite evident. Although the data is 2-dimensional, most of it lies on the $x_1=x_2$ line. Thus, PCA attempts to project this data onto the 1-dimensional Z_1 line which is very close to the $x_1=x_2$ line.

PC's can be generated in a variety of methods (Jolliffe, 1986), but the method used in this chapter is based on singular value decomposition (SVD). Given a data matrix, **D**, that has dimensions r x c, corresponding to r observations of c variables ($r \geq c$), SVD can produce the following decomposition (Jolliffe, 1986):

$$\mathbf{D} \cong \mathbf{U}^*\mathbf{S}^*\mathbf{V}^T \tag{2}$$

where:

1) **U** and **V** are (r x d), (c x d) matrices respectively, each of which has orthonormal columns so that $\mathbf{U}^*\mathbf{U}^T = \mathbf{I}_d$, $\mathbf{V}^T*\mathbf{V} = \mathbf{I}_d$.
2) **S** is a (d x d) diagonal matrix whose nonnegative elements, called singular values, are ordered in **decreasing order,** $\mathbf{S_{i+1} < S_i}$.
3) d is the true rank of **D**.

If there is **no noise** in the system and redundancy exists, equation 2 changes to an equality. Equation 3 is the relationship between singular values and PC's, designated as $\mathbf{Z_i}$'s:

$$\mathbf{Z} = \mathbf{U}^*\mathbf{S} \tag{3}$$

where :

Z is a (r x d) matrix whose columns correspond to the PC's; i.e., the ith column of **Z** is the ith PC and will be designated $\mathbf{Z_i}$.

From equations 2 and 3, each element of **D** can now be written as :

$$dij = \sum_{n=0}^{d} \mathbf{Z}_{in} * \mathbf{V}_{nj}^T + e_{ij} \cong \sum_{n=0}^{d} \mathbf{Z}_{in} * \mathbf{V}_{nj}^T \tag{4}$$

In matrix form:

$$\mathbf{D} \cong \mathbf{Z} * \mathbf{V}^T \qquad (5)$$

The approximation sign shown in the second half of equation 4 and in equation 5 is due to the fact that not all the PC's are being used. The error term (e_{ij}) is assumed to be zero for equation 5. As mentioned earlier, much of the variance in the data is captured by the first few $\mathbf{Z}_i$'s. The contribution by the each successive PC to the prediction of **D** decreases with each higher index; so, for example, Z_4 contains less data than Z_3 . $\mathbf{Z}_d$ signifies the last PC that contributes significantly to the prediction of **D** to within measurement error. Should all the generated singular values be used (S_1, ..., S_c), the resulting PC's would regenerate the data perfectly including the noise. Since the higher $\mathbf{Z}_i$'s capture the least amount of variance, they are assumed to reflect system noise; hence, their elimination reduces the noise level and achieves data compaction. The final result is a small set of transformed variables that capture most if not all the variability in the data.

3.3.3 Approach -- PC1 Time Series Clustering

PC1 Time Series Clustering is the technique described in this thesis that allows cluster analysis to be performed on multiple time series. PCA is performed on all the lots and a matrix of PC's is obtained. Of these, only the first component carrying the dominant variation of the data is used. Each lot is now represented by a single column vector with time imbedded in the row positions. These column vectors are equivalent to the input vectors normally used in the agglomerative clustering algorithm and it is now possible to analyze clusters of lots. Figure 3.4 is a schematic detailing the above. The distance measure that is used is again the Euclidean distance between the vectors:

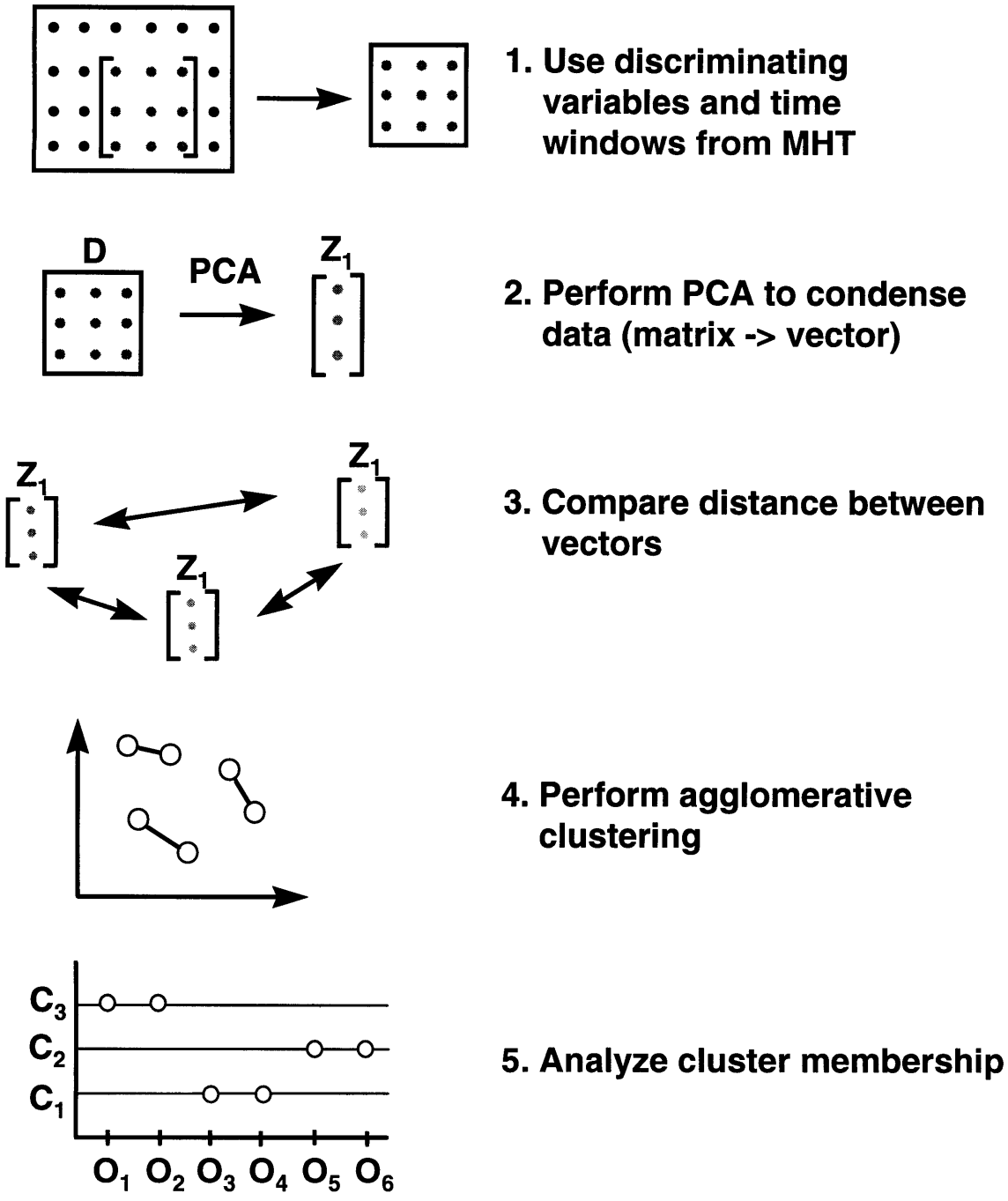


Figure 3.4 Recipe for using PC1 Time Series Clustering.

3.4 Results

The same data used in Chapter 2 is also used to test the PC1 Time Series Clustering. As mentioned previously, there are no universally accepted rules for determining the number of clusters *a priori*, so the number of clusters must be set by the user. To explore the homogeneity of the data, it is important to vary this number and observe the changes in the cluster composition. By observing how the cluster memberships evolve, it is possible to identify atypical lots as well as possible subclasses. It is important to note that the cluster memberships identified by CA are different from class memberships. In the latter, the lots are assigned to a class by a measure determined by the industrial source, such as yield, product concentration, etc. In the former, the memberships indicate which lots behave similar to one another regardless of its class designation. Hence, it is possible in CA to have both high and low lots in the same group if their profiles are similar.

To understand the results presented here, several charts of the type shown in Figure 3.5 will be used. The x-axis displays the relative lot number with the convention used that the high class lots (marked as circles) start first followed by the low class ones (marked as x's). So if there are 45 lots and the first 22 of them are high class lots, then positions 1 to 22 on the x-axis reflect these lots with 23 to 45 representing the low. The y-axis contains the cluster numbers as determined by CA. Each integer y-value represents a cluster. So at y = 1, for any point that falls on this line the corresponding lot associated with that point belongs to that cluster, in this case group 1. In varying the number of clusters, special attention must be paid to determining which lots fall in line with other lots - similar behaving lots fall into the same group and dissimilar ones in different clusters.

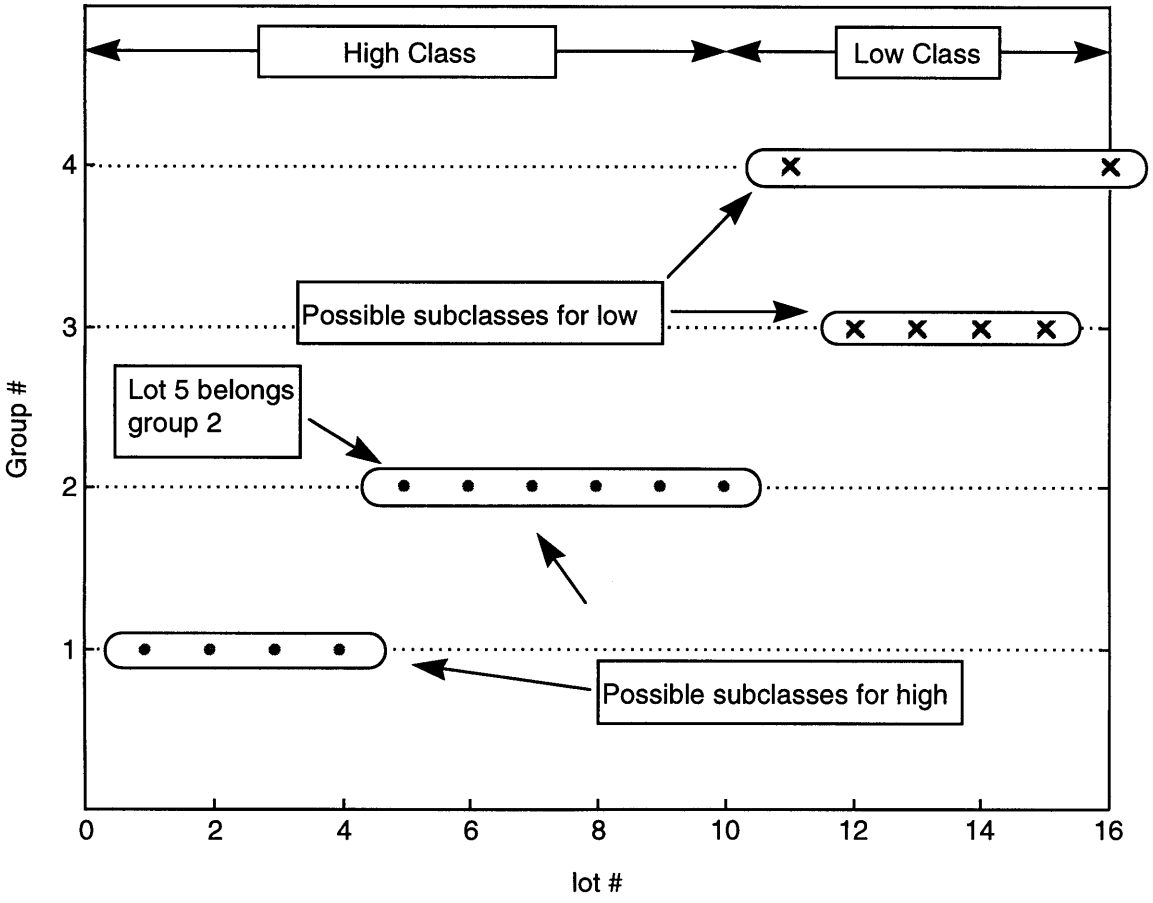


Figure 3.5 Chart for interpreting cluster analysis results.

If high and low classes are markedly different, then the behavior as seen in Figure 3.5 should be expected - each cluster composed of either pure high class or pure low class lots. Some outliers may exist but ideally they should be different from their parent as well as the "rival" class, denoting their outlying status. If more than one of these "outliers" (same parent class but not grouped with their siblings) form another cluster, this may indicate the presence of subclasses. Unfortunately, reality tends to produce clusters with overlapping memberships from both classes. While this may appear disconcerting at first, it is important to recognize that a low quality run is not a desired goal but the realization that something went wrong or was wrong with the process. Unless it is obvious from the beginning, a run will usually begin its life as good and because of some disturbance deviate to become substandard. Hence, the similarity between profiles of high and low classes might be considerable.

To reduce the computational time and to avoid information overload, MHT is used to reduce the data to contain only the discriminating variables and time windows. As will be seen in Chapter 4, having irrelevant data can adversely affect algorithm performance and lead to inaccurate conclusions.

3.4.1 Case Study 1

Type:	industrial fermentation
Mode:	fed-batch
Run length:	82 time points
Number of variables:	17 (see note in Chapter 2)
Number of classes:	2
Lots/class:	22 for high and 23 for low

The data analyzed here is the same as the one used in case study 1 of Chapter 2. From the MHT results, it is known that the individual variables[1] 4, 5, 9, and 11 are discriminating in the time window of 40-82; this information is used as the basis for the

[1]The 4 variable combination 7, 8, 11, and 12 was not selected because the clusters displayed by this combination tended to generate memberships which overlapped between the classes.

cluster analysis. Figure 3.6 displays the group memberships assuming only 2 clusters exist. As seen in the figure, it is observed that the group separation predicted by the PC1 Time Series Clustering is strongly reflective of the class membership. Group 1 is dominated mainly by high class lots while group 2 consists of mostly low class lots. There are also indications of subclasses as shown by the presence of some low lots in the high class. But this result cannot be verified, since only 2 groups are allowed, without increasing the number of clusters.

Increasing the number of groups to 3, see Figure 3.7, the heterogeneity of the low class becomes apparent. The low class consists now of 2 major groups, 2 and 3, with some overlap in group 1. In contrast, the high class is unchanged with lot 4 still being the only outlier in the class. It is also interesting to not that lots 37, 39, and 43 are associated with group 1, a predominantly high class cluster.

When 4 groups are considered, see Figure 3.8, several observations are made. First, the algorithm fractionates the high class into 2 groups, 1 and 2. Second, lot 4 is still associated with low lots as it has in the previous 2 situations. This finding is important to note because this suggests that lot 4 is no ordinary outlier but is a high lot that has characteristics in common with some low lots. As such, lot 4 is not a lot that should be entered into the training set of the high class model. Similarly, lot 39 is also problematic since it falls in group 2, which is dominated by the high class lots. This is also another lot that should be avoided for training purposes. The makeup of group 1 illustrates how lots from different classes can have similar profiles. In this group, lots 37 and 43 are present with 4 high class members. These particular lots may also produce difficulties in class discrimination if used in the training of the low class model.

Figure 3.9 shows the situation for 5 groups. Aside from the further separation of some low lots between groups 4 and 5, the results are essentially the same as in the 4 group case and further analysis was terminated.

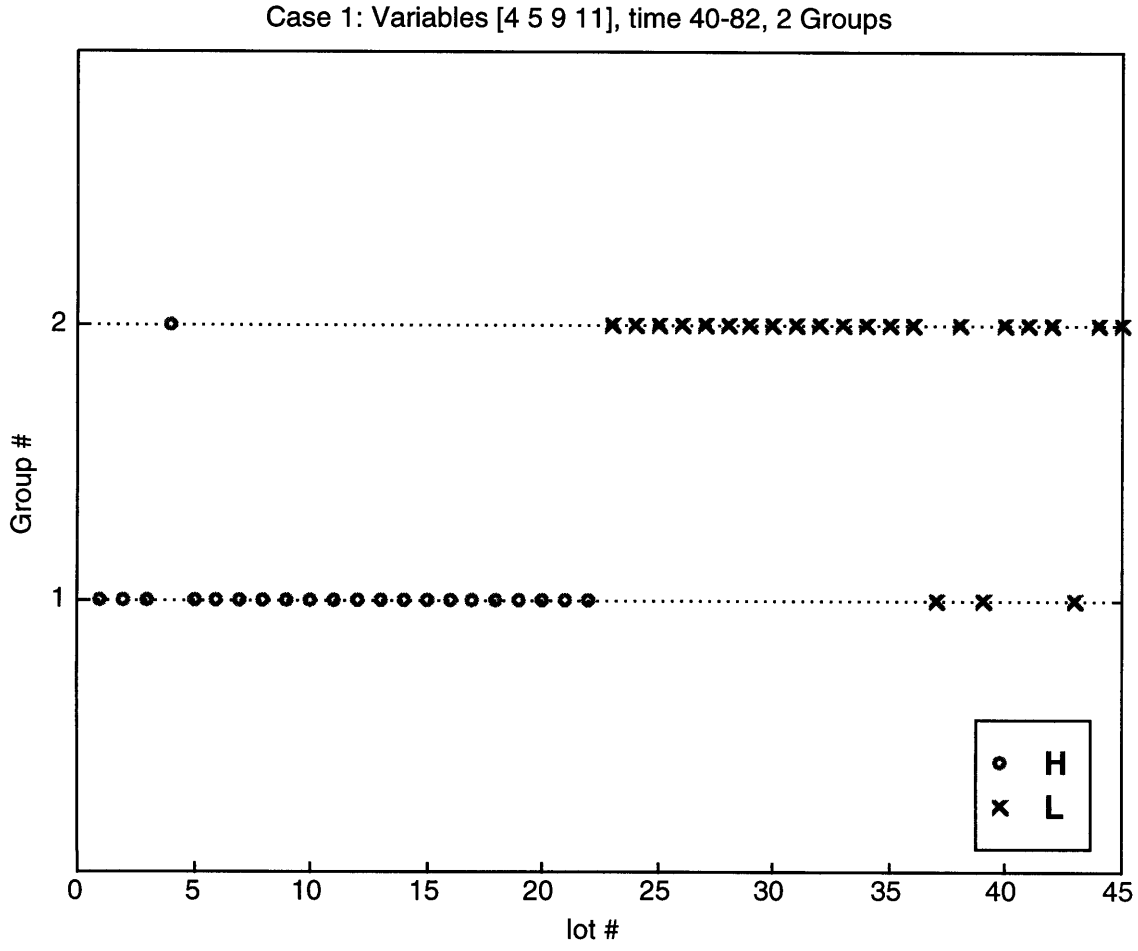


Figure 3.6 Clustering grouping with only 2 groups; high and low separate into their own groups.

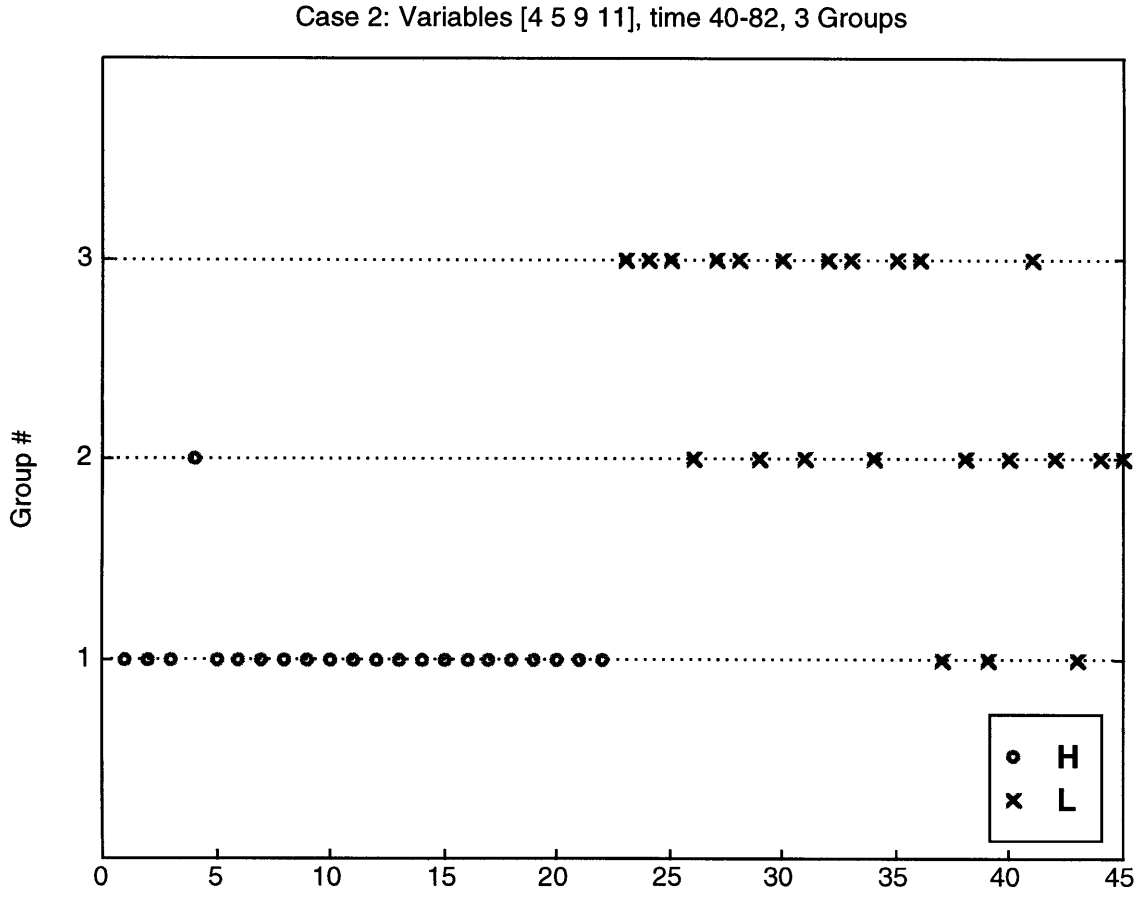


Figure 3.7 Clustering grouping with 3 groups.

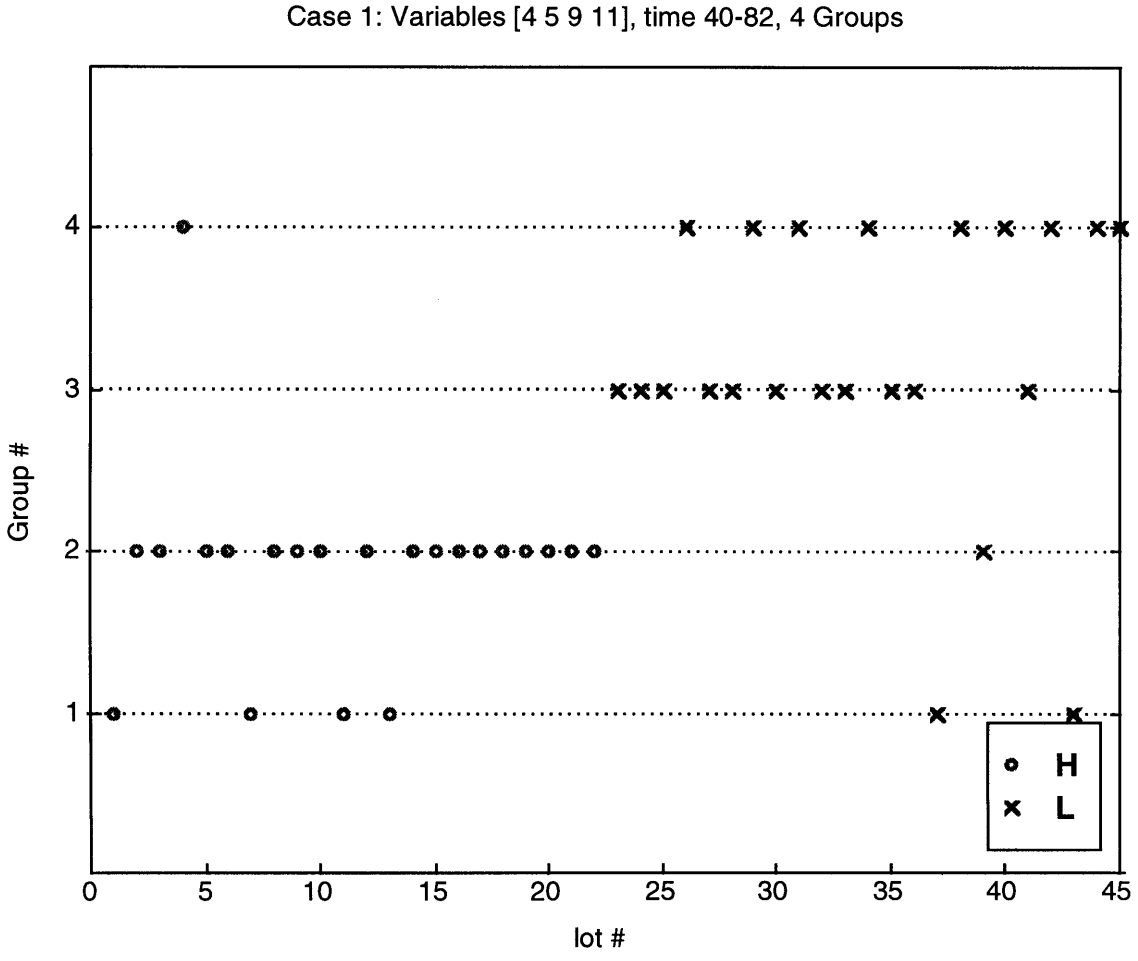


Figure 3.8 Clustering grouping with 4 groups.

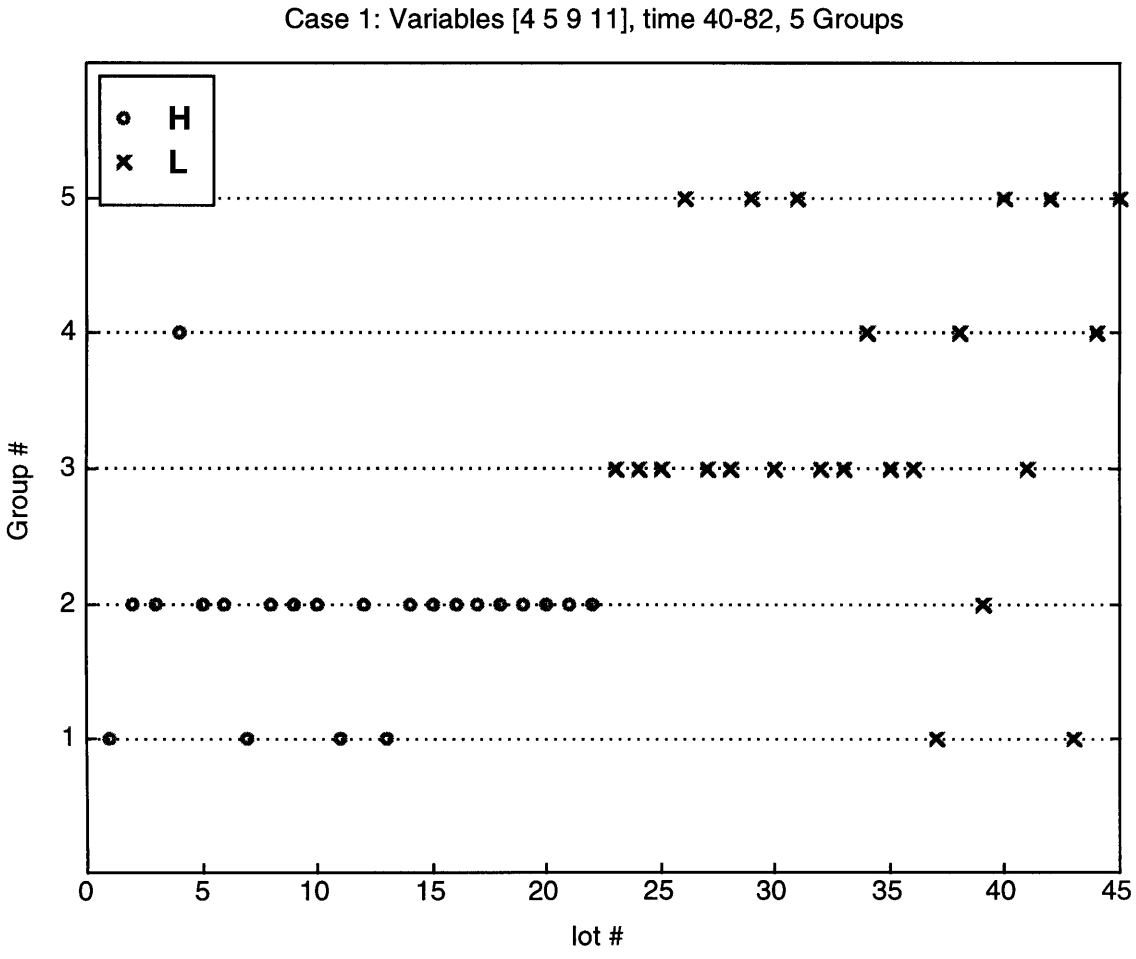


Figure 3.9 Clustering grouping with 5 groups. Good class separation.

By varying the number of clusters and observing the membership patterns of the individual lots, insight into the homogeneity of the data can be obtained. As the behavior of lots 4, 37, and 43 indicate, the problem is not simply a lot that does not resemble its fellow classmates but there are occasions where they behave similar to rival classes. These lots should not be used in the training set of models because they represent atypical (ambiguous) cases, not just simply outliers.

3.4.1.a Decision tree

The results of PC1 Time Series clustering were compared with the classification results obtained with dbminer (refer to Chapter 2 for details). It was observed that cluster analysis could provide some insight into the members selected by the tree branches. The results of the decision tree generated by dbminer using only variable 4, see Figure 3.10, show that lots 39 and 43 (both low lots) cannot be separated from the 22 high lots until further down the tree, implying that they may be similar. (This cannot be viewed from the figure but can be seen in dbminer). Lot 39 is finally separated from the high class lots at the third level of the tree, but lot 43 still remains with all of the highs. It is at the fourth level of the tree that lot 43 is finally segregated but in doing so the decision tree separates out 2 other high class lots, 4 and 13, with it. This last result is interesting to note because from Figure 3.9, both lots 4 and 13 have been linked with low lots (different groups). While the conclusion is still speculative, similar observations with other trees (not presented here) strongly suggest that CA might be useful in interpreting the fine separations made at lower levels of decision trees. This can be especially insightful when one considers that the decision tree appears to have difficulty separating those high and low class lots that CA predicts to be quite similar.

3.4.1.b Discriminating Time Windows and Variables

Although the results of MHT are used to perform cluster analysis, CA can also be used to verify the discrimination results from MHT. In Figure 3.11, using the discriminating variables 4, 5, 9, and 11 in a time window of 1 to 40, the class separation is poor as evidenced by the mixed memberships of both classes in the two

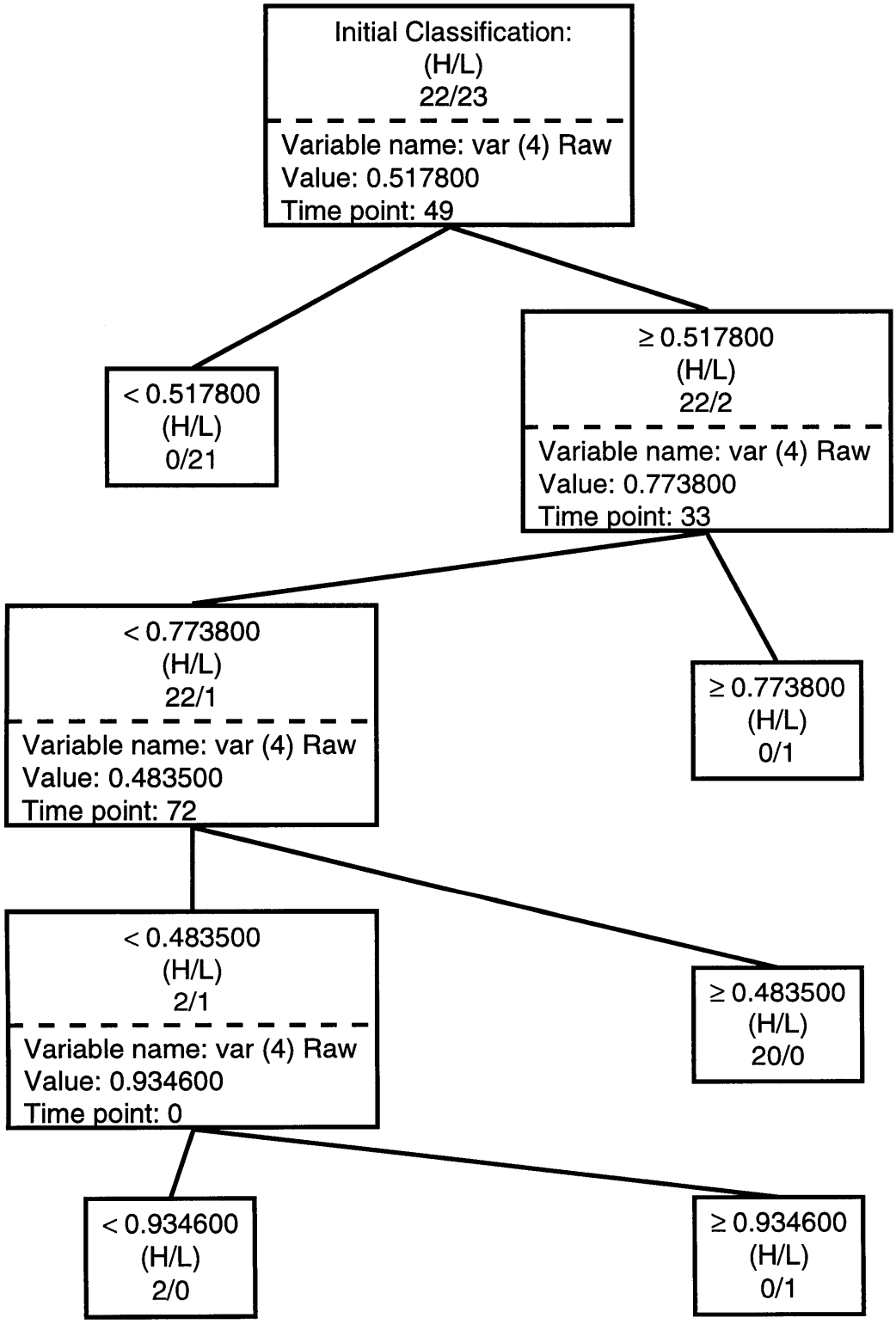


Figure 3.10 Decision Tree generated by *dbminer* using variable 4.

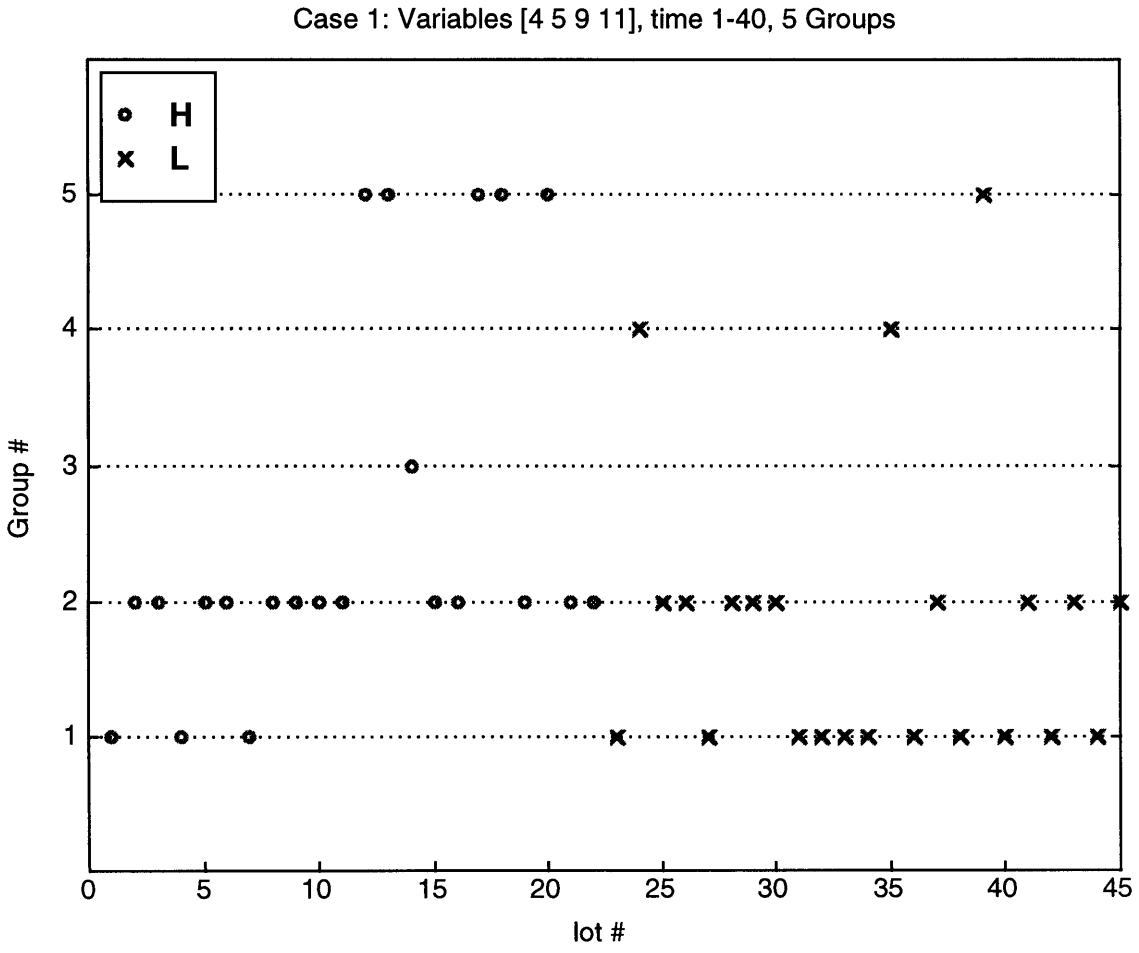


Figure 3.11 Mixed class membership indicates poor separation of high and low lots in time window 1-40.

largest groups, 1 and 2. This result is consistent with MHT's finding that 1-40 is a nondiscriminating time window. If the analysis is moved to a discriminating time window, the class separation is cleaner with group 2 being dominated by the high and low being split among groups 3 and 5 as seen in Figure 3.9. Likewise, the price of using poor discriminators can be seen in Figure 3.12 where variables 13, 14, 16, and 17 are used in a discriminating time window 40 to 82. Group 1 carries the bulk of both the high and low classes. As expected using a nondiscriminating time window of 1-40 does not alter the class separation significantly, see Figure 3.13.

3.4.2 Case Study 2

Type:	industrial fermentation
Mode:	fed-batch
Run length:	146 time points
Number of variables:	8
Number of classes:	2
Lots/class:	10 for high and 6 for low

This case illustrates the need to vary the number of clusters in exploring the homogeneity of the data. The first 10 lots are high class with 11-16 being the lows. Figure 3.14 shows the initial starting point of the analysis considering only 2 groups using the discriminating variables, 4 and 5, and time window 40-60 as suggested by MHT. Initially, it appears that the MHT's recommendation is incorrect as both the high and low classes fall into the same group. However, the correct interpretation is that lots 11 and 16 appear to behave differently enough from both highs and lows as to form their own class. If these 2 lots are deleted as shown in Figure 3.15(a), a more understandable separation occurs with group 1 dominated by the high and group 2 by the low. A similar result is obtained when the number of groups is increased to 3, see Figure 3.15(b).

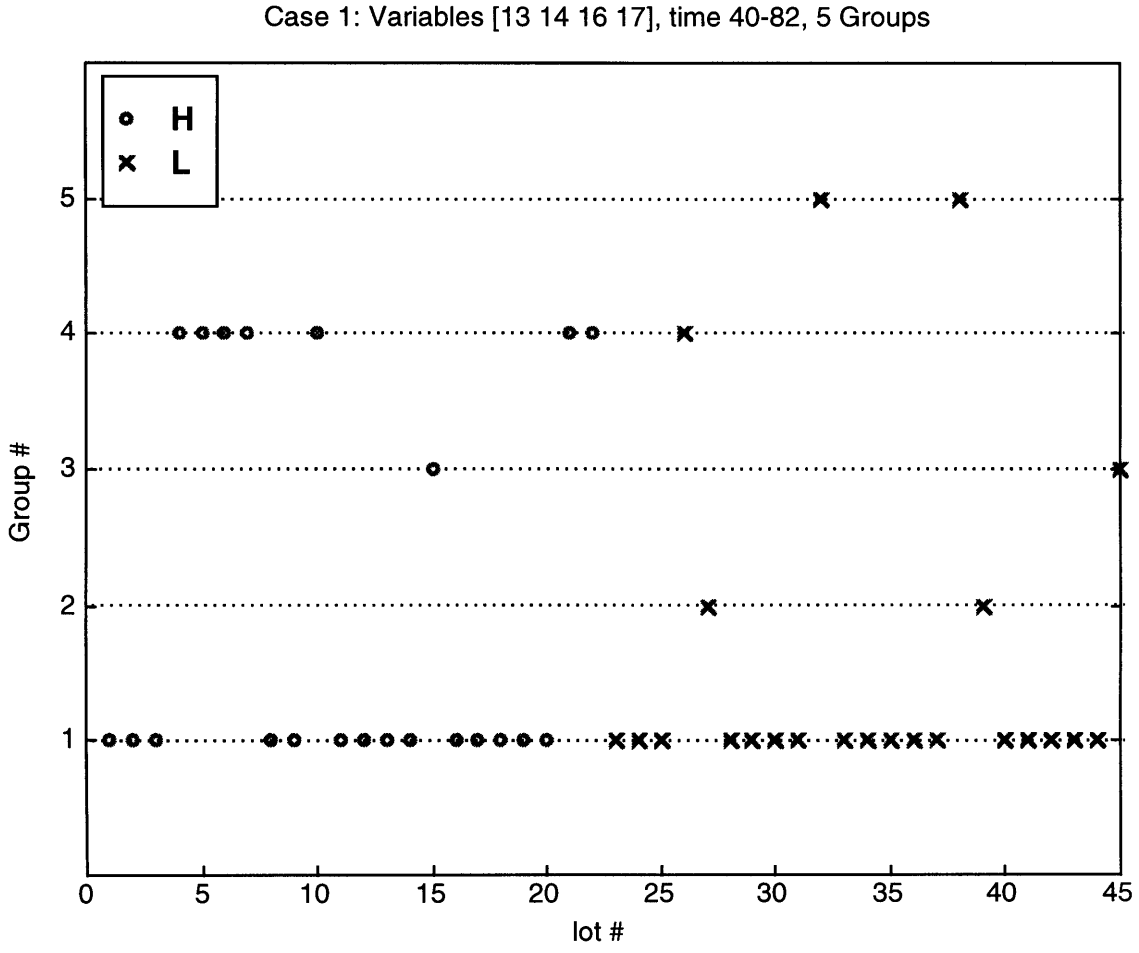


Figure 3.12 Discriminating time window w/ nondiscriminating variables produce very class-mixed groups indicating poor class separation.

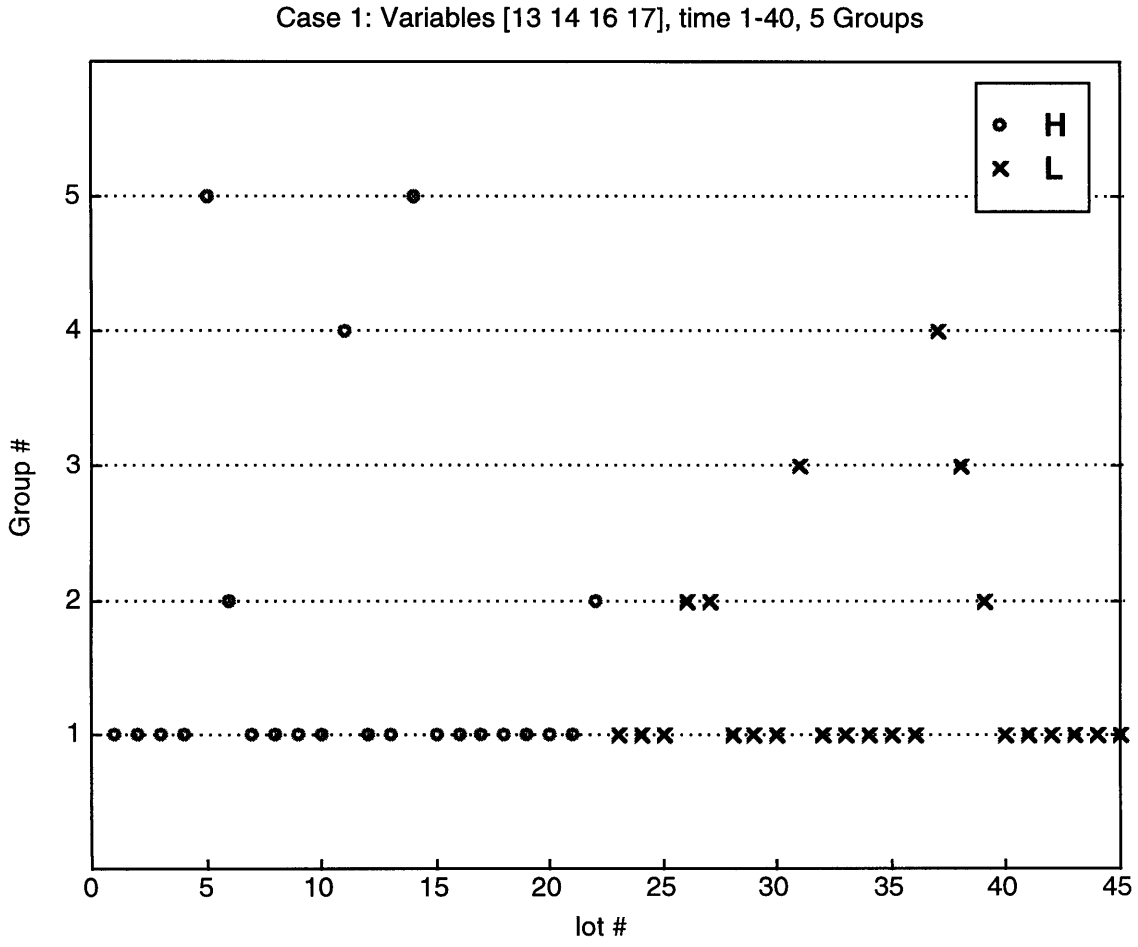


Figure 3.13 Discriminating time variables with nondiscriminating time window very class-mixed groups indicating poor class separation.

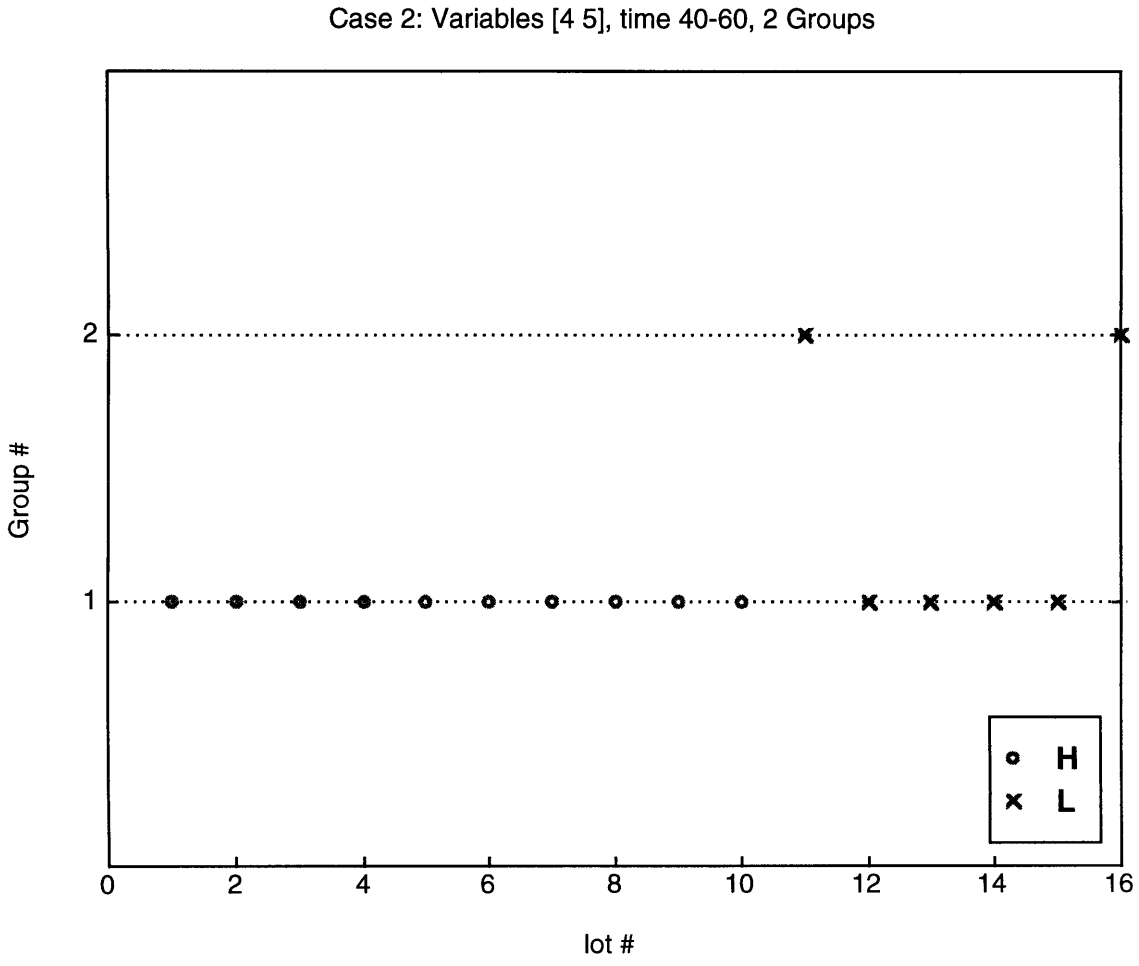
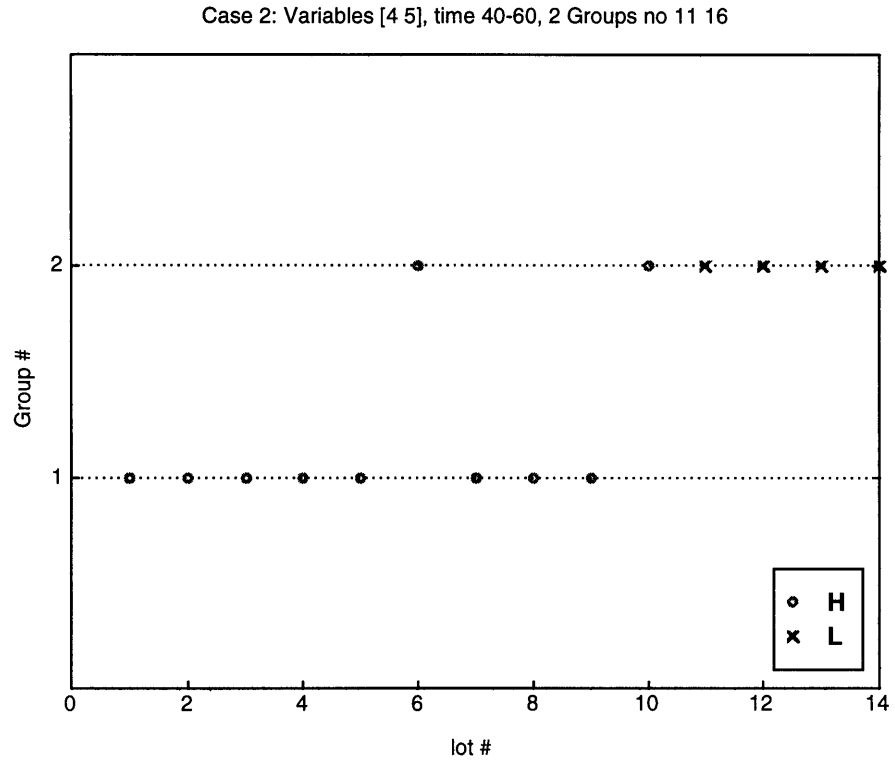


Figure 3.14 CA results with Case study 2 data and 2 groups.

(a)



(b)

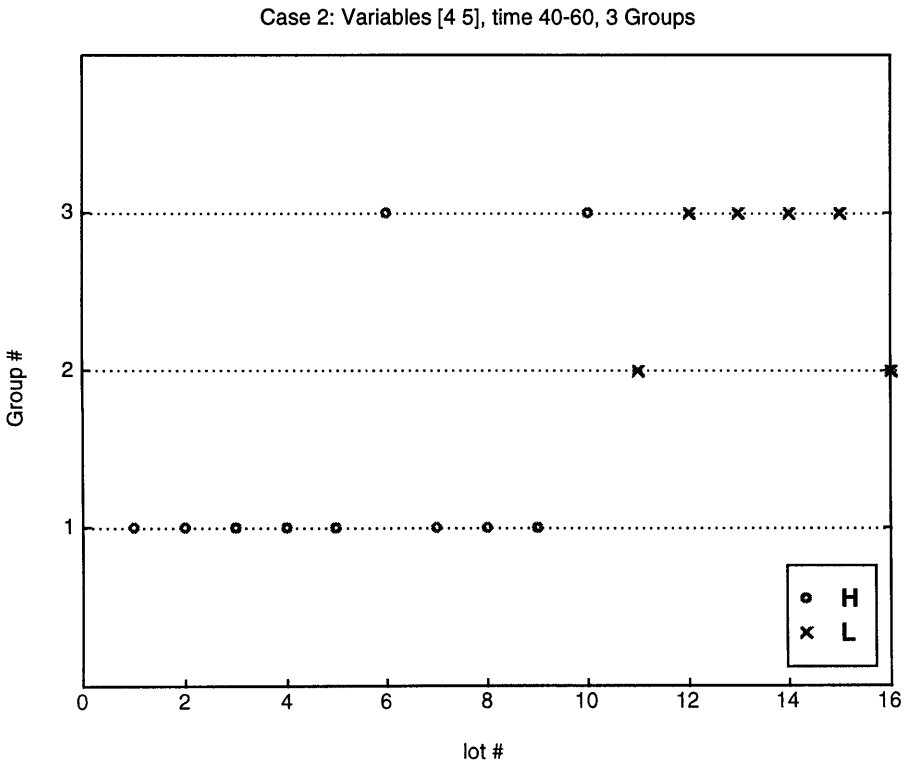


Figure 3.15 (a) Grouping if problematic lots removed; (b) Grouping if increase group number to 3.

Again lots 11 and 16 form a separate cluster apart from the highs as well as the lows. It is also interesting to note that lots 6 and 10 which first clustered with the lows when considering 2 groups (without 11 and 16) also do so when considering only 3 clusters. Going to 4 groups, see Figure 3.16, and aside from the separation of the high into 2 groups, not much changes. It appears in this case, there are 2 high lots, 6 and 10, which can be problematic in modeling the high class as well as a low subclass consisting of lots 11 and 16.

3.5 Conclusions

Cluster analysis in the form of PC1 - times series clustering is a systematic approach to understanding variability in data containing multiple time series. While this technique can also be used to indirectly identify discriminating variables and time windows, its main strength is still to explore data homogeneity. As seen in the case studies presented, the individual lots do not always follow the behavior of their classmates. As such there may be subclasses present within the existing class designation as well as class members which exhibit similarities with other classes. The former suggests that one should take members of each subclass into account if the objective is to accurately model the class as a whole. The later suggests that some lots are problematic and should not be used in the training of the model. These misfits, however, are not entirely useless as they share characteristics of both classes. Understanding why a low class lot shares so many characteristics with the high class but still failed or a high class lot that appears to be substandard but is still acceptable may provide insight into the dynamics that determine the success or failure of a run.

In either situation, PC1 - time series clustering provides the user with a rational basis for selecting members of a training set for modeling.

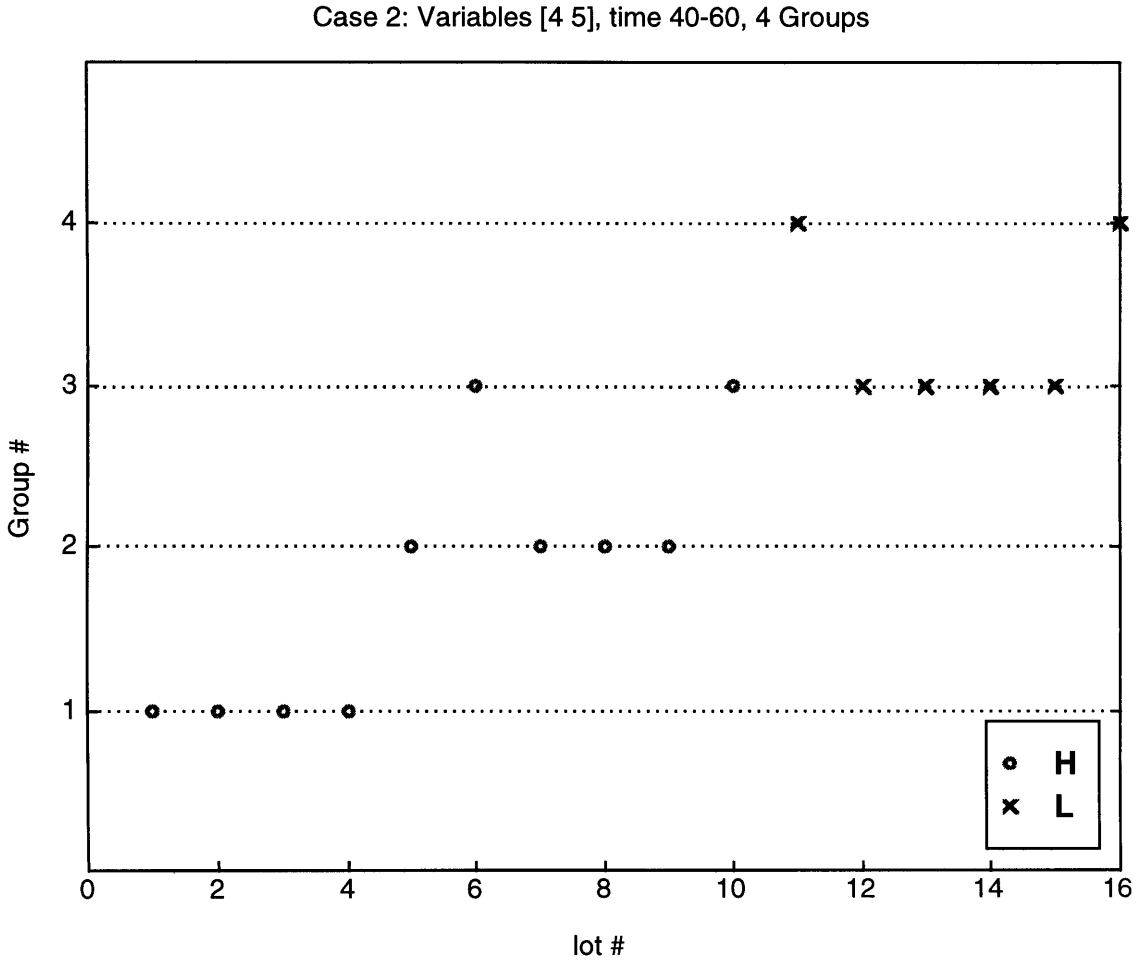


Figure 3.16 Clusters for 4 groups.

3.6 References

Bailey, J.E. and Ollis, D.F. (1986)

Biochemical Engineering Fundamentals, 2ed., McGraw-Hill, Inc.

Dillon, W. R. and Goldstein, M. (1984)

Multivariate Analysis: Methods and Applications, John Wiley & Sons, Inc

Guthke, R, and Roβmann, R. (1991)

"Fermentation analysis by clustering," *Bioprocess Engineering*, 6, p157-161

Jolliffe, I (1986)

Principal Components Analysis, Springer-Verlag

Kaufman, L and Rousseeuw, P. J. (1990)

Finding Groups in Data: An Introduction to Cluster Analysis

John Wiley & Sons, Inc.

Malinowski, E. and Howery, D.G. (1980)

Factor Analysis in Chemistry, Wiley, New York

Massart, D.L. and Kaufman, L (1983)

The Interpretation of Analytical Chemical Data by the Use of Cluster Analysis,

John Wiley & Sons, Inc.

Morrison, D.F. (1990)

Multivariate Statistical Methods, McGraw-Hill, New York

Shaw, C. T., and King, G. P. (1992)

"Using cluster analysis to classify time series," *Physica D,* 58, p 288-298.

Sharaf, M. A., Illman, D., and Kowalski, B. R. (1986)
Chemometrics, John Wiley & Sons, New York

Simutis, R., Havlik, I, Schneider,F, Dors, M and Lubbert, A (1995)
"Artificial neural networks of improved reliability for industrial process supervision," Preprints of Papers from CAB6 - Computer Applications in Biotechnology, p 59-65

3.7 Appendix

```
function c = agglom(x,nc,measure);
% AGGLOM : Basic Agglomerative Clustering
% c = agglom(x,nc,measure)
%      x      - d*n samples
%      nc     - number of clusters wanted
%      measure - distance measure used to group clusters
%      c      - calculated membership vector
% warning: AGGLOM is very computationally intensive and is not yet optimized
%         it was just implemented as a demonstration

% Copyright (c) 1995 Frank Dellaert
% All rights Reserved

[d,n] = size(x);

% initialize with singletons
c = linspace(1,n,n);
nk = n;

while nk>nc,
  %------------------------------------------------------
  % find nearest pair of distinct clusters
  %------------------------------------------------------
  dmin = feval(measure,cluster(x,c,1),cluster(x,c,2));
  imin = 1;
```

```
  jmin = 2;
  for i=1:nk-1,
    for j=i+1:nk,
      d = feval(measure,cluster(x,c,i),cluster(x,c,j));
      if d<dmin, dmin=d; imin=i; jmin=j; end
    end
  end
  %------------------------------------------------------
  % Merge cluster imin and cluster jmin
  %------------------------------------------------------
  for i=1:n,
    if c(i)==jmin, c(i)=imin; end
    if c(i)==nk, c(i)=jmin; end    % recover cluster index
  end
  %------------------------------------------------------
  % Decrement count
  %------------------------------------------------------
  nk = nk - 1
end
```

```
function d = dmean(x1,x2);
% DMEAN : distance between means of two clusters
% d = dmean(x1,x2)
%       x1, x2 - the two clusters
%       d      - the result

% Copyright (c) 1995 Frank Dellaert
% All rights Reserved

m1 = mean(x1')';
m2 = mean(x2')';
d = norm(m1-m2);
```

```
function [nr,m,v] = clusterstats(x,c,nc);
% CLUSTERSTATS(x,c) computes the statistics for each cluster
% [nr,m,v] = clusterstats(x,c[,nc])
%       x  - d*n matrix of samples
%            d - dimension of samples
%            n - number of samples
%       c  - 1*n matrix with the cluster identity for each sample x(:,i)
%            should contain numbers between 1 and nc
%       nc - the number of different clusters (max(c) if omitted)
%       nr - 1*nc matrix with the number of samples in each cluster
%       m  - d*nc matrix with the mean for each cluster
%       v  - d*nc matrix with the variance for each component
```

```
% Copyright (c) 1995 Frank Dellaert
% All rights Reserved

if (nargin<3) nc=max(c);end

% get dimensions of data
[d,n] = size(x);

% calculate sum and number of members in each cluster
nr = zeros(1,nc);
sum = zeros(d,nc);
sumsq = zeros(d,nc);
for i = 1:n,
  xi = x(:,i);
  sum(:,c(i)) = sum(:,c(i)) + xi;
  sumsq(:,c(i)) = sumsq(:,c(i)) + xi .* xi;
  nr(c(i)) = nr(c(i)) + 1;
end

% calculate mean and variance for each cluster
m = zeros(d,nc);
v = zeros(d,nc);
for j = 1:nc,
  m(:,j) = sum(:,j)/nr(j);
  v(:,j) = sumsq(:,j)/nr(j) - m(:,j) .* m(:,j);
end
```

function [n,m,v] = clustertest(nr);

```
% CLUSTERTEST : test clusterstats with really simple distribution
% [n,m,v] = clustertest(n)
%       nr - number of samples

% Copyright (c) 1995 Frank Dellaert
% All rights Reserved

data = zeros(3,nr);
data(1,:) = randn(1,nr);
data(2,:) = randn(1,nr)*2+1;
data(3,:) = randn(1,nr)*3+2;

datac = ones(1,nr);

[n,m,v] = clusterstats(data,datac);
```

PC1-Clustering algorithm

function [cZ1,Z1] = ZClusAnl(D,VL,useDExt,opt,strtmrk,endmrk,mrkpost,LotS,nc)

```
% DATE: 12/4/96, Roy Kamimura, copyright (c) by MIT
% DEFINTION:
```

```
% Performs cluster analysis using agglom on PC1 generated by
% the data.
% VL = variable list
% useDEXt = 0 for no use
% opt = 0,1 for DExt
% nc = number of clusters

if (useDExt ~= 0),
  D = DExt(D,opt,strtmrk,endmrk,mrkpost,LotS);
  LotS = endmrk - strtmrk + 1; % if using opt = 0 else undefined
end

DataMC = D - ones(row(D),1)*mean(D);
[u s v] = svd(DataMC(:,VL),0);
z = u*s;
z = z(:,1);

Z1 = PlotVar(z,LotS,1);
cZ1 = agglom(Z1,nc,'dmean');
plot(cZ1,'x')
axis([ 0 col(cZ1) 0 (nc+1)])
```

Chapter 4:

Data-Driven Modeling

4.1 Background

The complexity and uncertainty associated with observing biochemical systems coupled with their unsteady-state nature have hampered efforts to model most bioprocesses. As a result, the information required for creating a mechanistic-based model is not readily available. What is present, however, are the large volumes of process data routinely collected during the development phase and actual manufacturing. The knowledge stored in this database may be able to provide some insight into a process if sufficient attention is paid to identifying patterns in the data. The analogy is similar to that of a car owner's long-time history with their vehicle. Through frequent contact, he or she develops an intuition about the car's normal behavior. So, while the driver may not know details such as the engine's usual combustion temperature, normal gas composition of the exhaust gases, or other detailed variables monitoring the car's performance he or she may have a "feel" about their vehicle. If anything breaks this association, the driver is alerted to the possibility that something is wrong with the car. So in much the same way, the computer may not have access to the kind of detailed information needed in a mechanistic model, but this doesn't mean it cannot take advantage of the measurements that are available. By observing what profiles and trends occur for normal operation, the computer may be able to identify characteristic patterns. However, just like the driver, there are times when a "mechanic" is needed since the information required to make the diagnosis is not in the database.

One of the aims of this research is to have the computer model those associations that may exist between process measurements and process outcome.

This chapter will discuss models that have been developed to uncover this association so that it will be possible to determine as early as possible the final process behavior. The assumption is that production runs with similar outcomes should have a characteristic pattern, "fingerprint," present in the measurement matrix. There is, of course, the possibility that no such fingerprint exists. But if this is indeed the case, then this would imply that the data collected does not contain enough information to be used in process classification; and if this is the goal, more sensitive probes or other alternative measurements must be made.

4.2 Objectives

Perform early process classification via models based on historical records. Also, demonstrate the impact of integrating information provided by mean hypothesis testing (MHT) and cluster analysis (CA) in developing the classification models.

4.3 Theory

Using the results from MHT and CA, two different algorithms, historical mean time series and autoassociative neural networks will be used to model patterns in the historical database. Their performance will be evaluated by their ability to classify lots not present in their training phase.

Both algorithms use a classification scheme based on the principles of Soft Independent Modeling of Class Analogy (SIMCA), (Wold and Sjostrom, 1977; Saner and Stephanopoulos, 1992). This is a technique used in supervised statistical pattern recognition when using principal components analysis (PCA) to classify an unknown sample. SIMCA works by constructing models of each class and fitting the unknown to each of the classes. If the sample resembles the members used in the training set of the model, the resulting error in fit should be comparable to ones observed in the training phase. If this is indeed the case, the sample is assigned to the class with the best agreement provided it is rejected by the other models as well.

What makes SIMCA's classification unique, however, is the ability to recognize that a sample may have characteristics of several classes or none of them. The reasons why this is important are several. First of all, when modeling classes that are

very similar to each other, this classification scheme will show many both classifications. To be classified in a class, the sample must be a member of that group alone and not a member of the others; membership is exclusive. So when a sample is fitted by the high class model, it is not sufficient to say that it is high, it must also be not low for the high designation to be accepted. Second, it is important to know when one has gone beyond the training set of the models. A neither classification simply means that. Based on the training set that it was provided, the computer cannot classify the unknown sample as it is sufficiently different from all the classes considered

The following is a scoring system that can be used under SIMCA considering only 2 classes:

1	if the sample belongs to class 1 (accepted by model 1 and rejected by model 2)
2	if the sample belongs to class 2 (accepted by model 2 and rejected by model 1)
0	if belonging to both classes (accepted by models 1 and 2)
3	if belonging to neither class (rejected by models 1 and 2)

By comparison, what is done in the quality control literature is simply binary: acceptable or not acceptable. While this is fine for manufacturing, this may not be desired in developmental work. In such a case, neither represents something not seen in the training set and hence should be viewed as something new. As such, it could be good (an improvement) or bad (another failure).

4.3.1 Historical Mean Time Series (HMTS)

While MHT is able to identify discriminating variables, it cannot be used to classify samples directly. Since no covariance matrix or standard deviation exists for a single sample the multivariate T-statistic cannot be calculated and hypothesis testing cannot be performed. To compensate for this, the historical mean time series (HMTS) approach was created. It is based on whether the sample lies within the confidence limits set about the mean using the pivotal method (Mendenhall and Sincich,1992) and can only handle 2 classes at a time.

If the sample falls into the confidence interval of one class (and not in another) it is assigned to that class and given the appropriate designation of 1 or 2 for class 1 and 2 respectively. While this resembles an approach used by Guthke, 1994 and univariate checking limit mentioned in Davis, et. al, 1996, this algorithm differs by allowing more than one variable to be considered as well as the novel scoring system involved. Another notable difference from the univariate checking limit is that the acceptance conditions for a sample is not constant but varies taking into account that the scatter in the data can vary with time. As mentioned before, this scoring system is based on SIMCA and recognizes the states of both and neither in addition to the classes. For numerical reasons that will be discussed later, the scoring for the last 2 cases have to be modified if HMTS is to be used.

To classify the lot, each variable under selection is first scored based on its time-average value during the time window. After each variable has been assigned a class score, the scores of all the variables considered are then averaged and this is the classification assigned to the sample, see Figure 4.1.

Mathematically, for variable j, class i, at time t:

$$\mu_{ij} - t_{\alpha/2}\left(\frac{s}{\sqrt{n_i}}\right) < x_{uj} < \mu_{ij} + t_{\alpha/2}\left(\frac{s}{\sqrt{n_i}}\right) \qquad (1)$$

μ_{ij}: mean of variable j of class i at time t

x_{uj}: variable j of sample whose class is unknown

s: standard deviation

n_i: number of samples in class I

α: significance level

X_{ui} is subject to the region described by eq. 1 for both models. It is assigned to the class where eq. 1 is fulfilled provided at the same time it is rejected by the other model. So, to be assigned to class 1, x_{uj} must be accepted by eq. 1 of model 1 and outside the boundary set by eq. 1 for model 2. Repeating for each time point of a selected time

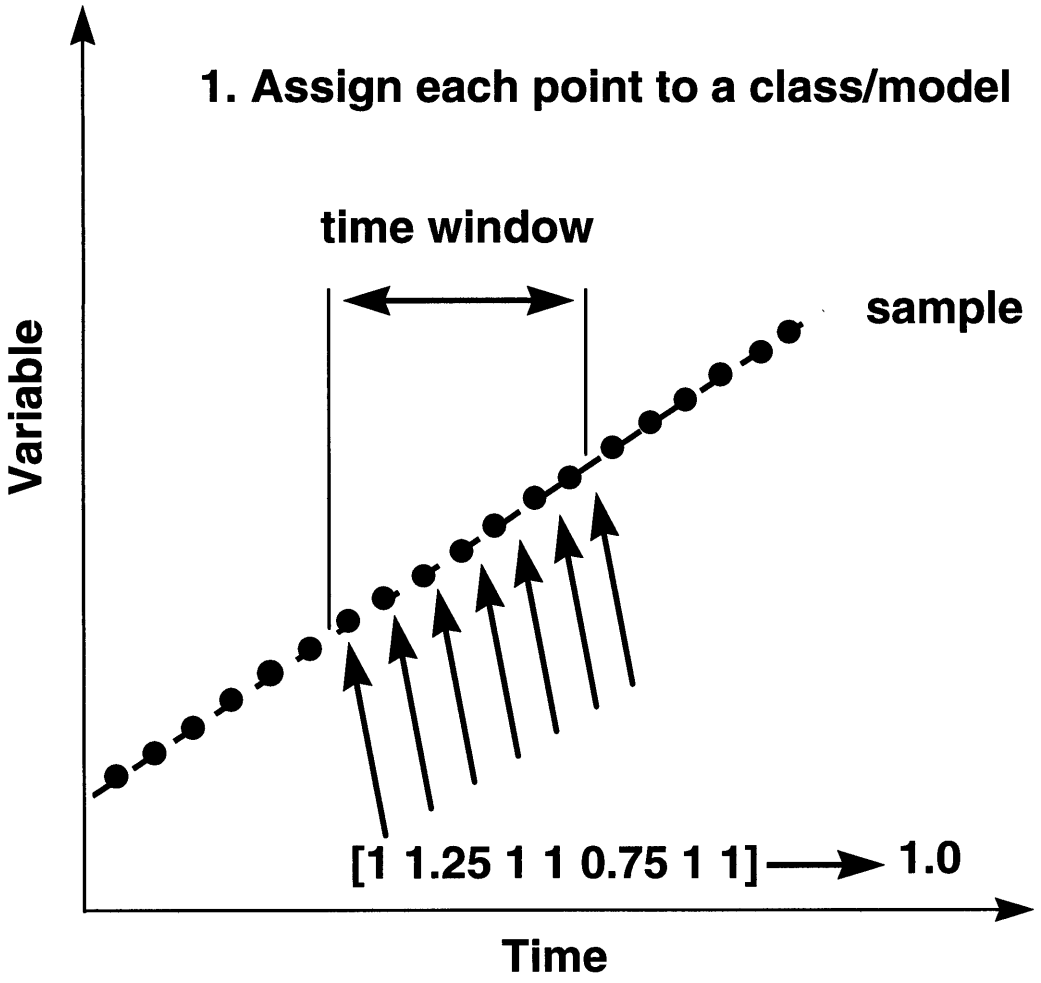


Figure 4.1 On-line classification by HMTS. First, calculate the average score for the class for each variable over the time window. Next, average the score over all the variable mean scores.

window, an average classification is assigned to each selected variable. After this, the classification is averaged over all the variables to obtain the sample's class score:

$$\text{score} = \frac{1}{c}\sum_{j=1}^{c}\bar{a}_{uj} \tag{2}$$

c: number of variables

$\bar{a}_{uj}$: time-window average classification of sample u for variable j

Because an average is calculated, a continuum range of values is needed if the correct interpretation is to be made. For example, if using only 2 variables and one of them lists the sample as belonging to class 0 (both) and the either as class 3 (neither) the average would be 2, which is not correct. To bypass this numerical obstacle, a revised scoring system is used where 1.5 represents the "both" category and 0.75, 1.25, 1.75, or 2.25 is assigned if the sample falls into the "neither" category depending on the following conditions:

(Note: by definition, x_{uj} must lie outside eq. 1 for both classes if it is neither.)

$$0.75 \text{ if } \|x_{uj} - x_{1j}\| < \|x_{uj} - x_{2j}\| \,\&\, \|x_{uj} - x_{1j}\| > \|x_{1j} - x_{2j}\| \tag{3}$$

$$1.25 \text{ if } \|x_{uj} - x_{1j}\| < \|x_{uj} - x_{2j}\| \,\&\, \|x_{uj} - x_{1j}\| < \|x_{1j} - x_{2j}\| \tag{4}$$

$$1.75 \text{ if } \|x_{uj} - x_{2j}\| < \|x_{uj} - x_{1j}\| \,\&\, \|x_{uj} - x_{2j}\| < \|x_{1j} - x_{2j}\| \tag{5}$$

$$2.25 \text{ if } \|x_{uj} - x_{2j}\| < \|x_{uj} - x_{2j}\| \,\&\, \|x_{uj} - x_{2j}\| > \|x_{1j} - x_{2j}\| \tag{6}$$

where:

$\|x_{uj} - x_{ij}\|$: distance between sample x_{uj} and model x_{ij}, I = class, j = variable

$\|x_{1j} - x_{2j}\|$: distance between models 1 and 2

Eq. 3 is a situation where the sample is outside of model 1 but even further away from model 2 with eq. 4 having the sample closer to model 1 but between models 1 and 2. Eq. 5 is an outlier of model 2 but on the side towards model 1 whereas eq. 6 represents a sample on the "far side" of model 2 away from model 1, see Figure 4.2.

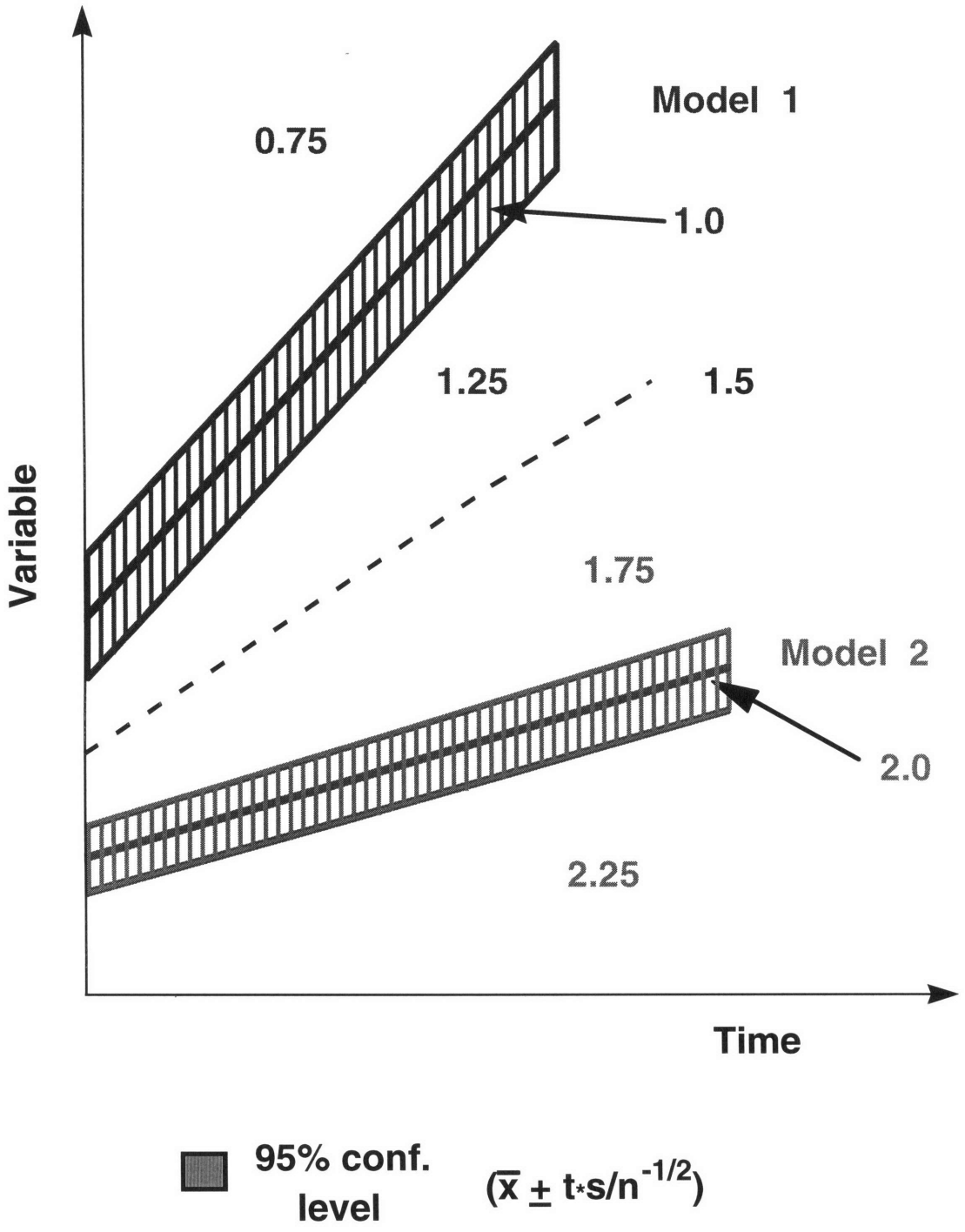


Figure 4.2 Classification regions for HMTS. $\overline{X}$ = mean, t = t-statistic, s = standard deviation, n= number of lots in training set of model.

4.3.2 AutoAssociative Neural Networks (AANN)

Autoassociative neural networks (AANN's) are feedforward networks trained to generate an identity association in which the network outputs approximate the given inputs using linear and sigmoidal transfer functions (Kramer, 1991;1992). The architecture of these networks is such that it does not learn the identity mapping perfectly. An internal constraint in the form of a bottleneck - a layer of hidden nodes smaller in dimension than either input or output - forces the network to reproduce an m-dimensional data set at its output using only f independent variables with $f < m$. The resulting identity mapping creates a global reduction of the data dimensionality and the extraction of significant features by the bottleneck nodes. The concept is similar to that of principal components analysis (PCA) where, if redundancy is present, the data can be reconstructed with a dimensional set smaller than the original data.

The architecture of the autoassociative network is shown in Figure 4.3.

The input to the network is a vector of process measurements, and so the size of the input layer corresponds to the dimension of the measurement vector. Since the output layer produces a reconstructed version of the inputs it is also of the same dimension as the input.

The autoassociative network shown in Figure 4.3 contains three hidden layers, the mapping layer involved in modeling the mapping function set, the bottleneck layer whose outputs represent the data features, and the demapping layer which modeling the demapping function set. The nodes of the mapping and demapping layers must have nonlinear transfer function. This is to allow for modeling of arbitrary mapping and demapping function sets, which are selected by the user. Nonlinear nodes, however, are not required in the bottleneck and in the output layers.

The training process involves the backpropagation method. The network weights are adjusted so that the reconstructed measurements vector at the output layer matches the input as closely as possible, in a least-squares sense, over the set of the training examples.

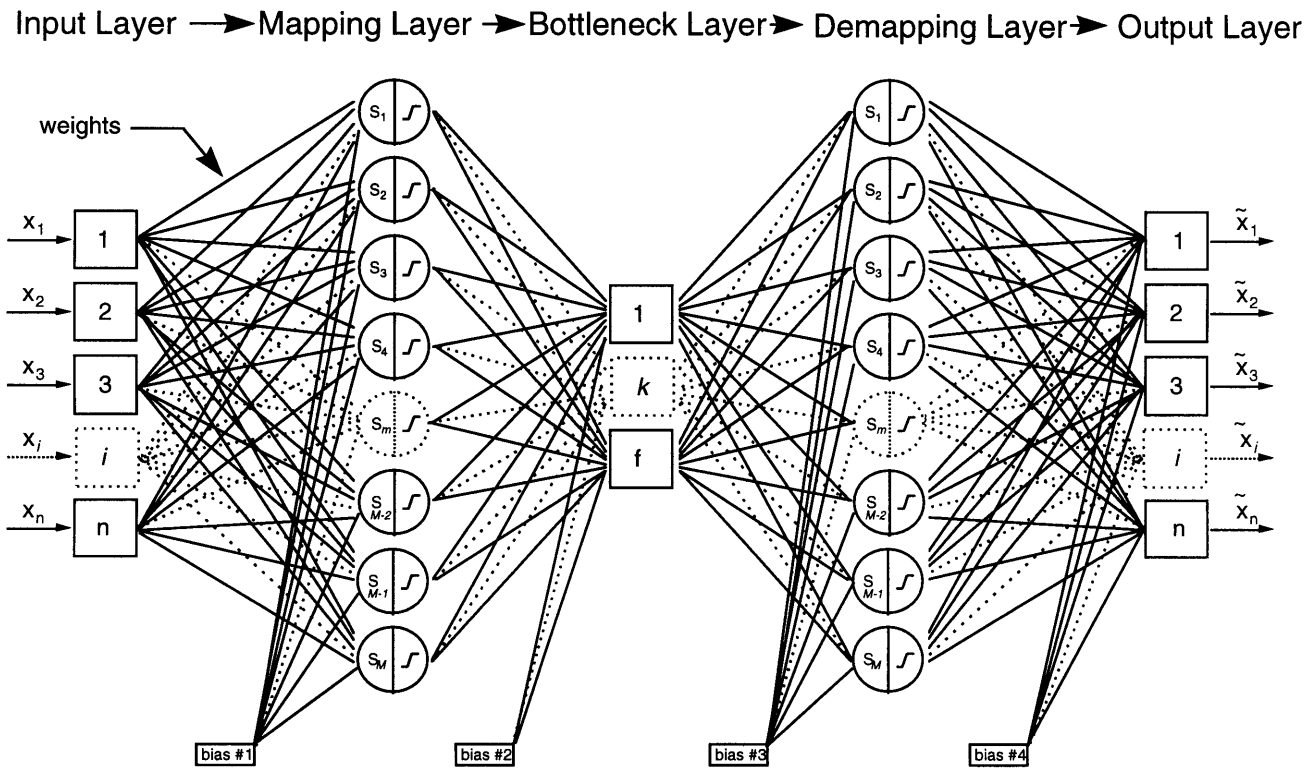


Figure 4.3 Architecture of AANN

The internal representation developed by the training procedure retains the maximum amount of information from the original data set for a given degree of dimensional compression represented by the number of nodes in the bottleneck layer.

Mathematically, AANN is a nonlinear generalized version of PCA. The input, mapping, and bottleneck layer together represent nonlinear function **G** which projects the inputs to a lower dimension space:

$$T_i = G_i(\mathbf{Y}) \qquad i = 1,\ldots,f \tag{7}$$

where T_i is the output of the ith bottleneck node and **Y** is the input vector. The bottleneck layer, the demapping layer and the output layer represent a second function **H** that reproduces an approximation of the inputs from the features at the output of the bottleneck layer:

$$Y'_i = H_j(\mathbf{T}) \qquad j = 1,\ldots,m \tag{8}$$

In summation, the data is first compressed to lower dimensionality and then reconstructed. The loss of information involved in this two-stage process is measured by the sum of the squares differences between the inputs and the outputs, summed over the training set:

$$SSE = \sum_{p=1}^{n} \sum_{i=1}^{m} (Y_i - Y_i')_p^2 \tag{9}$$

where n is the number of the training examples used to train the network. Minimizing the SSE during network training results in maximum signal reconstruction and in minimum information loss.

Once the networks have been trained for each class, classification can proceed. The scoring system used is the one presented earlier in section 4.3 utilizing 0, 1, 2, and 3. The procedure for classification is as follows:

1) Sample is fitted to both models, 1 and 2.

2) If the error generated by the sample lies within the standard reconstruction error - the mean error observed during the training phase + 2 standard deviations - it is tenatively assigned to that model's class either 1 or 2. Only if the sample is rejected by the other model, is it finally assigned the classification. So, a class 1 rank is acceptance eq. 10 and rejection by eq. 11 likewise for class 2.

$$1: \quad SSEtr1 - 2^* std1 < SSE < SSEtr1 + 2^*std1 \qquad (10)$$

$$2: \quad SSEtr2 - 2^* std2 < SSE < SSEtr2 + 2^*std2 \qquad (11)$$

SSE: sum of squared errors fitting sample to model

SSEtri: mean SSE observed in training set of model I

stdi: standard deviation of SSEtri observed during training phase

3) If it is observed that lot fits both models, it is given a score of 0 - both 14 and 15 are satisfied.

4) If it is observed that lot does not fit either class, it is given score of 3 - both conditions outlined in 14 and 15 are rejected

4.4 Results

Because of the length of the analysis, in the interest of space only one case study will be discussed. Both HMTS and AANN are applied to the case 1 data first presented in Chapter 2.

HMTS demonstrates a novel way to classify data looking directly at the measurements themselves. The AANN is used to show how powerful modeling tools can be affected by MHT and CA. Integrating results from MHT and CA, both techniques will classify the data based on a time window and a subset of variables. The goal here is to determine as early as possible the process outcome from the measurements- hence, the use of the time windows. These methods were designed for on-line use but since this was not possible at the time, on-line performance is simulated

by testing the algorithms on a cross-validation data set which is separate and not contained in the model training set.

Note: class 1 is referred to as high/good and class 2 low/bad in all of this analysis; the terms are used interchangeably.

Summary of Case 1 Data:

Type:	industrial fermentation
Mode:	fed-batch
Run length:	82 time points
Number of variables:	17 (see note in Chapter 2)
Number of classes:	2
Lots/class:	22 for high and 23 for low

4.4.1 Mean Hypothesis Testing (MHT)

Details of how these inputs are arrived at can be found in Chapter 2. From MHT, two variable sets are used. The first representing discriminating variables and consisting of 4, 5, 9, and 11. The other represents the nondiscriminating set of variables, 13, 14, 16, and 17. Also from MHT are the time windows used, namely 1-20, a nondiscriminating period, and 40-60, a discriminating region. Other time windows of 20-40 and 60-80 were also evaluated but not presented as the results do not differ significantly from the other 2.

One may note the discriminating set is not the one selected by the 4 variable combination. The reason for this comes from cluster analysis (CA). It was observed that the groups formed by [4, 5, 9, 11] set had less class overlap in the memberships than the other combinations, such as [7 8 11 12] for the time window of 40-60. This in turn should lead to less problems in misclassification.

4.4.2 Cluster Analysis (CA)

Chapter 3 explains in detail how these groups were selected. From CA, the goal is to show the effects of lot selection in the formation of the training sets. Table 4.1 shows the lot memberships of the groups that are used in this case study.

As mentioned previously, this grouping was chosen since it minimized the degree of class overlap in its group membership. The high and low classes can be represented quite satisfactorily by group 2 and 3 respectively.

Two types of training sets, homogeneous and heterogeneous, are used in this case study. Homogeneous means that the lots used in the training set come from clusters whose membership consists predominantly, if not all, of members of the same class. In this case study, this would be using group 2 lots for the training of the high class model and using group 3 lots for the low. Heterogeneous training sets attempts to capture as much of the variability present in the data as possible, so takes a representative sampling- members from different but large clusters. For high this would mean to take members from group 2 and group 1 - the 2 clusters where high class lots form a significant fraction of the leadership. For low, the members for the training set would come from group 3 and 5. Since the number of members of group 4 are small, they are ignored for training set selection. Table 4.2 and 4.3 show the respective training sets for each algorithm. **Note: unless otherwise stated, the default training set is the heterogeneous one for these subsequent analyses.**

4.4.3 Case Study 1 - HMTS

Table 4.4 shows a sample result of looking at one lot via HMTS. The number under each variable represents its average classification score received during that time window with the lot score being the average of the second row. It is important to note that using HMTS provides detail about the individual variable's behavior. The result in Table 4.4 implies that variable 4 acted very much like previous variable 4's of the high class during this time period. The values observed are consistent with the historical record of the training set for the high.

Table 4.1 Cluster analysis results using variables 4, 5, 9, 11 and time window 40-60.

Group	Lot Members
1	1, 7,11,13,37,43
2	2,3,5,6,8,9,10,12,14-22,39
3	23,24,25,27,28,30,32,33,35,36,41
4	4,34,38,44
5	26,29,31,40,42,45

Table 4.2 . Training Set: HMTS

Heterogeneous	Lots used
High Training Set (G1+G2)	2,5,8,10,14,16,18,20,22,1,7
Low Training Set (G3+G5)	23,25,28,32,35,41,26,29,31
Homogeneous	
High Training Set (G2)	2,5,8,10,14,16,18,20,22
Low Training Set (G5)	23,25,28,32,35,41

Table 4.3 . Training Set: AANN

Heterogeneous	Lots used
High Training Set (G1+G2)	1,2,3,5 6,7,8,9,10,12,14
Low Training Set (G3+G5)	23-33
Homogeneous	
High Training Set (G2)	2,3,5,6,8,9,10,12,14,15,16
Low Training Set (G5)	23,24,25,27,28,30,32,33,35,36
Poor Training Set1	
High Training (Hetero + Lot 4)	1,2,3,45 6,7,8,9,10,12
Low (Hetero)	23-33
Poor Training Set2	
High (Hetero)	1,2,3,5 6,7,8,9,10,12,14
Low (Hetero + lot 39)	23-32,39

Table 4.4. Sample result for one lot: variables 4,5,9,11 time window 40-60

variable	4	5	9	11
average score in time window	1.00	1.13	1.13	1.17
lot score	1.11			

4.4.3.a Effect of time windows and variable selection

Figure 4.4 is a graph of the HMTS classification results based on a heterogeneous training using variables 4, 5, 9, 11 over a time window of 1-20. The first 22 lots are high (class 1) and the next 23 are low (class 2). Similar to the results for MHT, it is observed that the dominant classification is 1.5 - the both category. Moving to a more discriminating time period, 40-60, the number of both classification drops but now the algorithm is able to correctly classify 67% of the lots, see Figure 4.5. As seen in the figure, many but not all of the lot scores move from 1.5 to their "true" values as the time window enters the more discriminating regions. One may note the odd behavior of some of the lots, noticeably the ones at position 1 and 27 on the graph- this will be examined in the next section 4.4.3.b.

The importance of variables is apparent when comparing Figure 4.6 to Figure 4.4. Variables 13, 14, 16, and 17 are at best able to classify only 2 out of 45 lots compared to the 30 correctly assigned by 4, 5, 9, and 11 for the same time window. Table 4.5 lists the breakdown of the classification results and their conditions.

The effectiveness of an algorithm can be limited by quality of the data presented to it. HMTS is no different from other methodologies in this regard. By comparing row 2 and 5 of Table 4.5, the effect of not capturing all the variability in the data is apparent. The model trained on the homogeneous set produces more of the neither category than the heterogeneous. Table 4.6 and Table 4.7 list the breakdown in scores by lot for both the homogeneous and heterogeneous case respectively. The dashed line represents the break between the high and low lots- 22 highs and 23 lows. The various G designations signify lots whose misclassification may be explained by the cluster analysis group from which they came.

Because HMTS uses the raw data directly in the classification, it was hoped that the evolution of a low run could be monitored. For example, it was hoped one could observe one variable go from 1.5 to 2 indicating a change from the both class to the low class and see if the other variables were also affected. Different time windows were used but the rate of change from both to bad (low) appears to be faster than the

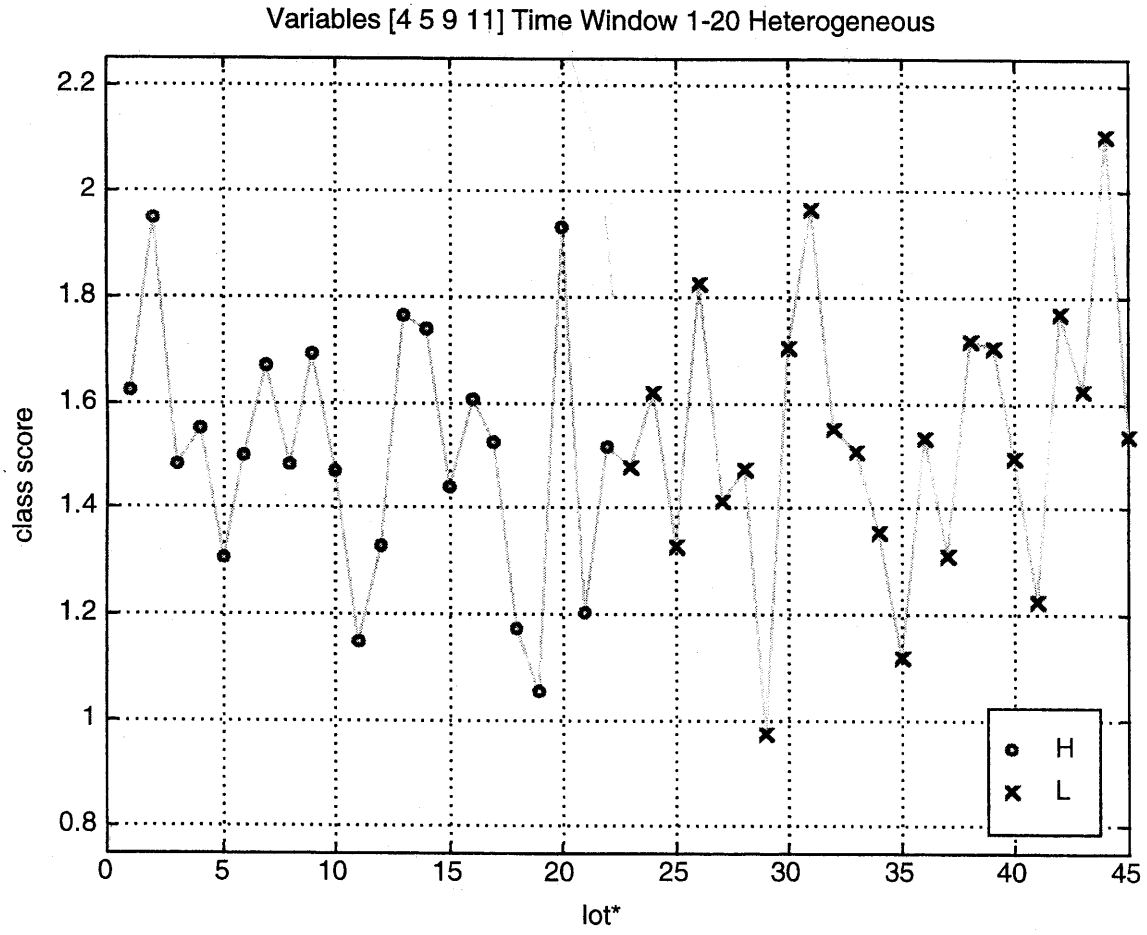


Figure 4.4 HMTS scoring of individual lots for a heterogeneous training set using discriminating variables and a nondiscriminating time window. Lot* denotes that the lots listed here are in order of training set first followed by cross-validation lot for each class.

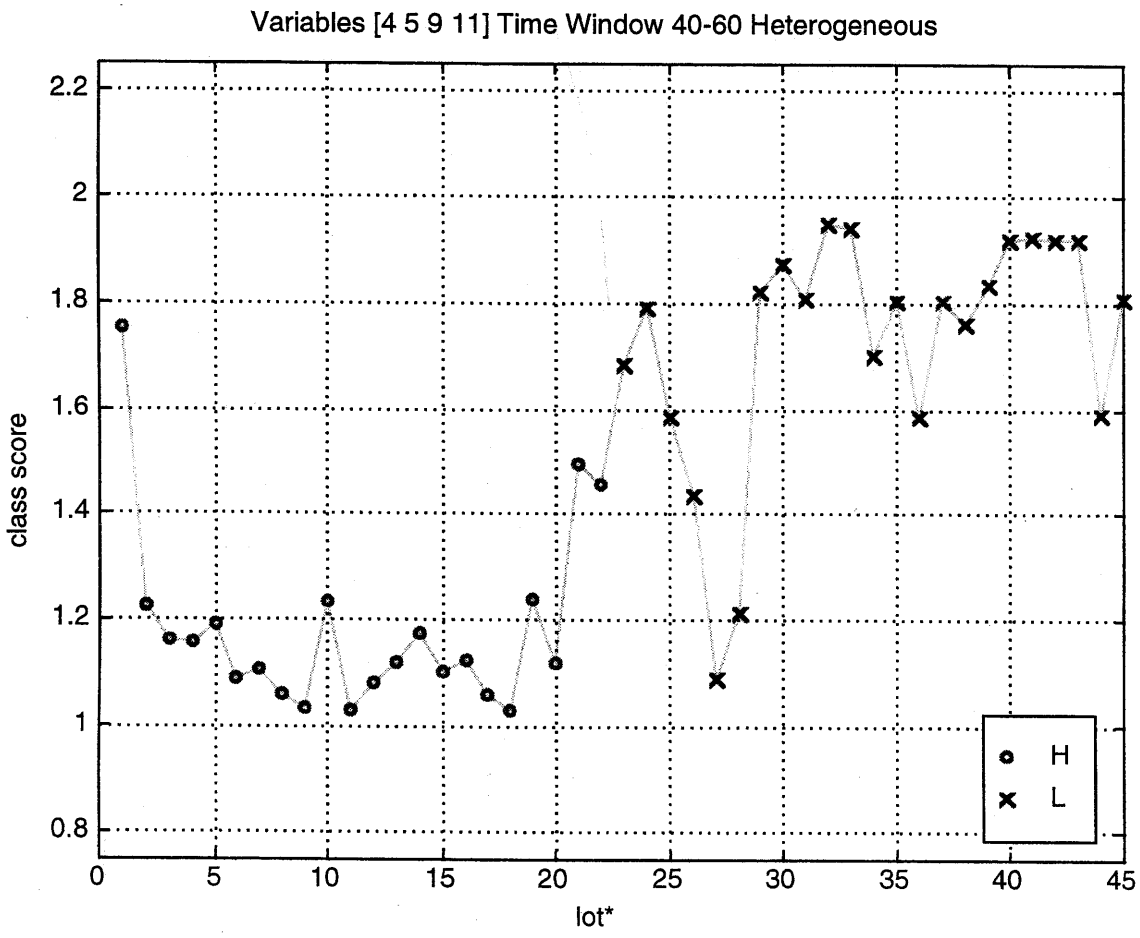


Figure 4.5 HMTS scoring of individual lots for a heterogeneous training set using discriminating variables and a discriminating time window.

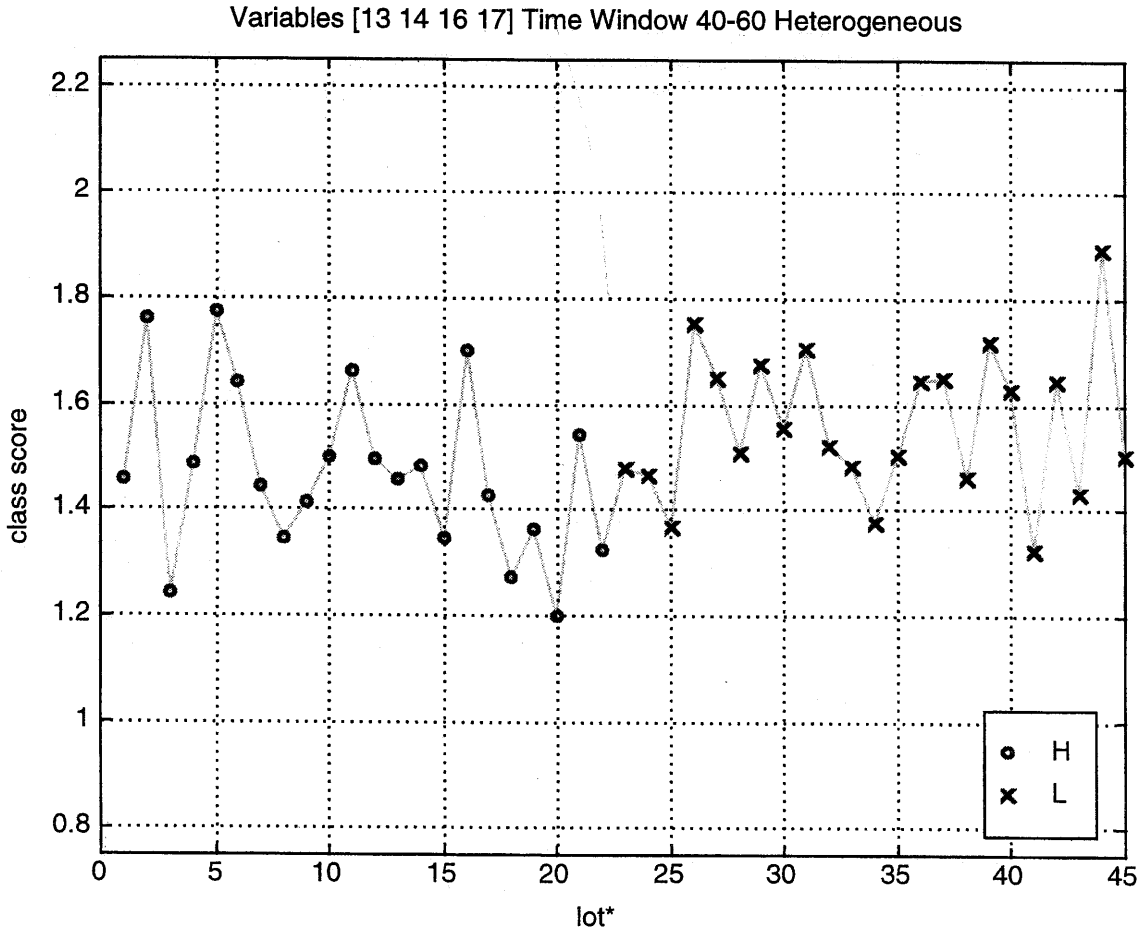


Figure 4.6 HMTS scoring of individual lots for a heterogeneous training set using nondiscriminating variables and a discriminating time window.

Table 4.5 MHTS classification results.

	Conditions	Correct class	Wrong class	Neither class	Both classes
1	[4 5 9 11] 1-20	8 (18%)	10 (22%)	3 (7%)	24 (53%)
2	[4 5 9 11] 40-60	30 (67%)	3 (7%)	6 (13%)	6 (13%)
3	[13 14 16 17] 1-20	2 (4%)	1 (2%)	13 (29%)	29 (64%)
4	[13 14 16 17] 40-60	2 (4%)	3 (7%)	4 (9%)	36 (80%)
5	Homogeneous [4 5 9 11] 40-60	26 (58%)	3 (7%)	11 (24%)	5 (11%)

Table 4.6 Homogeneous training set

Lot score	Lot	
1.4702	1	** G1
1.7202	4	x G4
1.4702	7	** G1
1.3095	11	** G1
1.0923	13	
1.2708	3	** G2
1.1071	6	
1.1429	9	
1.1667	12	
1.1071	15	
1.1280	17	
1.0685	19	
1.0387	21	
1.0952	2	
1.2262	5	
1.1667	8	
1.1310	10	
1.1161	14	
1.1042	16	
1.1012	18	
1.2321	20	
1.1637	22	
1.8423	26	
1.6220	29	** G5
1.7768	21	
1.6667	34	** G4
1.4554	37	** G1
1.7619	38	
1.2381	39	x G2
1.7024	40	** G5
1.7708	42	
1.1190	43	x G1
1.5744	44	** G4
1.5833	45	** G5
1.7887	24	
1.8690	27	
1.7589	30	
1.9911	33	
1.9613	36	
1.8661	23	
1.8304	25	
1.8274	28	
1.9851	32	
1.8958	35	
1.9048	41	

Table 4.7 Heterogeneous training class

Lot score	Lot	
1.7530	4	x G4
1.2292	3	
1.1637	6	
1.1577	9	
1.1935	12	
1.0893	15	
1.1071	17	
1.0625	19	
1.0357	21	
1.2351	11	
1.0327	13	
1.0833	2	
1.1220	5	
1.1756	8	
1.1042	10	
1.1250	14	
1.0595	16	
1.0298	18	
1.2381	20	
1.1220	22	
1.4970	1	** G1
1.4554	7	** G1
---	---	---
1.6815	34	** G4
1.7887	38	
1.5863	44	** G4
1.4345	37	** G1
1.0923	43	x G1
1.2143	39	x G2
1.8185	24	
1.8720	27	
1.8065	30	
1.9464	33	
1.9405	36	
1.6994	40	** G5
1.8036	42	
1.5863	45	** G5
1.8006	23	
1.7619	25	
1.8304	28	
1.9167	32	
1.9226	35	
1.9167	41	
1.9167	26	
1.5893	29	** G5
1.8065	31	

sampling frequency. In Figure 4.7, one is aware of the change so the time window was pushed back to 20-30, Figure 4.8. In this period, the both class is dominant. Moving to 30-35, the class separation seems to have already occurred, Figure 4.9. The change appears to be almost step-like in nature. There was no discernible pattern in the variable classifications themselves. Some of the low lots would have variables 4 and 5 getting a rating of 2 while 9 and 11 stayed around 1.5 and in some others, the situation would switch.

4.4.3.b Effect of training set

For the homogeneous case, the lots that are not correctly being classified come mainly from groups 1 and 4 for the high and groups 1,4, and 5 from the low. This result is not surprising in view that the training set consisted of groups 2 and 3 from high and low respectively. So, one would expect some incompatibilities across different groups.

For the heterogeneous training sets, the problematic lots include lots 4, 39, and 43 which are in the category of lots that resemble their rival classes.

4.4.4 Case Study 2 - AANN

The analysis presented here is the result of a collaboration with Dr. Silvio Bicciato, a visiting post-doc. His AANN algorithm demonstrates how MHT and CA have an impact on the performance of the network.

Figure 4.10 is a graph depicting how the AANN classifies the lots using variables 4, 5, 9, and 11 and a time window of 40-60. Consistent with MHT, this combination of variables and time windows provides good classification. Table 4.8 lists the other conditions examined.

4.4.4.a Effect of time windows and variable selection

In Chapter 2, MHT suggested that the time period from 30 to 82 would be discriminating. Using time windows of 20 units, AANN was performed on each section and the results were found to be consistent with the MHT. Many of the classifications in the early time windows (1-20 and 20-40), see conditions 3 and 4 in Table 4.8, are in

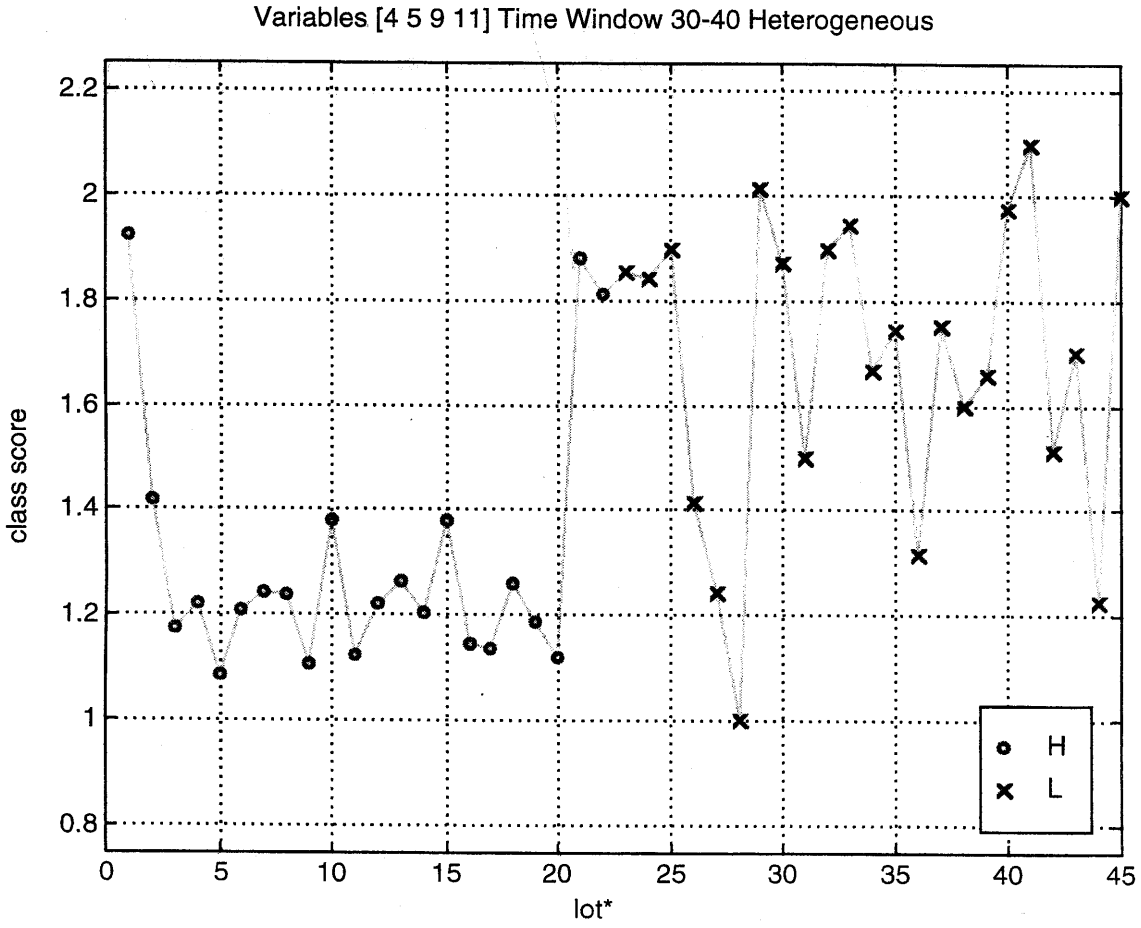


Figure 4.7 HMTS scoring of individual lots for a heterogeneous training set using discriminating variables and an 30-40 time window. Class separation has already occurred.

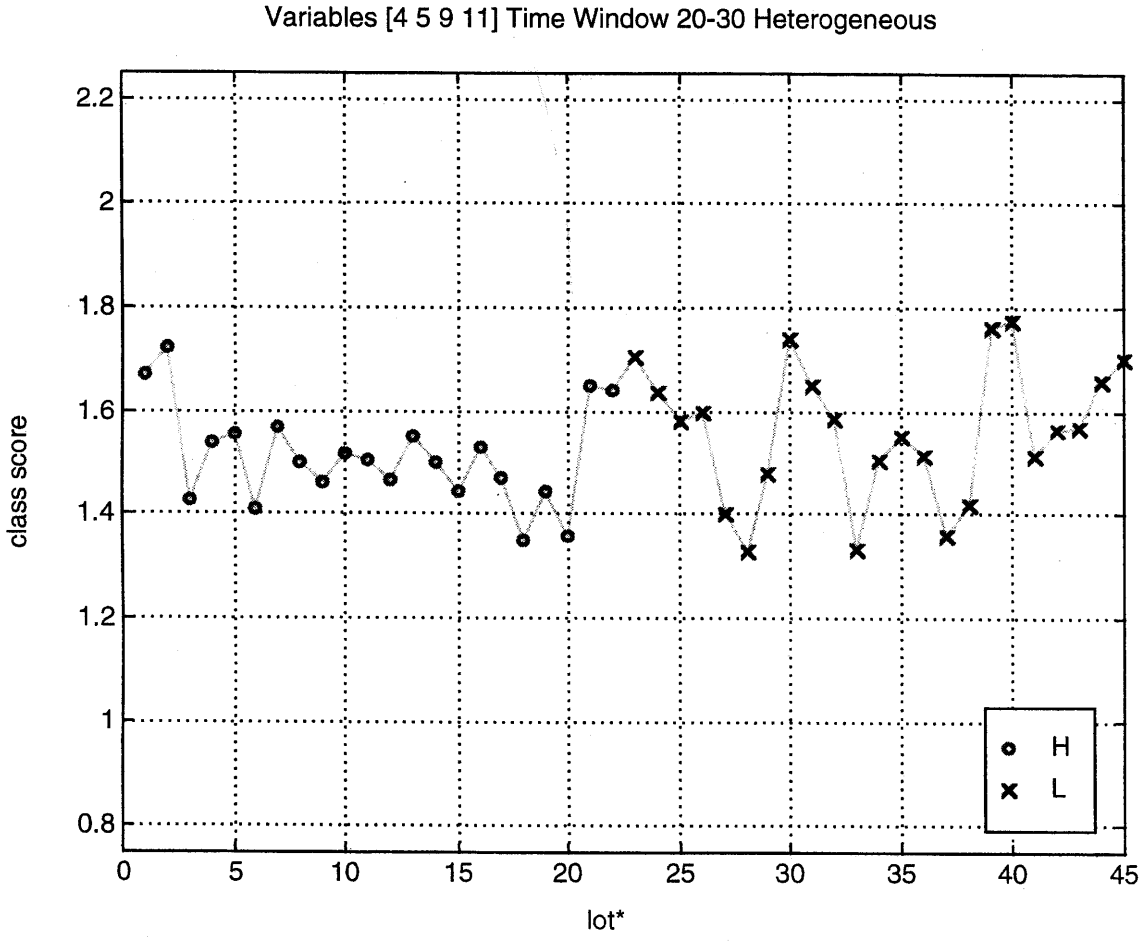


Figure 4.8 HMTS scoring of individual lots for a heterogeneous training set using discriminating variables and an 20-40 time window. No class separation observed.

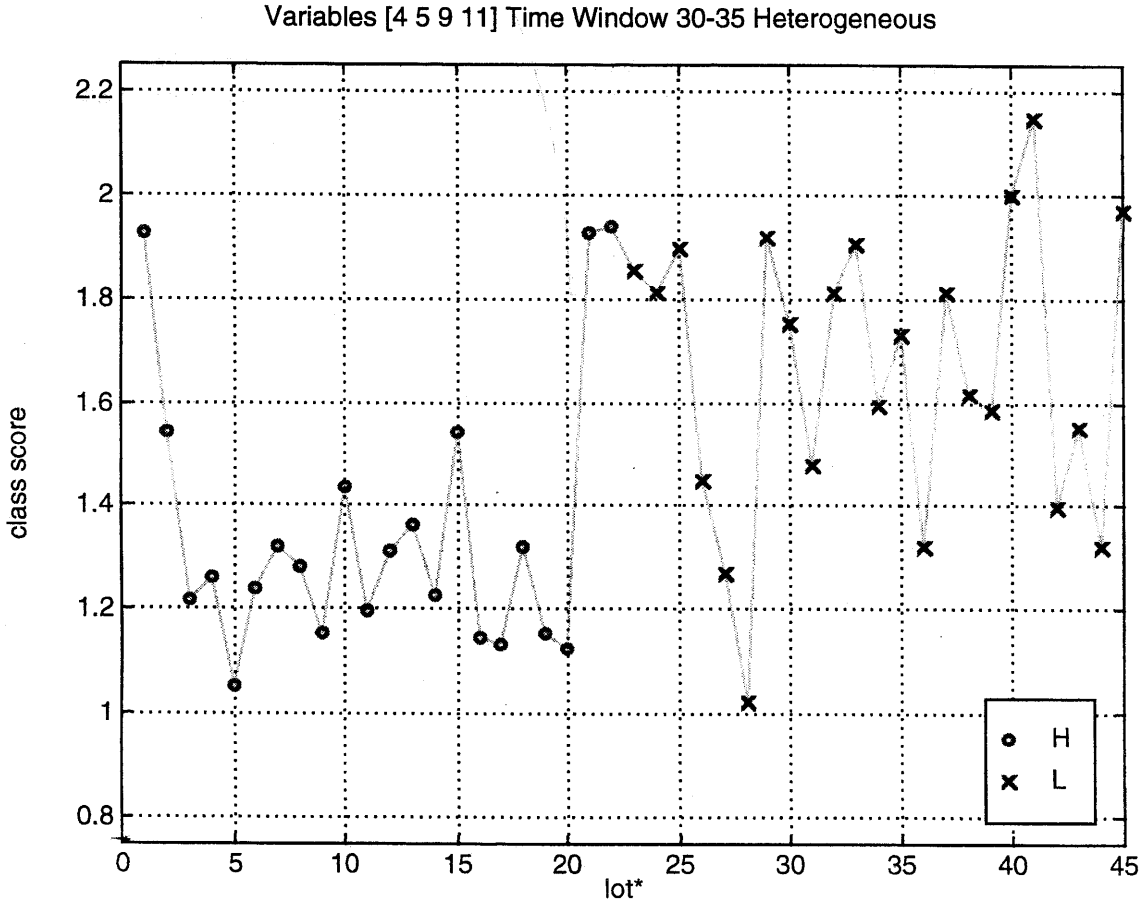


Figure 4.9 HMTS scoring of individual lots for a heterogeneous training set using discriminating variables and an 30-35 time window. Despite smaller time window, the class separation evolution cannot be observed.

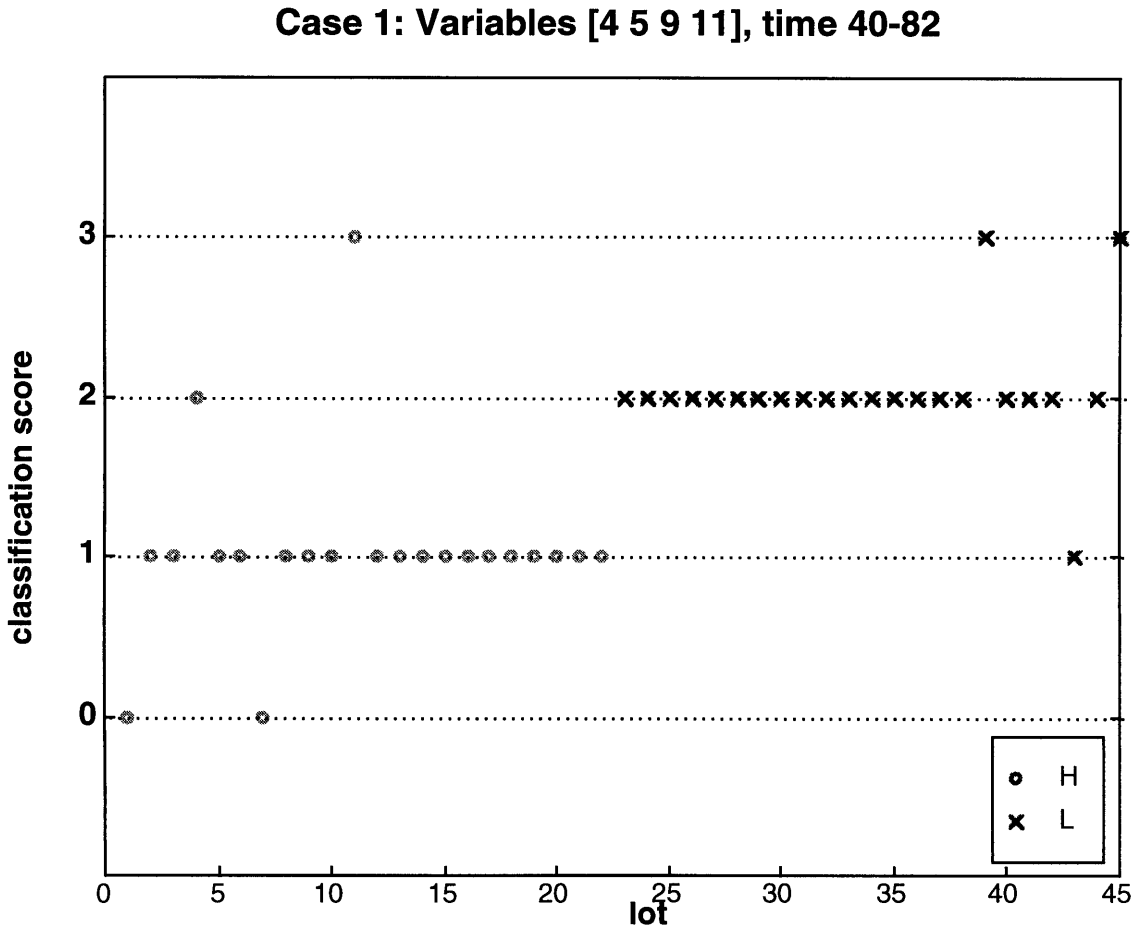


Figure 4.10 Sample classification by AANN using discriminating variables and time window.

Table 4.8 AANN classification results

	Conditions	Correct class	Wrong class	Neither class	Both classes
1	[4-17] 1-82	30 (67%)	1 (2%)	8 (18%)	6 (13%)
2	[4-17] 1-40	16 (36%)	1 (2%)	10 (22%)	18 (40%)
3	[4-17] 40-82	28 (62 %)	2 (4%)	14 (31%)	1 (2%)
4	[4 5 9 11] 1-20	4 (9%)	3 (7%)	3 (7%)	35 (77%)
5	[4 5 9 11] 20-40	17 (38%)	6 (13%)	2 (4%)	20 (44%)
6	[4 5 9 11] 40-60	38 (84%)	2 (4%)	3 (7%)	2 (4%)
7	[4 5 9 11] 60-82	37 (82%)	2 (4%)	6 (13%)	0 (0%)
8	[13 14 16 17] 1-20	3 (7%)	2 (4%)	8 (18%)	32 (71%)
9	[13 14 16 17] 40-60	18 (40%)	7 (16%)	3 (7%)	17 (38%)
10	Homogeneous [4 5 9 11] 40-60	35 (78%)	2 (4%)	8 (18%)	0 (0%)
11	poor set 1	36 (80%)	2 (4%)	1 (2%)	6 (13%)
12	poor set 2	21 (47%)	2 (4%)	1 (2%)	21 (13%)

the both category. As the window proceeds, the information becomes more discriminating and the number of correct classifications increases, reaching a maximum of 84% up from the 9%.

Just as important are what variables to select for modeling. As seen in rows 3 of Table 4.8, using all the variables even in a discriminating period does not produce the best classification. In fact what is interesting is that the performance is similar to that of HMTS using 4 variables. HMTS is a very simple technique whereas training an AANN can take a considerable amount of time spent on optimizing the network. This begs the question of why use a more sophisticated technique if a simple one is able to do the same thing. The short answer is there are cases where a more sophisticated model is needed but one first needs to determine if there is such a need in the first place. If all the information relevant to the modeling objective is found in a region that is not highly nonlinear, one may be able to use a linear technique and obtain comparable results.

4.4.4.b Effect of training set

As seen in section 4.4.3.b as well as in cluster analysis, some lots are atypical in that they are not simply outliers but have behave very similar to the rival class. Lots 4 and 39 are representative of this group. Lot 4 is a high lot which resembles a low while lot 39 is the opposite. Since these 2 are unusual one should not include them in training the model. But should one ignore such advice, the results can be seen in rows 6 and 7 of Table 4.8. While the effect of lot 4 does not seem to be as strong, using lot 39 has a devastating effect - lowering the classification rate from 84% to 47%. Hence, choosing this one lot can one mislead to believe that the algorithm is not working when it is. It should be stated that while lots such as 4 and 39 can be identified as problematic by trial and error- selectively choosing the lots- this is a time-consuming effort and having a systematic approach such as CA can shorten the search space.

4.5 Conclusions

Understanding the structure of the data is an important step prior to modeling. As seen in the case with both HMTS and AANN, time windows and discriminating

variables can have a major impact on modeling performance. Picking the wrong measurements or focusing on the wrong time period can be detrimental. Poor choice of variables can lead to erroneous conclusions about a model's capabilities. Knowing also when to look is just as important since discriminating variables do not always behave that way. The significance of this result is not just limited to modeling but can be extended to control. If there is a window of time that allows process discrimination, increasing the sampling frequency during that period might provide sufficient information to understand what causes the process to become substandard.

The effect of understanding the type of variability that exists in the data is also important. While there are methods to identify outliers, the situation is a different matter if there are distinct subclasses in the data. To capture the variability one needs to make sure that the training set has a representative sampling of the data that it is to model. This is illustrated by the improved classification rates obtained with the heterogeneous training sets over the homogeneous ones.

4.6 References

Davis, J., Bakshi, B, et. al, (1996)

"Process monitoring, Data Analysis, and Data Interpretation," in *First International Conference on Intelligent Systems in Process Engineering*, ed. by J.F. Davis, G. Stephanopoulos, and V. Venkatatsubramanian, AIChE Symposium Series 312, vol. 92, p 1-11.

Guthke, R and Ludwig, B (1994)

"Generation of Rules for Expert Systems by Statistical Method of Fermentation Data Analysis," *Acta Biotechnol*, 14, p13-26

Kramer, M.A. (1991)

"Nonlinear Principal Component Analysis Using Autoassociative Neural Networks", AIChE J., 37, 2, p 233-243.

Kramer, M.A. (1992)

"Autoassociative Neural Networks", *Computers Chem. Engng.*, 16, 4, p 313-328

Mendenhall, W and Sincich, T (1992)

Statistics for Engineering and the Sciences, 3rd ed., Dellen Publishing , San Francisco

Saner, U. and Stephanopoulos, G. (1992)

"Application of pattern recognition techniques to fermentation data analysis," *Modeling and Control of Biotechnical Processes 1992*, ed. by M.N. Karim and G. Stephanopoulos, IFAC, 10, p 123-128.

Wold, S. and Sjostrom, M. (1977)

"SIMCA: a method for analyzing chemical data in terms of similarity and analogy," *Chemometrics: Theory and Application*, ed. by B.R. Kowalski, ACS Symposium Series, 52

4.7 Appendix

function [VCls,scr,ovrlscr] = Vclassify(Dcv,Dt1,Dt2,VL,LotS)

```
% DATE: 12/2/96, Roy Kamimura, copyright (c) by MIT
% DEFINITION:
% Classify lots in Dcv based on nearness of point to means
% calculated from Dt1 and Dt2.
% This is designed to process only 1 lot at a time.
%
% Dcv = cross-validation data set
% Dt1 = training set of class 1
% Dt2 = training set of class 2
% VL =  variable list
% LotS = standard lot size
% VCls =
% scr  = score
% ovrlscr = overall score (mean)

[Dm1, Dstd1] = DatMean(Dt1,LotS);
[Dm2, Dstd2] = DatMean(Dt2,LotS);

n1 = row(Dt1)/LotS;
```

```
n2 = row(Dt2)/LotS;

a1 = tinv(0.975,n1-1)/sqrt(n1);
a2 = tinv(0.975,n2-1)/sqrt(n2);

Dm1H = Dm1 + a1*Dstd1;
Dm1L = Dm1 - a1*Dstd1;
Dm2H = Dm2 + a2*Dstd2;
Dm2L = Dm2 - a2*Dstd2;

for j = 1:col(VL),
  D = Dcv(:,VL(1,j));
  for k = 1:LotS,
    if (D(k,:) < Dm1H(k,VL(1,j))) & (D(k,:) > Dm1L(k,VL(1,j))),
      Cls1Cnter(k,1) = 1;
    else
      Cls1Cnter(k,1) = 0;
    end
    if (D(k,:) < Dm2H(k,VL(1,j))) & (D(k,:) > Dm2L(k,VL(1,j))),
      Cls2Cnter(k,1) = 2;
    else
      Cls2Cnter(k,1) = 0;
    end
    if (Cls1Cnter(k,1) == 1) & (Cls2Cnter(k,1) == 0),
      Cls(k,1) = 1; % class 1
    end
    if (Cls1Cnter(k,1) == 0) & (Cls2Cnter(k,1) == 2),
      Cls(k,1) = 2; % class 2
    end
    if (Cls1Cnter(k,1) == 1) & (Cls2Cnter(k,1) == 2),
      Cls(k,1) = 1.5; % both classes
    end
    if (Cls1Cnter(k,1) == 0) & (Cls2Cnter(k,1) == 0), % assign to nearest class
% 0.75 = outlier near class 1 but far away from class 2
% 1.25 = outlier that is between 1 and 2 but closer to 1
% 1.75 = outlier that is between 1 and 2 but closer to 2
% 2.25 = outlier near class 2 but far away from class 1
      disx1 = sqrt((D(k,:) - Dm1(k,VL(1,j)))^2);
      disx2 = sqrt((D(k,:) - Dm2(k,VL(1,j)))^2);
      dis12 = sqrt((Dm1(k,VL(1,j)) - Dm2(k,VL(1,j)))^2);
      if (disx1 < disx2),
        if (disx1 > dis12),
          Cls(k,1) = 0.75;
        else
          Cls(k,1) = 1.25;
        end
      end
      if (disx2 <= disx1),
        if (disx2 < dis12)
          Cls(k,1) = 1.75;
        else
          Cls(k,1) = 2.25;
        end
      end
```

```
    end
  end
  VCls(:,j) = Cls(:,1);
  Cls = [];
  end
end

scr(1,:) = VL;
scr(2,:) = mean(VCls);
for l = 1:col(VL),
  no0 = find(VCls(:,l) ~= 1.5);
  scr(3,l) = mean(VCls(no0,l));
end

ovrlscr = [mean(scr(2,:)) mean(scr(3,:))];
```

The following codes are the Matlab functions to perform the entire backpropagation algorithm (tbpx4.m) and the to calculate the deltas (learnbpm.m).

function [w1,b1,w2,b2,w3,b3,w4,b4,i,tr,bot,output] =
tbpx4(w1,b1,f1,w2,b2,f2,w3,b3,f3,w4,b4,f4,p,t,tp)

```
%TBPX4 Train 4-layer feed-forward network w/fast backpropagation.
%
%       [W1,B1,W2,B2,W3,B3,W4,B4,TE,TR] = TBP3(W1,B1,F1,W2,B2,F2,W3,B3,F3,W4,B4,F4,P,T,TP)
%         Wi - SixR weight matrix of ith layer.
%         Bi - Six1 bias vector of ith layer.
%         F  - Transfer function (string) of ith layer.
%         P  - RxQ matrix of input vectors.
%         T  - S2xQ matrix of target vectors.
%         TP - Training parameters (optional).
%       Returns:
%         Wi - new weights.
%         Bi - new biases.
%         TE - the actual number of epochs trained.
%         TR - training record: [row of errors]
%
%       Training parameters are:
%         TP(1) - Epochs between updating display, default = 25.
%         TP(2) - Maximum number of epochs to train, default = 1000.
%         TP(3) - Target Error Gradient , default = 1e-3.
%         TP(4) - Learning rate, 0.01.
%         TP(5) - Learning rate increase, default = 1.05.
%         TP(6) - Learning rate decrease, default = 0.7.
%         TP(7) - Momentum constant, default = 0.9.
%         TP(8) - Maximum error ratio, default = 1.04.
%       Missing parameters and NaN's are replaced with defaults.
%
%
```

```
%
% Modified by S. Bicciato - MIT - October 96
% from 'tbpx3.m' by Mark Beale, 1-31-92
% Revised 12-15-93, MB
% Copyright (c) 1992-94 by the MathWorks, Inc.
% $Revision: 1.1 $  $Date: 1994/01/11 16:29:35 $

if nargin < 14,error('Not enough arguments.'),end

% TRAINING PARAMETERS

if nargin == 13, tp = []; end
tp = nndef(tp,[25 1000 0.02 0.01 1.05 0.7 0.9 1.04]);
df = tp(1);
me = tp(2);
eg = tp(3);
lr = tp(4);
im = tp(5);
dm = tp(6);
mc = tp(7);
er = tp(8);
df1 = feval(f1,'delta');
df2 = feval(f2,'delta');
df3 = feval(f3,'delta');
df4 = feval(f4,'delta');

dw1 = w1*0;
db1 = b1*0;
dw2 = w2*0;
db2 = b2*0;
dw3 = w3*0;
db3 = b3*0;
dw4 = w4*0;
db4 = b4*0;
MC = 0;

% PRESENTATION PHASE
a1 = feval(f1,w1*p,b1);
a2 = feval(f2,w2*a1,b2);
a3 = feval(f3,w3*a2,b3);
a4 = feval(f4,w4*a3,b4);
e = t-a4;
SSE = sumsqr(e);


% PLOTTING FLAG
[r,q] = size(p);
[s4,q] = size(t);
plottype = (max(r,s4) == 1) & 0;

% TRAINING RECORD
%tr = zeros(2,me+1);

SSE_old = SSE;
```

```
delta =SSE;
tr(1:2,1) = [SSE; delta];
Rec = [];
for i=1:row(t)
 RecVar = corrcoef(t(i,:),a4(i,:));
 Rec = [Rec; RecVar(2,1)];
end
RecMin = min(Rec);

% PLOTTING

message = sprintf('TRAINBPX: %%g/%g epochs, lr = %%g, SSE = %%g, RecMin = %%g, delta =
%%g.\n',me);
fprintf(message,0,lr,SSE,RecMin,delta)
%h = my_plottr(tr(1:2,1),0,eg);

% BACKPROPAGATION PHASE
d4 = feval(df4,a4,e);
d3 = feval(df3,a3,d4,w4);
d2 = feval(df2,a2,d3,w3);
d1 = feval(df1,a1,d2,w2);
for i=1:me

 % CHECK PHASE
%  if  delta < eg, i=i-1; break, end
 if  RecMin > 0.9 | delta < 5e-5, i=i-1; break, end

 % LEARNING PHASE
 [dw1,db1] = learnbpm(p,d1,lr,MC,dw1,db1);
 [dw2,db2] = learnbpm(a1,d2,lr,MC,dw2,db2);
 [dw3,db3] = learnbpm(a2,d3,lr,MC,dw3,db3);
 [dw4,db4] = learnbpm(a3,d4,lr,MC,dw4,db4);
 MC = mc;
 new_w1 = w1 + dw1; new_b1 = b1 + db1;
 new_w2 = w2 + dw2; new_b2 = b2 + db2;
 new_w3 = w3 + dw3; new_b3 = b3 + db3;
 new_w4 = w4 + dw4; new_b4 = b4 + db4;

 % PRESENTATION PHASE
 new_a1 = feval(f1,new_w1*p,new_b1);
 new_a2 = feval(f2,new_w2*new_a1,new_b2);
 new_a3 = feval(f3,new_w3*new_a2,new_b3);
 new_a4 = feval(f4,new_w4*new_a3,new_b4);
 new_e = t-new_a4;
 new_SSE = sumsqr(new_e);

 % MOMENTUM & ADAPTIVE LEARNING RATE PHASE
 if new_SSE > SSE*er
  lr = lr * dm;
  MC = 0;
 else
  if new_SSE < SSE
   lr = lr * im;
 end
```

```
  w1 = new_w1; b1 = new_b1; a1 = new_a1;
  w2 = new_w2; b2 = new_b2; a2 = new_a2;
  w3 = new_w3; b3 = new_b3; a3 = new_a3;
  w4 = new_w4; b4 = new_b4; a4 = new_a4;
  e = new_e; SSE = new_SSE;
  bot = a2;
  output = a4;
  delta = abs(SSE_old-SSE);
  SSE_old = SSE;
  Rec = [];
  for j=1:row(t)
     RecVar = corrcoef(t(j,:),a4(j,:));
     Rec = [Rec; RecVar(2,1)];
  end
  RecMin = min(Rec);

 % BACKPROPAGATION PHASE
  d4 = feval(df4,a4,e);
  d3 = feval(df3,a3,d4,w4);
  d2 = feval(df2,a2,d3,w3);
  d1 = feval(df1,a1,d2,w2);
 end

 % TRAINING RECORD
 tr(1:2,i+1) = [SSE; lr];

 % PLOTTING

 if rem(i,df) == 0
    fprintf(message,i,lr,SSE,RecMin,delta)
%     h = my_plottr(tr(1:2,1:(i+1)),0,eg,h);
 end
end

% TRAINING RECORD
tr = tr(1:2,1:(i+1));

% PLOTTING
if rem(i,df) ~= 0
%   h = my_plottr(tr(1:2,1:(i+1)),0,eg,h);
  fprintf(message,i,lr,SSE,RecMin,delta)
end

% WARNINGS
if delta > eg & RecMin < 0.9
 disp(' ')
 disp('TRAINBP: Network error did not reach the error goal.')
 disp('  Further training may be necessary, or try different')
 disp('  initial weights and biases and/or more hidden neurons.')
 disp(' ')
end

function [dw,db] = learnbm(p,d,lr,mc,dw,db)
%LEARNBPM Backpropagation learning rule with momentum.
```

```
%
%       [dW,dB] = LEARNBM(P,D,LR,MC,dW,dB)
%         P  - RxQ matrix of input vectors.
%         D  - SxQ matrix of error vectors.
%         lr - the learning rate.
%         mc - momentum constant.
%         dW - SxR weight change matrix.
%         dB - Sx1 bias change vector (optional).
%       Returns:
%         dW - a new weight change matrix.
%         dB - a new bias change vector (optional).
%
%       See also NNLEARN, BACKPROP, SIMFF, INITFF, TRAINBPX.

% Mark Beale, 1-31-92
% Revised 12-15-93, MB
% Copyright (c) 1992-93 by the MathWorks, Inc.
% $Revision: 1.1 $  $Date: 1994/01/11 16:25:05 $

if nargin < 5,error('Not enough input arguments'),end

x = (1-mc)*lr*d;
dw = mc*dw + x*p';
if nargout == 2
  [R,Q] = size(p);
  db = mc*db + x*ones(Q,1);
end
```

Chapter 5:

Summary and Significance of Work

5.1 Summary of Thesis Main Points

As stated in the first chapter, the objectives of this thesis are the following 3 topics:

1) identification of discriminating variable and time windows for class discrimination
2) systematic approach for designing training sets for data-driven models
3) early process outcome classification via models based on historical records

The first objective has been achieved by the development of mean hypothesis testing (MHT). This method can provide useful insight into the structure of data in terms of their ability to provide discriminating information. MHT identifies which variables and time windows behave differently on average between 2 classes of data. As shown in this thesis, MHT's ability to identify discriminating information has been verified in 3 manners: by visual inspection, by agreement with decision trees, and by the classification results of the historical mean time series and autoassociative neural networks. The impact of using discriminating variables and time windows is particularly noticeable in the modeling results where the wrong choice of inputs can cause an otherwise effective algorithm to perform poorly. The key again is not to model everything which is there but what is relevant to the modeling objective and MHT provides a means to accomplish this task better than existing methodologies. MHT

competitors such as PCA and neural networks often require a great deal of trial and error searching the variable space to find the same information MHT does in a systematic fashion, especially if the results of the univariate are used to reduce the number of combinations to consider. Also, this method is more flexible than PCA in that it can handle variable sets that contain no redundancy which PCA can have problems with since it relies on dimensional reduction. MHT's ability to handle more than 1 variable also makes it a strong contender with decision trees. Decision trees are a powerful tool for identifying discriminating variables and they provide more detailed information than MHT, however, the analysis is limited only to single variables and pairwise data if ratios are involved. As such, decision trees cannot deal with variable interaction effects which MHT can handle.

The second objective of this thesis has been achieved by PC1-time series cluster analysis, which provides a novel way to analyze multiple time-series data which is common to many chemical processes. In addition, it provides a methodology for exploring the variability in the data. While the results are not as decisive as MHT, cluster analysis is nevertheless useful as a modeling aid to help select members of training sets. It provides a systematic basis to selectively remove or add samples. As seen with the case with autoassociative neural net, selection of the wrong lots in the training sets can have adverse effects. Also, seen with this method is the value of identifying subclasses within the data. If the class is very heterogeneous - composed of many subclasses - this needs to be taken into account when selecting members for the training set of the model. One choice is to model each subclass separately, which is recommended for data-driven models such as PCA. Another approach is to take a representative sampling from each subclass to insure that enough of the class variability is captured in the training set. The latter appears to work better with techniques such as neural networks which are robust enough to handle large variability in the data. The traditional approach of randomly selecting samples to go into a training set simply does not address the issue of data heterogeneity.

It should be noted that both MHT and PC1 Time Series Clustering are generic enough to be used in situations where one is comparing or contrasting any 2 groups of data, not just process data. There are plans in the works to apply these methodologies to analyzing even gene sequences for patterns. MHT has even wider applicability as it is not limited to dealing with just time series data.

The last objective of this thesis was to determine as early as possible the outcome of a bioprocess from historical records. This was achieved by using the historical mean time series and the autoassociative network approaches. Achieving correct classification rates as high as 84% it is possible to classify lots using only a window of time. Hence, it is not required that the entire time profile be known to classify a run but just a relevant portion.

In summary, the algorithms presented in this thesis are a toolbox that any "database miner" can use to extract useful information about his or her system.

5.2 Ramifications of Findings

The results presented in this thesis suggest much about the role and use of data-driven models in process modeling. First, data-driven models can be used in process classification but significant premodel processing must be initially performed. Recall that most data-driven models are empirical in nature and hence are not derived from basic biological, chemical, and physical principles. Therefore, it is important that the structure of the data be well-understood prior to modeling to ensure that the assumptions underlying these models are not violated. Second, a rational basis for comparing models is needed. There are several comparisons made on the performance of different algorithms and suggestions made explaining the superiority of one methodology over another. While these studies are informative and useful, they can be misleading. For example, a nonlinear technique such as artificial neural networks (ANN's) will be able to capture nonlinear aspects better than linear methods such as principal components analysis but this choice is only appropriate if the behavior of interest is nonlinear. In the cases presented in this

thesis, the importance of time windows strongly indicates that the entire time profile is not required for good classification to occur. It may be that all the discriminating information is in a time period where the process behavior is relatively linear. If this is indeed the case, it is appropriate to compare PCA's classification accuracy to that of the ANN's over this time window. PCA's performance may be compromised by its attempt to model nondiscriminating nonlinear regions. The focus of the models should not be on what is present but what is important. This last statement is what current data-driven models do not consider. A systematic analysis of the structure of the data is required if accurate conclusions are to be drawn from these models.

5.3 Future Work

Some aspects of the methodologies presented in this thesis are open to further investigation. One drawback to MHT is its inability to describe in detail what ways are the variables discriminating. In its current form MHT simply states which variables behave differently between 2 classes but it does not describe in what way. If the MHT results are combined with a histogram analysis, more detailed information may be extracted. For example, from MHT, knowledge of which variables and time windows to consider is known. A histogram is made of the values expressed by the discriminators at the relevant time periods. By analyzing the distribution of class lots and the corresponding values, it is possible to determine what percentage of high/low lots fall or are above a threshold value in much the same way as decision trees. This approach can be investigated in more detail.

Another area to explore further is in cluster analysis. The PC1-Time Series Clustering uses the most simple approach to cluster analysis namely agglomerative clustering by nearest neighbor. Several other cluster analysis methodologies exist but were not explored. It would be interesting to see how other clustering techniques would perform.

In using the HMTS model, an attempt was made to determine if it were possible to detect the deviation of a good lot into a bad one over time by

observing which of the process variables fell first from the good class to the bad class and if this caused the other variables to be adversely affected in some way. Unfortunately, the sampling rate of the data did not allow a transition period to be visualized. However, when the SIMCA classification method was modified with fuzzy logic rules developed by Prof. Hiroshi Shimizu, a visiting professor, there were indications of a transition period. It was observed that during the transition between the both and high classes the fuzzy grade of the 0 (both) score would start to decrease and the 1 (high) score would increase. A fuzzy grade placed on top of a SIMCA-type classification would be an area of further study to see if this could be used to determine how low class lots are generated. Are they deviations from a high class lot or do they go straight from both to low? The answer to this question is still not known.