ABSTRACT

Three Applications of Linear Dimension Reduction

Gabriel J. Odom, Ph.D.

Chairperson: Dean M. Young, Ph.D.

Chairperson: Amanda S. Hering, Ph.D.

Linear Dimension Reduction (LDR) has many uses in engineering, business, medicine, economics, data science and others. LDR can be employed when observations are recorded with many correlated features to reduce the number of features upon which statistical inference may be necessary. Some of the benefits of LDR are to increase the signal to noise ratio in noisy data, rotate features into orthogonal space to reduce feature correlation effects, reduce the number of parameters to estimate, and decrease computational and memory costs associated with model fitting. In this manuscript, we will discuss applications of LDR to poorly-posed classification, ill-posed classification, and statistical process monitoring.

Three Applications of Linear Dimension Reduction

by

Gabriel J. Odom, B.Th., B.A., B.S., M.Div., Th.D.

A Dissertation

Approved by the Department of Statistical Science

_____

Jack D. Tubbs, Ph.D., Chairperson

Submitted to the Graduate Faculty of
Baylor University in Partial Fulfillment of the
Requirements for the Degree
of
Doctor of Philosophy

Approved by the Dissertation Committee

_____

Dean M. Young, Ph.D., Chairperson

_____

Amanda S. Hering, Ph.D., Chairperson

_____

David J. Kahle, Ph.D.

_____

James D. Stamey, Ph.D.

_____

Cindy Riemenschneider, Ph.D.

Accepted by the Graduate School
December 2017

_____

J. Larry Lyon, Ph.D., Dean

Page bearing signatures is kept on file in the Graduate School.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AD	Adaptive-Dynamic
CER	Conditional Error Rate
HLDR	Heteroscedastic Linear Dimension Reduction
IC	In Control
LDA	Linear Discriminant Analysis
LDF	Linear Discriminant Function
LDR	Linear Dimension Reduction
MVSPC	Multivariate Statistical Process Control
PCA	Principal Component Analysis
QDA	Quadratic Discriminant Analysis
QDF	Quadratic Discriminant Function
SPC	Statistical Process Control
SVD	Singular Value Decomposition
WWT	Waste-Water Treatment

ACKNOWLEDGMENTS

I would like to thank my wife, Tremaine, and my family for their support through this endeavor.

Additionally, I would like to thank my advisors, Mandy Hering and Dean Young, for their countless hours of guidance and support; my co-authors Phil D. Young, John A. Ramey, Kathryn B. Newhart, Tzahi Y. Cath, and Ben J. Barnard for their scientific expertise and hard work; my fellow students, especially Whitney V. Worley, for their support and patience; and the additional members of my committee, David J. Kahle, Cindy Riemenschneider, and James D. Stamey, for their valuable time and feedback.

Further, I thank the Colorado School of Mines for their partnership on my research into waste-water treatment monitoring. Also, I acknowledge the partial funding for my research from the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) under Award No: OSR-2015-CRG4-2582 and the National Science Foundation PFI:BIC Award No: 1632227.

Finally, I express gratitude to my Lord and Savior, Jesus Christ, and to the precious Holy Ghost, for without their grace I would have been even more hopeless.

DEDICATION

To my wife and family

CHAPTER ONE

Introduction

Linear Dimension Reduction (LDR) has many uses in engineering, business, medicine, economics, and other disciplines. LDR is a technique to *reduce* the number of features (*dimensions*) retained in a problem or data set through *linear* combinations or subsets of the original features. More specifically, we discuss linear *feature extraction*, or combining features via weighted sums of vectors to reduce redundant information and separate out noisy measurements.

In this dissertation, the specific form of LDR we employ is *principal component analysis* (PCA). For a given data matrix $\mathbf{X}$, PCA has many forms in many disciplines, but we specifically focus on three: the *spectral decomposition* (or *eigen-decomposition*) which operates on the Gramian of a mean-centered data matrix ($\mathbf{X}^T\mathbf{X}$), the *singular value decomposition* (or discrete Karhunen-Loeve transformation) of the data matrix $\mathbf{X}$, and the spectral decomposition of $\mathbf{X}$ or $\mathbf{X}^T\mathbf{X}$ with time-dependent observations. Some of the benefits of PCA are to increase the signal to noise ratio in noisy data, rotate features into orthogonal space to reduce feature correlation effects, reduce the number of parameters to estimate, and decrease computational and memory costs associated with model fitting. PCA can be employed when observations are recorded with many correlated features to reduce the number of features upon which statistical inference may be necessary. In this manuscript, we will discuss applications of PCA and its variants to poorly-posed classification, ill-posed classification, and statistical process monitoring.

In chapter two, we develop a heteroscedastic linear dimension reduction (*HLDR*) procedure for poorly-posed ($p < N < p^2/2$) heteroscedastic multivariate populations

that combines shrinkage estimation of precision matrices with the SVD of a rectangular data sufficiency matrix. Our new *HLDR* technique often preserves, nearly preserves, or even improves the original feature-space conditional error rate in a reduced feature space, especially in cases where training-sample sizes are small relative to the original-feature dimension. For certain population configurations, our new process outperforms competing *HLDR* methods—including the well-known Loog & Duin (2004), sliced inverse regression (Li, 1991), and sliced average variance estimation (Cook & Yin, 2001) *HLDR* techniques—by more efficiently incorporating discriminatory information contained in the disparate covariance matrices. We demonstrate the efficacy of our new *HLDR* approach when applied in conjunction with the sample quadratic classification procedure via three Monte Carlo simulations for three multivariate normal population configurations as well as for a real-data example.

In chapter three, we compare the computational costs of PCA to SVD for ill-posed or *high-dimensional* ($N \leq p$) classification scenarios. PCA is often employed to reduce the dimensionality of samples in contexts where $N \ll p$. However, the commonly-employed PCA algorithm can be thousands of times longer in computation and hundreds of times larger in space complexity than the SVD in high-dimensional contexts. We apply PCA and SVD to observations from seven real data sets before classification with either linear discriminant analysis (LDA) or quadratic discriminant analysis (QDA), and show that the discrimination behavior does not significantly change between PCA and SVD dimension reduction techniques. Further, we compare different eigenvector ordering schemes for each combination of dimension reduction method and discriminant function. We also offer remarks about the conditional error rates (CERs) of PCA and SVD when combined with LDA and QDA.

In chapter four, we discuss modifications to the PCA routine to account for time-dependent, nonlinear, and non-stationary observations drawn from multiple process states. In some engineered systems, additional designed features create a known

multi-state switching scheme among multiple autocorrelated, non-linear, and non-stationary processes, and incorporating this known information into PCA can significantly improve the ability to detect changes (faults) in the system. Adaptive-dynamic PCA (AD-PCA) has been shown in previous research to do as well as or better than nonlinear dimension reduction methods in flagging outliers in such environments. In simulations with one of three types of faults introduced, we compare accounting for the states versus ignoring them. We find that multi-state AD-PCA (MSAD-PCA) reduces the proportion of false alarms and reduces the average time to fault detection. Conversely, we also investigate the impact of assuming multiple states when only one exists, and find that as long as the number of observations is sufficient, this misspecification is not detrimental. We then apply MSAD-PCA to real-world data collected from a decentralized wastewater treatment (WWT) system during in control (IC) and out of control (OoC) conditions. MSAD-PCA flags a strong system fault earlier and more consistently than its single-state competitor. Furthermore, accounting for the physical switching system does not increase the number of false alarms when the process is IC and may ultimately assist with fault attribution.

In chapter five we describe the `R` software package `mvMonitoring` for implementing multi-state adaptive-dynamic PCA useful for multivariate statistical process monitoring. We describe briefly our motivation for creating this package and also describe four of the main functions within the package, including a detailed description of the synthetic data generation process used in chapter four. We further provide real and synthetic results from applying the `mvMonitoring` package to multivariate process monitoring data from a decentralized WWT plant in Golden, CO. We offer some brief concluding remarks in chapter six.

CHAPTER TWO

Regularized Linear Dimension Reduction for the Sample Quadratic Discriminant Function

ABSTRACT

We develop a new heteroscedastic linear dimension reduction (*HLDR*) procedure for heteroscedastic multivariate populations that combines shrinkage estimation of precision matrices with the concept of a theoretical *HLDR* result. Our new *HLDR* technique, which we refer to as the *SYS HLDR* method, often preserves, nearly preserves, or even improves the original feature-space conditional error rate in a reduced feature space, especially in cases where training-sample sizes are small relative to the original-feature dimension. For certain population configurations, the *SYS HLDR* process outperforms competing *HLDR* methods—including the well-known Loog and Duin, sliced inverse regression, and sliced average variance estimation *HLDR* techniques—by more efficiently incorporating discriminatory information contained in the disparate covariance matrices. We demonstrate the efficacy of our new *HLDR* approach when applied in conjunction with the sample quadratic classification procedure via three Monte Carlo simulations for three multivariate normal-population configurations as well as for a real-data example.

2.1 Introduction

If one assumes known class-conditional probability densities, then the optimal error rate of a statistical supervised classifier does not increase as the feature-space dimension p increases. However, this perspective is impractical because one must estimate all parameters from training data. Specifically, the performance of a statistical discriminant rule can be significantly degraded when the feature dimension, p, is large relative

to the class-specific training-sample sizes n_i, $i = 1, \ldots, m$, where m is the number of pre-identified classes. Generally, as p increases, the ratio $2n_i/p^2$, $i = 1, \ldots, m$, must be sufficiently large to avoid what Bellman (1961) first called the *curse of dimensionality*. In statistical classification, linear dimension reduction (*LDR*) is a widely practiced approach that avoids the curse of dimensionality by decreasing the number of feature dimensions from p to some reduced dimension $q < p$.

Here, we propose a new *HLDR* technique that stabilizes each class precision matrix by inducing bias via the shrinkage method of Haff (1979). When $p < n_i < p^2/2$, class precision matrix estimators can have large variability. To this end, we replace the unbiased sample precision estimator with Haff's biased shrinkage estimator, which greatly stabilizes the training-sample LDR matrix. As a result, our new *HLDR* process generally preserves or reduces the full-feature *conditional error rates* (*CER*s) in a reduced dimension for a statistical supervised classification rule for heteroscedastic populations.

For classification scenarios examined in this paper, our *HLDR* method outperforms four considered competitors in terms of classifier stability and smaller median *CER*s. We demonstrate the superior classification efficacy of our new *HLDR* approach compared to four current competing sufficient-statistic-based *HLDR* routines while considering three synthetic heteroscedastic multivariate normal data sets as well as a real data set. We remark that we do not compare methods of choosing an optimal reduced dimension $r < p$, but rather aim to discuss the behavior of the the *CER*s for $r < p$. For some methods of choosing an optimal dimension r, see Schott (1994) and Cook & Forzani (2008).

The *LDR* technique derived by Fisher (1936), sometimes known as the linear discriminant function (*LDF*), holds for two populations with equal covariance matrices. As mentioned above, *HLDR* methods extend the *LDF* to incorporate information in the differences in covariance matrices. Unfortunately, as Velilla (2008) states, "there

seems to be no universally accepted dimension reduction method" for heteroscedastic populations. However, several attempts have been recorded in the literature.

One such method has been proposed by Loog & Duin (2004), who use an eigenvector-based *HLDR* approach for greater than two populations utilizing the *Chernoff criterion* and have extended the well-known *LDA* technique using directed distance matrices. Partial least squares as an *HLDR* process has been proposed by Nguyen & Rocke (2002), Barker & Rayens (2003), and Boulesteix & Strimmer (2007). Additional *HLDR* methods have been proposed by Young, Marco, & Odell (1987a), Fischer & Thiele (1979), Velilla & Hernandez (2005), Velilla (2008), and Mahanta, Aghaei, Plataniotis, & Pasupathy (2012), as well as Fukunaga (1990), Kumar, Andreou, & Andreou (1996), Hennig (2004), Fan, Ke, Liu, & Xia (2015), and Ounpraseuth, Young, Van Zyl, Nelson, & Young (2015).

The remainder of the paper proceeds as follows. In Section 2.2.1, we establish notation and give a brief introduction to the sample quadratic classifier. We provide conditions under which the optimal error rate of the quadratic classification rule with all parameters known is preserved in a low-dimensional transformed feature space in Section 2.2.2. Also, we derive an *HLDR* projection matrix preserving the optimal error rate. In Section 2.2.3, we describe our new *SYS HLDR* routine—a robust and stabilizing modification wherein we apply shrinkage to the precision estimates. Further, in Section 2.3, we summarize four competing sufficient-statistic-based *HLDR* procedures. We describe our Monte Carlo simulation design in Section 2.4. In Section 2.5, we present the results of three Monte Carlo simulations under various multivariate normal parameter configurations with heteroscedastic covariance matrices using five competing *HLDR* processes, including our new *SYS HLDR* technique. Furthermore, in Section 2.6, we compare the efficacy of our new *HLDR* approach with four well-known *HLDR* methods using a non-parametric bootstrap simulation for a real data example. Finally, we give some concluding remarks in Section 2.7.

2.2 Main Results

2.2.1 The Quadratic Classifier

Let $\mathbb{R}_{m\times n}$ denote the set of all $m\times n$ matrices with entries in the field $\mathbb{R}$. Let $\mathbb{S}_p \subset \mathbb{R}_{p\times p}$ denote the set of $p \times p$ real symmetric matrices, and let $\mathbb{R}_p^{>} \subset \mathbb{S}_p$ denote the interior of the cone of $p \times p$ real symmetric positive-definite matrices. Also, for a matrix $\mathbf{A} \in \mathbb{R}_{m\times n}$, we let $\mathbf{A}^+ \in \mathbb{R}_{n\times m}$ represent the Moore-Penrose pseudo-inverse of $\mathbf{A}$, we let $\mathcal{C}(\mathbf{A})$ represent the column space of $\mathbf{A}$, and we let $\mathcal{N}(\mathbf{A})$ denote the null space of $\mathbf{A}$.

Suppose we have m distinct classes, $\Pi_1, \ldots, \Pi_m$. Let $\mathbf{x} \in \mathbb{R}_{p\times 1}$ represent a real observation with p measured features. Also, let $f_i(\mathbf{x})$ be a p-dimensional multivariate density corresponding to population Π_i such that essentially all classification information for random samples from $f_i(\mathbf{x})$ reside in the means $\boldsymbol{\mu}_i$ and covariances $\boldsymbol{\Sigma}_i$, where $i = 1, \ldots, m$. Some example distributions include the multivariate Student's t with non-heavy tails, symmetric multivariate Laplace, the multivariate logistic distribution, and other distributions within the multivariate elliptical family. For $m \geq 2$ multivariate populations or classes with *a priori* class membership α_i, the criterion function for the i^{th} class can be expressed as

$$d_i(\mathbf{x}) := \log|\boldsymbol{\Sigma}_i| - 2\log(\alpha_i) + (\mathbf{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i), \quad i = 1, \ldots, m. \tag{2.1}$$

Then, the well-established decision rule classifies the unlabeled observation $\mathbf{x}$ into the class Π_k if $d_k(\mathbf{x}) = D(\mathbf{x})$, where

$$D(\mathbf{x}) := \min\{d_i(\mathbf{x});\ i = 1, \ldots, m\}. \tag{2.2}$$

The classification rule (2.2) is known as the *quadratic discriminant function* (*QDF*) or the *quadratic classifier*. If all parameters in (2.1) are estimated from training data and $n_i > p$, then (2.1) becomes

$$\hat{d}_i(\mathbf{x}) := \log|\mathbf{S}_i| - 2\log(\alpha_i) + (\mathbf{x}-\bar{\mathbf{x}}_i)^T \mathbf{S}_i^{-1}(\mathbf{x}-\bar{\mathbf{x}}_i), \tag{2.3}$$

where $\bar{\mathbf{x}}_i$, $\mathbf{S}_i$, and $\mathbf{S}_i^{-1}$ are the maximum likelihood estimates of $\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$, and $\boldsymbol{\Sigma}_i^{-1}$, respectively, for class i, $i = 1, \ldots, m$.

2.2.2 A Linear Dimension Reduction Model Result

We now present a result that enables us to construct an *HLDR* transformation of full-dimensional feature data that preserves the optimal error rate associated with the full-dimension feature space. The following result is motivated by the discussion of Peters, Redner, & Decell (1978) on linear sufficient statistics for the differences in m heteroscedastic multivariate observations, specifically that $(\boldsymbol{\Sigma}_i\boldsymbol{\mu}_i - \boldsymbol{\Sigma}_1\boldsymbol{\mu}_1)$ and $(\boldsymbol{\Sigma}_i - \boldsymbol{\Sigma}_1)$, $i = 2, \ldots, m$, are sufficient to compare each population i to population 1. Group 1 is chosen as the reference group arbitrarily. Although Ounpraseuth et al. (2015) have provided similar results to the theorem below and the lemmas in the appendix, our proofs have the following major dissimilarities: this proof does not assume multivariate normality, the following proof uses the *QDF* shown in (2.2) for its criterion function instead of the ratio of multivariate normal densities, and all of our proofs are more concise. The *HLDR* method presented in the subsequent theorem constructs a relationship between classifying an observation in p-dimensional feature space and in a q-dimensional subspace, where $q < p$, when all parameters are known.

Theorem. *Suppose we have m multivariate populations with full classification information within the means $\boldsymbol{\mu}_i$ and covariance matrices $\boldsymbol{\Sigma}_i \in \mathbb{R}_p^{>}$, and let α_i denote a priori class membership, where $i = 1, \ldots, m$. Also, let*

$$\boldsymbol{M} := \left[\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 | \ldots | \boldsymbol{\Sigma}_m^{-1}\boldsymbol{\mu}_m - \boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 | \boldsymbol{\Sigma}_2 - \boldsymbol{\Sigma}_1 | \ldots | \boldsymbol{\Sigma}_m - \boldsymbol{\Sigma}_1\right]. \tag{2.4}$$

Let $\boldsymbol{M} = \boldsymbol{F}\boldsymbol{G} \in \mathbb{R}_{p\times s}$ be a full-rank decomposition of $\boldsymbol{M}$, where $\boldsymbol{F} \in \mathbb{R}_{p\times q}$, rank$(\boldsymbol{F}) = q < p$, and $s = (m-1)(p+1)$. Then, for an unlabeled observation vector $\boldsymbol{x} \in \mathbb{R}_{p\times 1}$, $D(\boldsymbol{x}) = D\left(\boldsymbol{F}^{+}\boldsymbol{x}\right)$, where $D(\boldsymbol{x})$ is defined in (2.2).

Proof. First let $\mathbf{F} \in \mathbb{R}_{p\times q}$ with rank$(\mathbf{F}) = q$. Now let $\mathbf{P}_F^{\perp} := \left(\mathbf{I} - \mathbf{F}\mathbf{F}^{+}\right)$, and $\mathbf{C} := \mathbf{R}\mathbf{P}_F^{\perp}$ where $\mathbf{R} \in \mathbb{R}_{(p-q)\times p}$ and rank$(\mathbf{R}) = p - q$. Using Lemmas A.1, A.2, and A.3,

with a matrix $\mathbf{A} = \left[\mathbf{F}^{+T}, \mathbf{C}^T\right]^T \in \mathbb{R}_p^>$, we have that

$$
\begin{aligned}
d_i(\mathbf{x}) &= d_i(\mathbf{A}\mathbf{x}) \\
&= -2\log(\alpha_i) + \log\left|\mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T\right| + \left[\mathbf{A}(\mathbf{x}-\boldsymbol{\mu}_i)\right]^T \left(\mathbf{A}\boldsymbol{\Sigma}_i\mathbf{A}^T\right)^{-1} \left[\mathbf{A}(\mathbf{x}-\boldsymbol{\mu}_i)\right] \\
&= -2\log(\alpha_i) + \log\begin{vmatrix} \mathbf{F}^+\boldsymbol{\Sigma}_i\mathbf{F}^{+T} & \mathbf{F}^+\boldsymbol{\Sigma}_i\mathbf{C}^T \\ \mathbf{C}\boldsymbol{\Sigma}_i\mathbf{F}^{+T} & \mathbf{C}\boldsymbol{\Sigma}_i\mathbf{C}^T \end{vmatrix} \\
&\quad + \begin{bmatrix} \mathbf{F}^+(\mathbf{x}-\boldsymbol{\mu}_i) \\ \mathbf{C}(\mathbf{x}-\boldsymbol{\mu}_i) \end{bmatrix}^T \begin{bmatrix} \mathbf{F}^+\boldsymbol{\Sigma}_i\mathbf{F}^{+T} & \mathbf{F}^+\boldsymbol{\Sigma}_i\mathbf{C}^T \\ \mathbf{C}\boldsymbol{\Sigma}_i\mathbf{F}^{+T} & \mathbf{C}\boldsymbol{\Sigma}_i\mathbf{C}^T \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{F}^+(\mathbf{x}-\boldsymbol{\mu}_i) \\ \mathbf{C}(\mathbf{x}-\boldsymbol{\mu}_i) \end{bmatrix} \\
&= -2\log(\alpha_i) + \log\begin{vmatrix} \mathbf{F}^+\boldsymbol{\Sigma}_i\mathbf{F}^{+T} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}\boldsymbol{\Sigma}_i\mathbf{C}^T \end{vmatrix} \\
&\quad + \begin{bmatrix} \mathbf{F}^+(\mathbf{x}-\boldsymbol{\mu}_i) \\ \mathbf{C}(\mathbf{x}-\boldsymbol{\mu}_i) \end{bmatrix}^T \begin{bmatrix} \mathbf{F}^+\boldsymbol{\Sigma}_i\mathbf{F}^{+T} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}\boldsymbol{\Sigma}_i\mathbf{C}^T \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{F}^+(\mathbf{x}-\boldsymbol{\mu}_i) \\ \mathbf{C}(\mathbf{x}-\boldsymbol{\mu}_i) \end{bmatrix} \\
&= -2\log(\alpha_i) + \log\left|\mathbf{F}^+\boldsymbol{\Sigma}_i\mathbf{F}^{+T}\right| + \log\left|\mathbf{C}\boldsymbol{\Sigma}_1\mathbf{C}^T\right| \\
&\quad + \left[\mathbf{F}^+(\mathbf{x}-\boldsymbol{\mu}_i)\right]^T \mathbf{F}^T\boldsymbol{\Sigma}_i^{-1}\mathbf{F} \left[\mathbf{F}^+(\mathbf{x}-\boldsymbol{\mu}_i)\right] \\
&\quad + \left[\mathbf{C}(\mathbf{x}-\boldsymbol{\mu}_1)\right]^T \left[\mathbf{C}\boldsymbol{\Sigma}_1\mathbf{C}^T\right]^{-1} \left[\mathbf{C}(\mathbf{x}-\boldsymbol{\mu}_1)\right] \\
&= -2\log(\alpha_i) + \log\left|\mathbf{F}^+\boldsymbol{\Sigma}_i\mathbf{F}^{+T}\right| \\
&\quad + \left[\mathbf{F}^+(\mathbf{x}-\boldsymbol{\mu}_i)\right]^T \mathbf{F}^T\boldsymbol{\Sigma}_i^{-1}\mathbf{F} \left[\mathbf{F}^+(\mathbf{x}-\boldsymbol{\mu}_i)\right] + c \\
&= d_i\left(\mathbf{F}^+\mathbf{x}\right) + c,
\end{aligned}
$$

where $c := \log\left|\mathbf{C}\boldsymbol{\Sigma}_1\mathbf{C}^T\right| + \left[\mathbf{C}(\mathbf{x}-\boldsymbol{\mu}_1)\right]^T \left[\mathbf{C}\boldsymbol{\Sigma}_1\mathbf{C}^T\right]^{-1} \left[\mathbf{C}(\mathbf{x}-\boldsymbol{\mu}_1)\right]$. Thus $d_i(\mathbf{x}) > d_j(\mathbf{x})$ if and only if $d_i\left(\mathbf{F}^+\mathbf{x}\right) > d_j\left(\mathbf{F}^+\mathbf{x}\right)$ for all $i, j = 1, \ldots, m;\ i \neq j$. Hence, $D(\mathbf{x}) = D\left(\mathbf{F}^+\mathbf{x}\right)$. □

In the theorem above, we have provided conditions under which the optimal error rate of the *QDF* is preserved in a low-dimensional transformed feature space. Moreover, we have derived an *HLDR* projection matrix preserving the optimal error rate for the reduced subspace of dimension $q = \text{rank}(\mathbf{M}) < p$ in the theoretical case

when all population parameters are known, and class means and precision matrices are unequal.

2.2.3 The SYS HLDR Method

First, let

$$\widehat{\mathbf{M}} := \left[\mathbf{S}_2^{-1}\bar{\mathbf{x}}_2 - \mathbf{S}_1^{-1}\bar{\mathbf{x}}_1 | \ldots | \mathbf{S}_m^{-1}\bar{\mathbf{x}}_m - \mathbf{S}_1^{-1}\bar{\mathbf{x}}_1 | \mathbf{S}_2 - \mathbf{S}_1 | \ldots | \mathbf{S}_m - \mathbf{S}_1\right] \tag{2.5}$$

be an estimator of (2.4). Because $\text{rank}(\widehat{\mathbf{M}}) = p$, one cannot directly obtain the *HLDR* matrix $\mathbf{F}^+ \in \mathbb{R}_{p\times q}$ that preserves the full-feature *CER*. Moreover, we may wish to obtain a still lower dimensional representation of the original data with dimension r, say, where $1 \leq r < q < p$. Thus, we seek to construct an r-dimensional *HLDR* matrix that preserves essentially all of the original p-dimensional *CER*.

Motivated by the theorem in Section 2.2.2, we propose a new *HLDR* procedure that utilizes a regularized estimate of the precision matrix in scenarios when n_i is small relative to p^2. For cases when $p < n_i < p^2/2$, the Haff shrinkage estimator of $\mathbf{S}_i^{-1}$ has smaller variance than the inverse of the unbiased or maximum likelihood estimators for $\mathbf{\Sigma}_i$ (Haff, 1979). This induced bias will stabilize the chosen estimator of $\mathbf{\Sigma}^{-1}$. Next, let

$$\widehat{\mathbf{M}}_{SYS} := \left[\widetilde{\mathbf{S}}_2^{-1}\bar{\mathbf{x}}_2 - \widetilde{\mathbf{S}}_1^{-1}\bar{\mathbf{x}}_1 | \ldots | \widetilde{\mathbf{S}}_m^{-1}\bar{\mathbf{x}}_m - \widetilde{\mathbf{S}}_1^{-1}\bar{\mathbf{x}}_1 | \mathbf{S}_2 - \mathbf{S}_1 | \ldots | \mathbf{S}_m - \mathbf{S}_1\right] \tag{2.6}$$

be our new estimator of (2.4). In this equation, we have used the Haff shrinkage estimator for $\mathbf{\Sigma}_i^{-1}$, defined as

$$\widetilde{\mathbf{S}}_i^{-1} := (1 - t(U_i))\,(n_i - p - 2)\,\mathbf{S}_i^{-1} + \frac{t(U_i)\,(pn_i - p - 2)}{\text{tr}\,(\mathbf{S}_i)}\mathbf{I}_p, \tag{2.7}$$

where

$$t(U_i) := \min\left\{\frac{4\,(p^2-1)}{(n_i - p - 2)\,p^2}, 1\right\} U_i^{1/p}$$

and

$$U_i := \frac{p\,|\mathbf{S}_i|^{1/p}}{\text{tr}\,(\mathbf{S}_i)},$$

for $i = 1, \ldots, m$. We remark that Peck, Jennings, & Young (1988) have shown that the *QDF* performs well when (2.7) estimates $\mathbf{\Sigma}_i^{-1}$ in (2.4). However, one could use other regularized estimators of $\mathbf{\Sigma}_i^{-1}$, $i = 1, \ldots, m$.

Our approach to obtaining an r-dimensional *HLDR* matrix, $1 \leq r < p$, is the *singular value decomposition* of Eckart & Young (1936). Let $\mathbf{A} = \mathbf{UDV}^T$ denote the singular value decomposition of some arbitrary matrix $\mathbf{A}$, and let $\mathbf{U}^{(r)}$ denote the r eigenvectors of $\mathbf{A}$ corresponding to the $r < p$ largest singular values of $\mathbf{A}$. Finally, let $\widehat{\mathbf{M}}_{SYS} = \mathbf{U}_{SYS}\mathbf{D}_{SYS}\mathbf{V}_{SYS}^T$ be the singular value decomposition of (2.6), and define $\widehat{\mathbf{F}}_{SYS}^{(r)} := \mathbf{U}_{SYS}^{(r)}$. Then $\left[\widehat{\mathbf{F}}_{SYS}^{(r)}\right]^T \in \mathbb{R}_{r \times p}$ is the *SYS HLDR* matrix for reducing the feature dimension from p to r, $1 \leq r < p$. In brief, the *SYS HLDR* approach, employing inverse-covariance matrix shrinkage, can yield relatively small *CER*s, reduced classifier variability, and is especially useful when n_i is small.

Concerning the selection of an appropriate reduced dimension r, we have mentioned other research completed on the optimal choice of the reduced dimension r in Section 2.1. While we respect the validity of such work, we firmly believe that for the task of supervised classification, the median conditional error rate is the most appropriate criterion for choosing a reduced dimension. We employ a bootstrap estimator of the error rate.

2.3 Four Competing HLDR Methods

In this section, we describe four sufficient-statistic-based *HLDR* methods that we contrast against the *SYS HLDR* method. For these four *HLDR* techniques, let $\mathbf{A} = \mathbf{UDV}^T$ denote the singular value decomposition of a matrix $\mathbf{A}$ and $\mathbf{U}^{(r)}$ be the r eigenvectors of $\mathbf{A}$ corresponding to the $r < p$ largest singular values of $\mathbf{A}$.

2.3.1 The SY Method

Let $\widehat{\mathbf{M}}_{SY}$ be defined as in (2.5), where $\mathbf{S}_i$ and $\bar{\mathbf{x}}_i$ are the maximum likelihood estimates of $\boldsymbol{\Sigma}_i$ and $\boldsymbol{\mu}_i$ in (2.4), respectively, provided $n_i > p$, $i = 1, \ldots, m$. While the *SY HLDR* routine from Ounpraseuth et al. (2015) uses the full-rank decomposition of the matrix $\mathbf{M}_{SY}$, we equivalently let $\widehat{\mathbf{M}}_{SY} = \mathbf{U}_{SY}\mathbf{D}_{SY}\mathbf{V}_{SY}^T$ be the singular value decomposition of $\widehat{\mathbf{M}}_{SY}$. Then, for $\widehat{\mathbf{F}}_{SY}^{(r)} := \mathbf{U}_{SY}^{(r)}$, the $r \times p$ *SY HLDR* matrix is $\left[\widehat{\mathbf{F}}_{SY}^{(r)}\right]^T$.

2.3.2 The LD HLDR Method

For the *HLDR* procedure proposed by Loog & Duin (2004), we wish to determine a matrix $\mathbf{A} \in \mathbb{R}_{r\times p}$ that maximizes the Chernoff distance

$$J_C(\mathbf{A}) := \sum_{i=1}^{m-1}\sum_{j=i+1}^{m} \alpha_i\alpha_j \operatorname{tr}\left[\left(\mathbf{A}\mathbf{S}_W\mathbf{A}^T\right)^{-1}\mathbf{A}\mathbf{S}_W^{1/2}\left(\tilde{\mathbf{S}}_{ij}^*\right)\mathbf{S}_W^{1/2}\mathbf{A}^T\right], \tag{2.8}$$

where

$$\begin{aligned}\tilde{\mathbf{S}}_{ij}^* := &\left(\mathbf{S}_{(i,j)}^*\right)^{-\frac{1}{2}}\mathbf{S}_W^{-1/2}\left(\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j\right)\left(\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j\right)^T\mathbf{S}_W^{-1/2}\left(\mathbf{S}_{(i,j)}^*\right)^{-\frac{1}{2}} \\ &+ \frac{1}{\pi_{(i|j)}\pi_{(j|i)}}\left[\log\left(\mathbf{S}_{(i,j)}^*\right) - \pi_{(i|j)}\log\left(\mathbf{S}_{(i,i)}^*\right) - \pi_{(j|i)}\log\left(\mathbf{S}_{(j,j)}^*\right)\right].\end{aligned} \tag{2.9}$$

Here $\mathbf{S}_{(a,b)}^* := \mathbf{S}_W^{-1/2}\left(\pi_{(a|b)}\mathbf{S}_a + \pi_{(b|a)}\mathbf{S}_b\right)\mathbf{S}_W^{-1/2}$, $\pi_{(a|b)} := \alpha_a/\left(\alpha_a + \alpha_b\right)$, α_i is the *a priori* probability for the i^{th} class such that $\sum_{i=1}^m \alpha_i = 1$, and $i = 1, \ldots, m$. Also,

$$\mathbf{S}_W := \sum_{i=1}^{m}\alpha_i\mathbf{S}_i \tag{2.10}$$

and

$$\mathbf{S}_B := \sum_{i=1}^{m}\left(\bar{\mathbf{x}}_i - \bar{\bar{\mathbf{x}}}\right)\left(\bar{\mathbf{x}}_i - \bar{\bar{\mathbf{x}}}\right)^T \tag{2.11}$$

are the sample within-class covariance and sample between-class scatter matrices, respectively. Moreover, $\bar{\mathbf{x}}_i$ is the sample mean vector for the i^{th} class and $\bar{\bar{\mathbf{x}}} = \sum_{i=1}^m \alpha_i\bar{\mathbf{x}}_i$ is the overall mean. Let

$$\widehat{\mathbf{M}}_{LD} := \sum_{i=1}^{m-1}\sum_{j=i+1}^{m}\alpha_i\alpha_j\mathbf{S}_W^{-1}\mathbf{S}_W^{1/2}\left[\tilde{\mathbf{S}}_{ij}^*\right]\mathbf{S}_W^{1/2},$$

where $\tilde{\mathbf{S}}_{ij}^{*}$ is given in (2.9), and let $\widehat{\mathbf{M}}_{LD} = \mathbf{U}_{LD}\mathbf{D}_{LD}\mathbf{V}_{LD}^{T}$ be the singular value decomposition of $\widehat{\mathbf{M}}_{LD}$. Then for $\widehat{\mathbf{F}}_{LD}^{(r)} := \mathbf{U}_{LD}^{(r)}$, the $r \times p$ *LD HLDR* matrix is $\left[\widehat{\mathbf{F}}_{LD}^{(r)}\right]^{T}$.

2.3.3 The SIR HLDR Method

We now describe the sliced inverse regression (*SIR*) *HLDR* approach, originally proposed by Li (1991). For the case where parameters are unknown, we use the *SIR* matrix

$$\widehat{\mathbf{M}}_{SIR} := \widehat{\mathbf{\Gamma}}^{-1/2}\mathbf{S}_{B}\widehat{\mathbf{\Gamma}}^{-1/2}, \tag{2.12}$$

where $\widehat{\mathbf{\Gamma}} := \mathbf{S}_{B} + \mathbf{S}_{W}$ is the estimated marginal covariance matrix of $\mathbf{X}$, and $\mathbf{S}_{W}$ and $\mathbf{S}_{B}$ are given in (2.10) and (2.11), respectively. Let $\widehat{\mathbf{M}}_{SIR} = \mathbf{U}_{SIR}\mathbf{D}_{SIR}\mathbf{V}_{SIR}^{T}$ be the singular value decomposition of (2.12), and let $\widehat{\mathbf{F}}_{SIR} := \widehat{\mathbf{\Gamma}}^{-1/2}\mathbf{U}_{SIR}^{(r)}$. Then, the $r \times p$ *SIR HLDR* matrix is $\left[\widehat{\mathbf{F}}_{SIR}^{(r)}\right]^{T}$.

2.3.4 The SAVE HLDR Method

Lastly, we describe the sliced average variance estimation (*SAVE*) routine proposed by Cook & Weisberg (1991) and discussed in Cook & Yin (2001). For the case where parameters are unknown, we use a form of *SAVE* given in Velilla (2008), which is

$$\widehat{\mathbf{M}}_{SV} := \left(\mathbf{I} - \widehat{\mathbf{\Gamma}}^{-1/2}\mathbf{S}_{B}\widehat{\mathbf{\Gamma}}^{-1/2}\right)^{2} + \widehat{\mathbf{\Gamma}}^{-1/2}\mathbf{S}_{\widehat{\mathbf{\Gamma}}}\widehat{\mathbf{\Gamma}}^{-1/2}. \tag{2.13}$$

In (2.13), $\mathbf{S}_{\widehat{\mathbf{\Gamma}}} := \frac{1}{m}\sum_{i=1}^{m}\left(\mathbf{S}_{i} - \mathbf{S}_{W}\right)\widehat{\mathbf{\Gamma}}^{-1}\left(\mathbf{S}_{i} - \mathbf{S}_{W}\right)$, and $\mathbf{S}_{W}$ and $\mathbf{S}_{B}$ given in (2.10) and (2.11), respectively. Let $\widehat{\mathbf{M}}_{SV} = \mathbf{U}_{SV}\mathbf{D}_{SV}\mathbf{V}_{SV}^{T}$ be the singular value decomposition of (2.13). Then, $\left[\widehat{\mathbf{F}}_{SV}^{(r)}\right]^{T} := \left[\widehat{\mathbf{\Gamma}}^{-1/2}\mathbf{U}_{SV}^{(r)}\right]^{T}$ is the $r \times p$ *SAVE HLDR* matrix.

2.4 Monte Carlo Simulation Description

In this section, we describe the simulation design we used to contrast our new *SYS HLDR* routine derived in Section 2.2.3 against the *SY*, *LD*, *SIR*, and *SAVE HLDR* approaches of Section 2.3. Specifically, due to the skewness of the *CER*s, we evaluate

the classification efficacy in terms of the median *CER* (*MCER*). We reiterate that we examine the *conditional* error rates and their variability, not the behavior of the *expected* error rates. We use Monte Carlo simulations for three different configurations of multivariate normal populations with $p = 10$.

First, let $MCER_r(\cdot)$ denote the estimated *MCER*s for each of the chosen procedures with reduced dimension $r < p$. We performed the following steps for each parameter configuration:

Step 1. Generate 5,000 observations from the designated 10-dimensional multivariate normal distribution per each of the $m = 3$ classes.

Step 2. Partition the $(5000m)$ observations into training and test data sets.

i. Simulate poorly-posed covariance matrix scenarios taking $n = 15$ training observations per class.

ii. Hold the remaining 4,985 observations per class aside for testing.

Step 3. Repeat the following steps for each *HLDR* technique:

i. Project the training and test data from p to r dimensions, where r is fixed.

ii. Construct the classifier.

iii. Classify the test observations into one of the m distinct classes.

iv. Record the *CER*.

Step 4. Repeat Steps 1–3 2,500 times.

Step 5. Increase n_i from 15 to 30 then 60, and repeat Steps 1–4.

In Step 3, we apply the five competing *HLDR* matrices—$\widehat{\mathbf{F}}_{SYS}^{(r)}$, $\widehat{\mathbf{F}}_{SY}^{(r)}$, $\widehat{\mathbf{F}}_{LD}^{(r)}$, $\widehat{\mathbf{F}}_{SIR}^{(r)}$, and $\widehat{\mathbf{F}}_{SV}^{(r)}$—to project the data from p to r dimensions, $r = 1, \ldots, 9$. Then, we calculate the *CER*s for the five reduced-dimensional *QDF*s. Specifically, in Step 3(iii), the test data are assigned to either Π_1, Π_2, or Π_3 using *QDF* classifier (2.3). In Step 4, we calculate the *MCER* for each *HLDR* process within each reduced dimension r by

taking the median of the 2,500 estimated *CER*s from Step 3. In Step 5, we vary the training sample sizes on the five *HLDR* methods using $n_i = 1.5p, 3p, 6p$, for $i = 1, 2, 3$. Each training sample size is less than the 65 parameters to be estimated per class.

The simulation parameter configurations are as follows.

(1) *Configuration 1:* $m = 3$ *with diverse population covariance matrices.* For Configuration 1, we have three distinct multivariate normal populations: $N_p(\boldsymbol{\mu}_{1i}, \boldsymbol{\Sigma}_i)$, for $i = 1, 2, 3$, and $p = 10$; $\boldsymbol{\mu}_{1i}$ and $\boldsymbol{\Sigma}_i$ are given in 2.0.1 and 2.0.2, respectively. For this configuration of parameters, $\text{rank}(\mathbf{M}) = 2$, and we display results for $n_i = 30$.

(2) *Configuration 2:* $m = 3$ *with two similar covariance matrices and one spherical covariance matrix.* In the second Monte Carlo simulation, we examined the results of a configuration with three multivariate normal populations, $N_p(\boldsymbol{\mu}_{2i}, \boldsymbol{\Sigma}_i)$, for $i = 1, 2, 3$ and $p = 10$; $\boldsymbol{\mu}_{2i}$ and $\boldsymbol{\Sigma}_i$ are given in 2.0.1 and 2.0.3, respectively. For this configuration of parameters, $\text{rank}(\mathbf{M}) = 2$, and we display results for $n_i = 15$.

(3) *Configuration 3:* $m = 3$ *with diffuse similar covariance matrices except for the first two dimensions.* Here we have three multivariate normal populations: $N_p(\boldsymbol{\mu}_{3i}, \boldsymbol{\Sigma}_i)$, for $i = 1, 2, 3$ and $p = 10$; $\boldsymbol{\mu}_{3i}$ and $\boldsymbol{\Sigma}_i$ are given in 2.0.1 and 2.0.4, respectively. Also, $\text{rank}(\mathbf{M}) = 4$, and we display results for $n_i = 60$.

2.5 Monte Carlo Simulation Results

We present the results of our Monte Carlo simulations. We performed three Monte Carlo simulations using the statistical computing resource `R`, versions 3.2.0–3.3.1. The code will be released as a package separately. In Figures 2.1–2.3, we display boxplots of the *CER*s of the five competing *HLDR* techniques for the various population configurations and values of n_i and r, where $i = 1, \ldots, m$ and $r = 1, \ldots, 7$. For $r > 7$, all five *HLDR* procedures behaved similarly. The estimated standard error of all *MCER*s

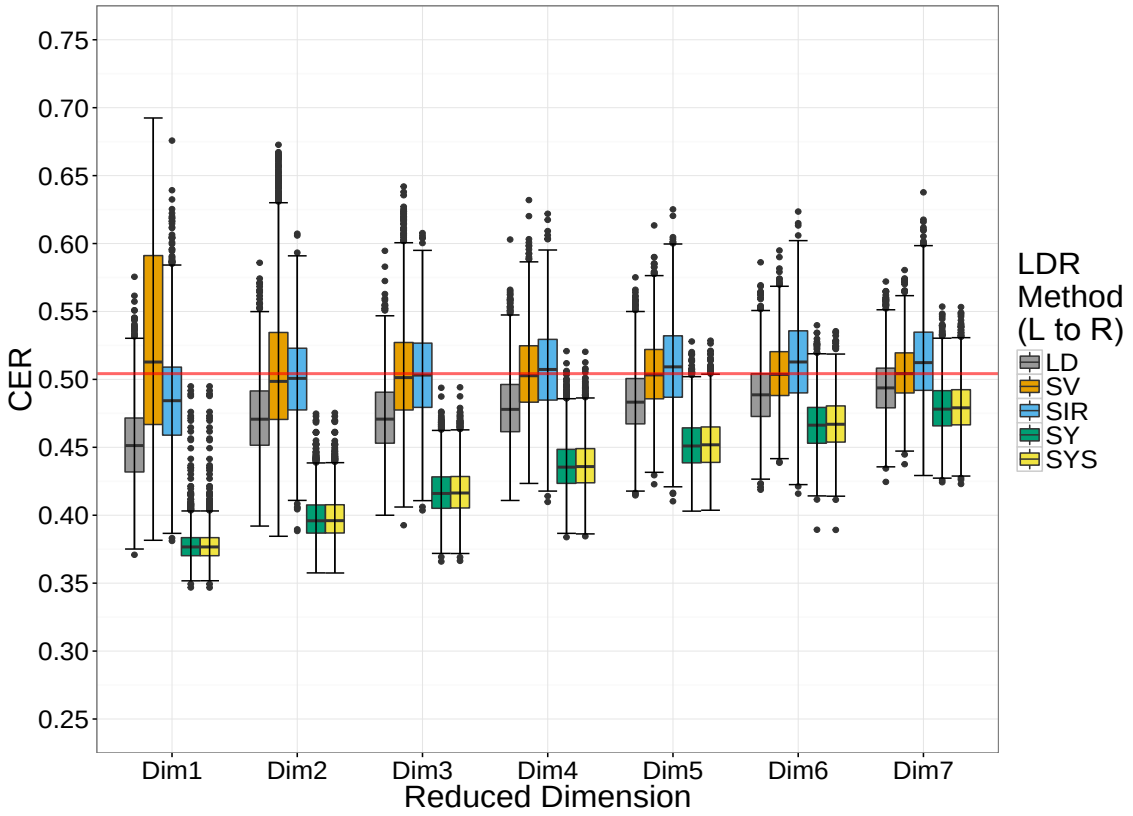


Figure 2.1: Conditional Error Rate Plots for Simulation Configuration 1 with $n_i = 30$.

was less than 0.008. Unless otherwise stated, *CER* values are rounded to the nearest 0.005.

2.5.1 Configuration 1: $m = 3$ with diverse population covariance matrices

In Figure 2.1, the vertical box plots represent the summary of the *CER*s for the 2,500 replications of each approach within each dimension. The *HLDR* processes are – from left to right within each dimension – *LD* from Loog & Duin (2004), *SAVE* from Velilla (2008), *SIR* from Li (1991), *SY* from Ounpraseuth et al. (2015), and *SYS* from Section 2.2.3.

The line at $CER = 0.505$ in Figure 2.1 is the *MCER* for the original 10-dimensional data. In Figure 2.1, we see that as r decreased, the *MCER* decreased for the *LD*, *SY*, and *SYS HLDR* techniques. Moreover, the variability of the *SY*

and *SYS HLDR* routines decreased as r decreased, yielding more stable discriminant functions. However, the *MCER* for the *SIR* and *SAVE HLDR* procedures remained relatively constant, and classifier variability increased for the *SAVE HLDR* procedure. In scenarios where the covariance matrix estimates are highly disparate, the *SIR* and *SAVE* methods suffered because they depended upon a pooled covariance matrix estimate. However, in these same situations, the *SY* and *SYS* approaches appeared to benefit.

Because $\text{rank}(\mathbf{M}) = 2$, the theoretical *CER*-preserving reduced-dimension lower bound is at $q = 2$, which, we remark, is not the *practical* lower bound. The practical lower bound, however, yielding the lowest *MCER* of the five methods, occurred at $r = 1$. By the principle of parsimony, describing our data using two parameter estimates per class instead of five parameter estimates further decreased the *MCER* for both the *SY* and *SYS HLDR* processes. By reducing the full dimensional data from $p = 10$ to the reduced dimension $r = 1$, we also greatly stabilized our classifier via the *SYS HLDR* technique. Specifically, we decreased the *MCER* from 0.505 to 0.375 with the *SY* and *SYS HLDR* routines, which resulted in an *MCER* improvement of approximately 0.130.

Overall, the *CER* behavior of each *HLDR* procedure was similar for each $n_i = 15, 30, 60$. From $p = 10$ to $r = 1$, *CER* boxplots for the *LD*, *SY*, and *SYS HLDR* approaches monotonically decreased, even past the theoretical *CER*-preserving reduced-dimension lower bound at $q = 2$. For $n_i = 15$, the five-method minimum *MCER* of 0.390 was achieved concurrently by the *SY* and *SYS HLDR* methods at the reduced dimension $r = 1$. The original 10-dimensional data yielded an *MCER* of 0.565, resulting in an *MCER* decrease of 0.175.

Furthermore, for $n_i = 60$, the five-method minimum *MCER* was 0.370, achieved concurrently by the *SY* and *SYS HLDR* approaches at the reduced dimension $r = 1$. The *MCER* for the original 10-dimensional data was $CER = 0.455$, yielding an *MCER*

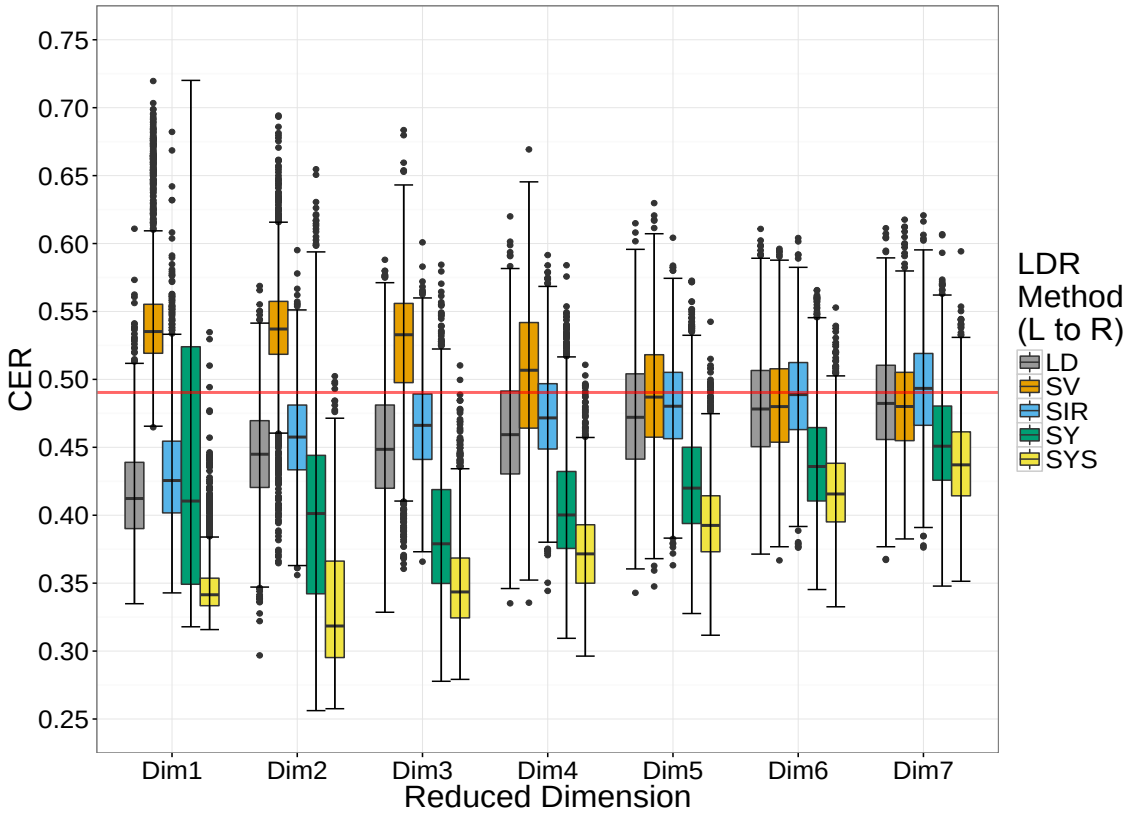


Figure 2.2: Conditional Error Rate Plots for Simulation Configuration 2 with $n_i = 15$.

decrease of 0.085. Further, increasing the training sample size from 15 to 30 or 60 per class reduced all the *MCER*s and the *CER* variability for each of the five competing *HLDR* techniques.

2.5.2 Configuration 2: $m = 3$ with two similar covariance matrices and one spherical covariance matrix

In Figure 2.2, the line at $CER = 0.490$ is the *MCER* for the original 10-dimensional data. In Figure 2.2 we see that as r decreased, the *MCER* decreased for the *LD*, *SIR*, *SY*, and *SYS HLDR* procedures. However, the variability of the *SY HLDR* method increased dramatically as r decreased, yielding a less stable quadratic discriminant function. Moreover, the *MCER* for the *SAVE HLDR* procedure increased as r decreased, although the variability of this classifier decreased. We remark that

the variability of the *LD*, *SIR*, and *SYS HLDR* methods remained roughly stable as r decreased.

Because $\text{rank}(\mathbf{M}) = 2$, the theoretical *CER*-preserving reduced-dimension lower bound occurred at $q = 2$. In this instance, the practical and theoretical error-rate preserving lower bounds coincided. However, for $1 \leq r \leq 4$, the *SAVE HLDR*-based classifier produced greater *MCER*s than when $p = 10$. By reducing the full dimensional data from $p = 10$ to $r = q = 2$, we decreased the *MCER* from 0.490 to 0.320 by using the *SYS HLDR* routine, which resulted in an *MCER* improvement of approximately 0.170.

While varying n_i, the *LD HLDR CER* behavior of the procedure changed from monotonically decreasing for $n_i = 15$, to slightly concave for $n_i = 30$ with a minimum *MCER* at $r = 3$, to concave with a larger radius of curvature for $n_i = 60$ with a minimum at $r = 2$. The *CER* behavior of the *SAVE* and *SYS HLDR* approaches was similar to the *LD HLDR* process. The *CER* behavior of the *SIR HLDR* method became more flat as n_i increased. Furthermore, the *MCER* behavior of the *SY HLDR* routine was similar to the *MCER* behavior as shown in Figure 2.2, but the variability decreased markedly for $n_i = 30, 60$.

For $n_i = 30$, the five-method minimum *MCER* of 0.270 was achieved by the *SY HLDR* approach at the reduced dimension $r = 2$. The *MCER* for the original 10-dimensional data was at $CER = 0.395$, yielding an *MCER* decrease of 0.125. Furthermore, for $n_i = 60$, the five-method minimum *MCER* was 0.260, achieved by the *SYS HLDR* procedure at reduced dimension $r = 2$. The *MCER* for the original 10-dimensional data was at $CER = 0.335$, yielding an *MCER* decrease of 0.075. Finally, increasing the training-sample size from 15 to 30 or 60 per class reduced all the *MCER*s and the *CER* variability for all five competing *HLDR* techniques.

2.5.3 *Configuration 3: $m = 3$ with diffuse similar covariance matrices except for the first two dimensions*

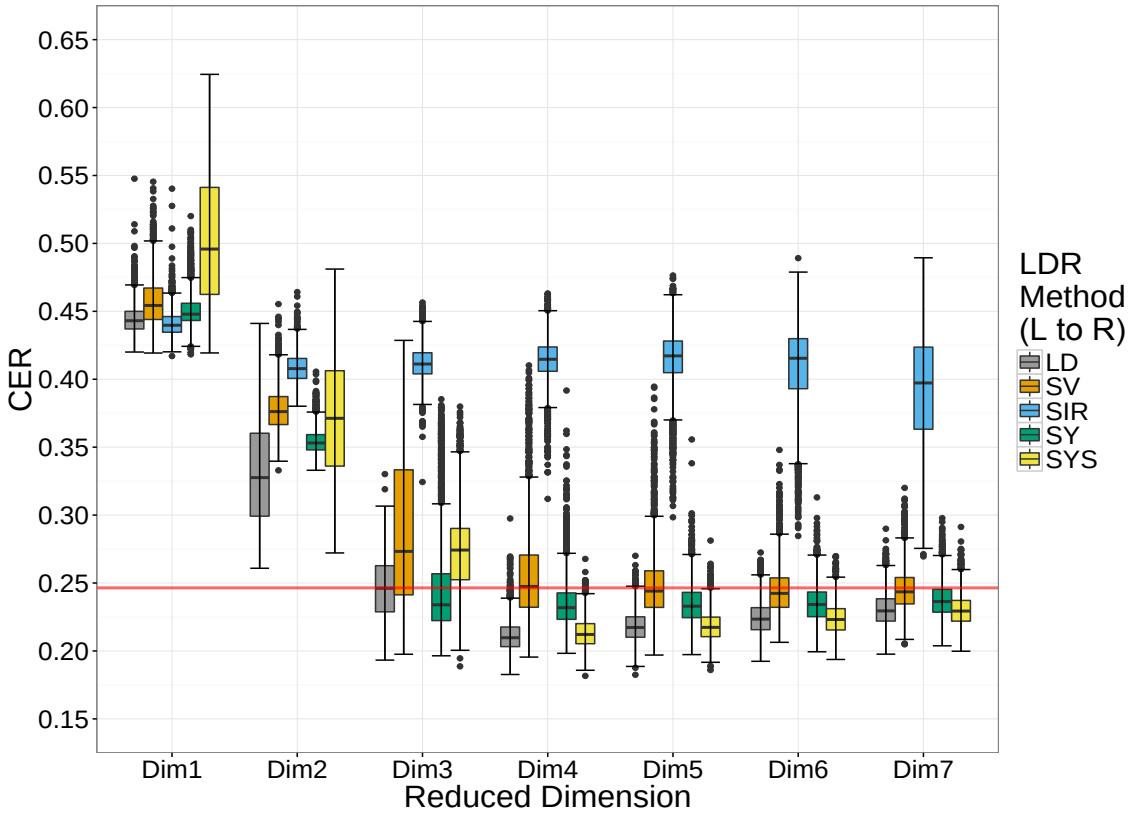


Figure 2.3: Conditional Error Rate Plots for Simulation Configuration 3 with $n_i = 60$.

In Figure 2.3, the line at $CER = 0.245$ is the *MCER* for the original 10-dimensional data. We see in Figure 2.3 that as r decreased, the *MCER* slightly decreased for the *LD*, *SY*, and *SYS HLDR* techniques until $r = q = 4$, then increased for $1 \leq r \leq 3$. The variability of the *LD*, *SY*, and *SYS HLDR* processes remained roughly constant until $r = 4$, while the variability of the *SIR HLDR* process actually decreased as r decreased. However, this increase in classifier stability means little because the *MCER* for the *SIR HLDR* approach increased while r decreased. Moreover, for $4 \leq r \leq 10$, the *SAVE HLDR* method performed only nominally better than classification in the original 10-dimensional space, and the *MCER* for the *SAVE HLDR* method was greater than 0.240 for all $r < p = 10$.

We attribute this poor showing of the classifiers in $r = 1, 2, 3$ to rank($\mathbf{M}$). Because rank($\mathbf{M}$) $= 4$, $q = 4$ is the theoretical *CER*-preserving reduced-dimension lower bound. For this configuration, the practical and theoretical lower bounds coincided. Only the *LD*, *SY*, and *SYS* procedures produced any practical improvement in the *MCER*, and for $r = 4$ the *SYS HLDR* approach showed fewer instances with *CER* > 0.245 than the *LD HLDR* routine, while remaining competitive with the *CER* of the *LD HLDR* routine. We remark that $MCER_4(LD) = MCER_4(SYS) = 0.210$. By reducing the full dimensional data from $p = 10$ to $r = 4$, we decreased the *MCER* from 0.245 to 0.210 with the *LD* and *SYS HLDR* processes, which resulted in an *MCER* improvement of approximately 0.035.

Overall, the *CER* behavior of each *HLDR* procedure was similar for each $n_i = 15, 30, 60$. As n_i increased, we observed a more pronounced increase in the *CER*s for $1 \leq r \leq 3$. From $p = 10$ to $r = 4$, *CER* boxplots for the *LD*, *SY*, and *SYS HLDR* approaches monotonically decreased. The *CER* boxplots for the *SIR HLDR* routine were relatively flat for each value of n_i selected. However, for $n_i = 15,\ 30$, the width of the *CER* boxplots for the *SAVE HLDR* method monotonically increased as r decreased.

For $n_i = 15$, the five-method minimum *MCER* of 0.300 was achieved by the *SYS HLDR* technique at the reduced dimension $r = 3$. The *MCER* for the original 10-dimensional data was 0.465, yielding an *MCER* decrease of 0.165. Furthermore, for $n_i = 30$, the five-method minimum *MCER* was 0.25, achieved by the *SYS HLDR* process at $r = 4$. The *MCER* for the original 10-dimensional data was 0.325, yielding an *MCER* decrease of 0.075 for the *SYS HLDR* method. Further, increasing the training-sample size from 15 to 30 or 60 reduced all of the *MCER*s and the variability of the *CER*s for each *HLDR* technique, for $4 \leq r \leq 10$.

2.6 A Real-Data Example

The radar data of Sigillito, Wing, Hutton, & Baker (1989) was collected by a signal collection system in Goose Bay, Labrador. According to the University of California–Irvine Machine Learning Repository summary, this system consisted of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. "Good" radar returns showed evidence of some type of structure in the ionosphere. "Bad" returns did not; their signals passed through the ionosphere.

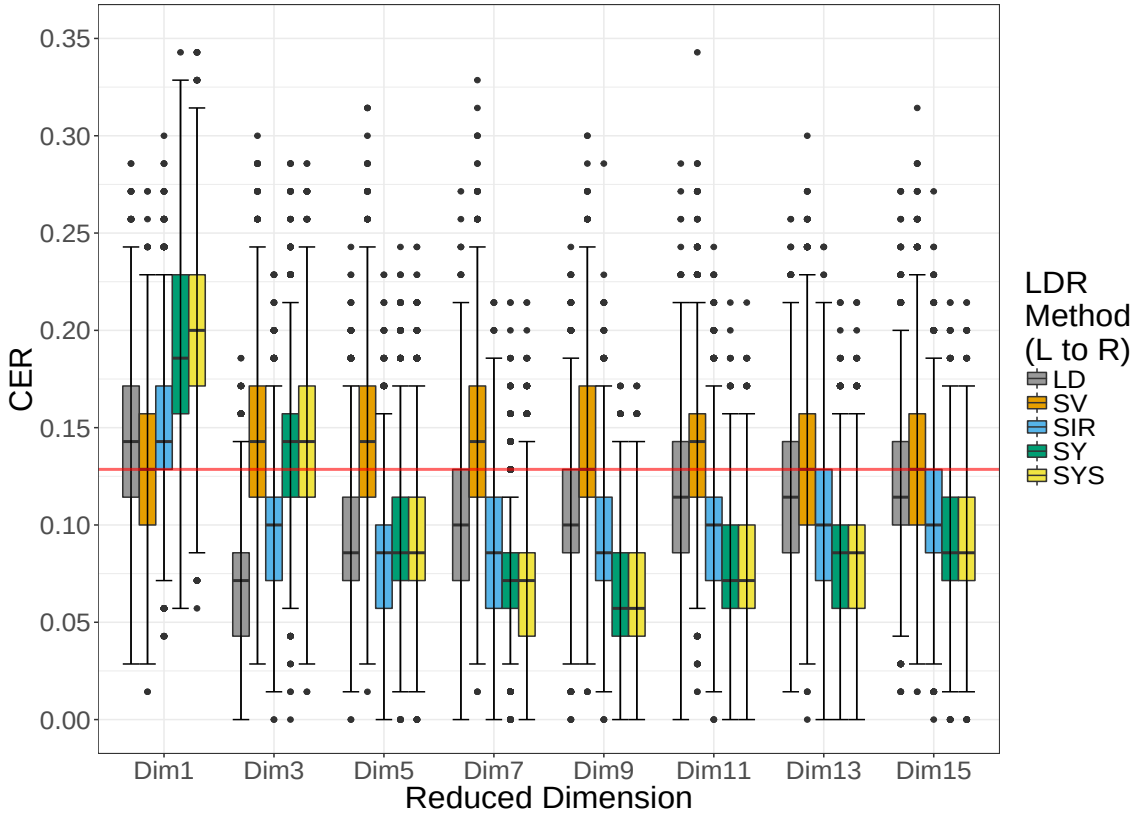


Figure 2.4: Conditional Error Rate Plots for Ionosphere Data

Researchers processed signals using an autocorrelation function whose arguments were the time of a pulse and the pulse number. The Goose Bay system had 17 pulse numbers. Because electromagnetic signals are complex-valued, instances in this

database are described by two attributes per pulse number. In each pair, the first attribute corresponds to the real component while the second represents the imaginary. Because measures "V1" and "V2" were binary (all feature names and descriptions are proprietary), we removed these two original features.

After preliminary data cleaning, the data had 351 observations with 32 features. The "Good" category had 225 observations while the "Bad" category had only 126 observations. Also, the covariance matrices were substantially unequal, as confirmed by the p-value of a likelihood ratio test statistic. The $\widehat{\mathbf{M}}$ matrix was full rank (as to be expected with real data), but the sum of the smallest 23 weighted singular values of $\widehat{\mathbf{M}}$ was less than 0.10, thus suggesting that the SY and SYS dimension reduction techniques should improve the classification accuracy.

We implemented a non-parametric bootstrap simulation as follows: we varied the training data percentage from 40% to 90% over 10 percentage point intervals. We chose an 80% training proportion to minimize cross-validated *MCER*, and we randomly selected 80% (rounded down) of the observations for the training data set. That is, we randomly selected 180 "Good" and 100 "Bad" observations from the 351 observations to form our training-sample data set and then followed the substeps in Step 3 of Section 2.4 to record a single *CER* for each of the five competing *HLDR* methods within each of the r reduced dimensions, $r = 1, \ldots, 31$. We repeated this process 5,000 times.

In Figure 2.4, the line at $CER = 0.13$ is the *MCER* for the original 32-dimensional data. For brevity, we display eight of the 31 reduced dimensions. We chose this subset of reduced dimensions ($7 \leq r \leq 14$) which surrounded $r = 8, 9$ where the effective global minimum *MCER*s occurred.

We now describe the behavior of the five competing *HLDR* approaches. For the *LD HLDR* technique, the *MCER* remained roughly constant from $p = 32$ to $r = 19$ and then slowly decreased from $r = 18$ to its minimum of 0.071 at $r = 2$. For the

SAVE HLDR procedure, the *MCER* remained roughly constant at 0.13 from $p = 32$ to $r = 13$, then increased starting at $r = 12$ before slowly returning to 0.13. For $r < 32$, the *MCER*(*SAVE*) was greater than 0.13. For the *SIR HLDR* routine, the *MCER* remained roughly constant from $p = 32$ to $r = 27$, then decreased from $r = 26$ to its minimum of 0.086 at $r = 6$. As r decreased from 32 dimensions to 9 dimensions, the *SY* and *SYS HLDR MCER*s slowly decreased. The *SY* and *SYS HLDR* methods achieved the lowest *MCER* of all *HLDR* methods—0.056 for $r = 8, 9$.

Because $\min\{n_i\} = 100 > 10 * r,\ i = 1, 2$, the *SY* and *SYS HLDR* routines performed almost equivalently. Thus, we used the mean *CER* for further comparison. For $r = 8$, the mean *CER*s for the *SYS* and *SY HLDR* approaches were slightly smaller than the same means for $r = 9$. Also for $r = 8$, the mean *CER* for the *SYS HLDR* procedure was slightly smaller than the mean *CER* for the *SY HLDR* procedure. By reducing the full-dimensional data from $p = 32$ to $r = 8$, thus reducing the number of parameters to estimate from 1120 to 88, we reduced the *MCER* from 0.13 to 0.056 by using the *SY* and *SYS HLDR* processes—an *MCER* improvement of approximately 0.074.

2.7 Discussion

In summary, we have derived a new *HLDR* technique employing inverse-covariance matrix shrinkage that can yield relatively small *CER*s and reduced classifier variability. Furthermore, using Monte Carlo simulations under various heteroscedastic multivariate normal parameter configurations, we have corroborated our claim that our new *SYS HLDR* method with shrinkage estimators of class precision matrices can decrease the *CER*s, especially in scenarios when $n_i/p^2 < 0.5$.

In scenarios when the eigenstructure of $\mathbf{M}$ from (2.4) is somewhat spiked and monetary or temporal limitations constrain researchers from collecting sufficient training data, the *SYS HLDR* procedure generally outperforms the four chosen *HLDR*

competitors in terms of classifier stability and reduced median *CER*s. Furthermore, considering the principle of parsimony, we have exhibited scenarios wherein reducing the dimension beyond the theoretical lower bound decreased the *MCER*. We do not claim that our new *SYS HLDR* technique has uniformly lower *MCER*s. However, we have shown that the *SYS HLDR* approach offers superior classification ability over four current competing *HLDR* approaches for three simulated-data population configurations and is non-inferior for the real ionosphere data set.

Acknowledgments

We acknowledge Ben J. Barnard for his help in developing an associated `R` package, Whitney V. Burrow for her coding assistance, and the Machine Learning Repository of Bache & Lichman (2013), hosted by the University of California—Irvine, for providing real machine learning data. This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

CHAPTER THREE

A Comparison of Principal Component Analysis and the Singular Value Decomposition as Linear Dimension Reduction Methods

ABSTRACT

Principal component analysis (PCA) and the *singular value decomposition* (SVD) are often employed to reduce the dimensionality of sampled data in high-dimensional ($N \ll p$) contexts. However, the commonly-employed PCA algorithm can take thousands of times longer in computation and be hundreds of times larger in space complexity than the SVD in high-dimensional contexts. We apply PCA and the SVD to observations from six real high-dimensional microarray data sets combined with classification for both linear discriminant analysis or quadratic discriminant analysis. We show that the discrimination behavior does not significantly change between the PCA and the SVD dimension-reduction techniques over the considered supervised classifiers. Further, we compare different eigenvector ordering schemes for each combination of dimension reduction method and discriminant function. We also offer remarks and observations concerning the estimated expected error rates of PCA and SVD when combined with linear discriminant analysis and quadratic discriminant analysis.

3.1 Introduction

When researchers have recorded data with many features—that is, p is large relative to N—they often use *linear dimension reduction* (LDR) techniques to mitigate what Bellman (1961) has described as the *curse of dimensionality*. One commonly accepted LDR technique is *principal component analysis* (PCA), in which one decomposes a sample covariance matrix into its eigenvectors and their associated eigenvalues (Pearson, 1901). Similarly, one can use the *singular value decomposition* (SVD) to de-

compose the data matrix into singular vectors and their associated singular values (Eckart & Young, 1936).

Consider this motivating example. One may desire to apply supervised classification to the stem cell factors in the glioma data set of Sun, Hui, Su, Vortmeyer, Kotliarov, Pastorino, Passaniti, Menon, Walling, Bailey, Rosenblum, Mikkelsen, & Fine (2006). This data set has 54,613 microarray feature values with a sample size of 180 subjects, so that $N \ll p$. Because PCA is $O(p^3)$ in computation time and $O(p^2)$ in space complexity, calculating a sample covariance matrix for this data is a non-trivial task, and calculating the eigen-decomposition of this covariance matrix is not tractable because of memory overflow. In contrast, because the SVD is $O(Np^2)$ in computation time and $O(Np)$ in space complexity, performing the SVD on this data matrix is tractable. However, despite its potentially exorbitant computational costs in high-dimensional contexts, PCA remains the default option to reduce the observation dimension, as one can see in Raychaudhuri, Stuart, & Altman (2000).

In this paper, for the case when the number of observations is small relative to a large number of dimensions ($N \ll p$), we calculate benchmark results comparing the SVD and PCA when combined with supervised classification via linear discriminant analysis (LDA) or quadratic discriminant analysis (QDA). Further, we present supervised classification results with the linear and quadratic discriminant functions to verify that LDR with the SVD yields non-inferior statistical classification results to LDR with PCA. Moreover, we discuss the condition numbers of PCA and the SVD to prove that the SVD has a smaller loss of numerical precision than PCA. We also provide a theoretical justification for replacing PCA with the SVD. Additionally, we describe three eigenvector ordering schemes and compare their statistical classification results with singular or eigenvalue ordering of eigenvectors.

Instead of directly employing the SVD to reduce the feature space in high-dimensional scenarios, many researchers attempt to augment PCA by using feature

pre-screening, gene identification, and other preliminary dimension reduction techniques. Satagopan & Panageas (2003) have provided a summary of such techniques. One set of pre-screening methods is to calculate gene-specific t-tests or gene-specific Wilcoxon Mann-Whitney tests, and then "screen" features by selecting the first few hundred genes after ordering these genes by their statistic values. However, Wang & Gehan (2005) remark that this test-statistic ranking method fails to account for correlation among the screened features. Resampling techniques, such as those discussed in Westfall & Young (1993), are an attempt to correct the corresponding p-values of these statistics for the inherent correlation of the genes. Further, the t-statistic and similar feature-ranking approaches fail when observations are drawn from more than two populations. Pavlidis (2003) has also discussed ranking the genes by decreasing test statistic scores from multiple one-way ANOVA F-statistics on each feature. Additionally, many other methods can be found in the literature.

Unfortunately, Satagopan & Panageas (2003) have also remarked that when these gene identification techniques are performed, often "every sample ... is utilized in these analyses" (p. 490). This implies that observations from the testing and even the validation sets are indirectly used in the model selection process. Oftentimes, overfitting the classification model (by feature filtering with both the training and test data sets) can cause severe bias, as discussed in Wang, Yin, Pei, Yu, & Yu (2006).

In this paper, we show that the SVD enables one to reduce the feature dimension effectively using observations reserved for training alone, thereby reducing the bias inherent in statistical gene selection, all while being computationally superior to PCA. In Section 3.2, we establish notation for high-dimensional observations, give definitions for the linear and quadratic discriminant functions, and define our PCA- and SVD-based dimension reduction methods. In Section 3.3, we describe three alternatives to ordering the eigenvectors by the eigenvalue magnitude, including canonical variates

ordering. In Section 3.4, we discuss the non-parametric boostrap simulation design. In Section 3.5, we discuss the condition numbers of PCA and the SVD and also present some benchmark results on the time and memory costs for these two LDR routines. In Section 3.6, we present the estimated expected error rates (EEER) for six real data cases and discuss their behavior under four eigenvector-ordering schemes and two statistical discriminant functions. Finally, we offer some concluding remarks in Section 3.7.

3.2 Notation and Methods

Consider $N \ll p$ observations or subjects from G mutually exclusive populations, $\Pi_1, \ldots, \Pi_G$, with class sample sizes $N_1, N_2, \ldots, N_G$, so that

$$N := \sum_{g=1}^{G} N_g.$$

Further, let the *a priori* class membership proportions be $\alpha_g := N_g/N$. Let $\mathbb{R}_{r\times p}$ denote the set of all $r \times p$ matrices with entries in the field $\mathbb{R}$. Let $\mathbb{S}_p \subset \mathbb{R}_{p\times p}$ denote the set of $p \times p$ real symmetric matrices, and let $\mathbb{R}_p^{>} \subset \mathbb{S}_p$ denote the interior of the cone of $p \times p$ real symmetric positive-definite matrices.

Without loss of generality, draw from group g the $1 \times p$ measurement vector for subject i,

$$\mathbf{x}_{ig} \sim f_g\left(\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g\right),$$

where $\boldsymbol{\mu}_g$ and $\boldsymbol{\Sigma}_g \in \mathbb{R}_p^{>}$ are the mean vector and covariance matrix, respectively, of the p-dimensional multivariate elliptical density f_g. Additionally, let the maximum likelihood estimates for $\boldsymbol{\mu}_g$ and $\boldsymbol{\Sigma}_g$ be

$$\bar{\mathbf{x}}_g := \frac{1}{N_g}\sum_{i=1}^{N_g} \mathbf{x}_{ig}$$

and

$$\mathbf{S}_g := \frac{1}{N_g}\sum_{i=1}^{N_g} (\mathbf{x}_{ig} - \bar{\mathbf{x}}_g)^T(\mathbf{x}_{ig} - \bar{\mathbf{x}}_g),$$

respectively. Further, let the full data matrix be denoted by $[\mathbf{x}_1 \vdots \mathbf{x}_2 \vdots \cdots \vdots \mathbf{x}_N]^T$. Finally, without loss of generality, assume that the data have been sample-mean centered, so that the data matrix is given by

$$\mathbf{X} := [\mathbf{x}_1 - \bar{\bar{\mathbf{x}}} \vdots \mathbf{x}_2 - \bar{\bar{\mathbf{x}}} \vdots \cdots \vdots \mathbf{x}_N - \bar{\bar{\mathbf{x}}}]^T \in \mathbb{R}_{N \times p},$$

where

$$\bar{\bar{\mathbf{x}}} := \frac{1}{G} \sum_{g=1}^{G} \alpha_g \bar{\mathbf{x}}_g$$

is the grand mean for the total training data.

3.2.1 Two Statistical Discriminant Functions

Assume that the group-specific covariance matrices are $\mathbf{\Sigma}_g = \mathbf{\Sigma},\ g = 1, \ldots, G$. Then the Anderson (1951) approximate linear discriminant criterion function of group g for observation $\mathbf{x}$ with pooled covariance estimate $\mathbf{S}$ is

$$\hat{d}_g^L(\mathbf{x}) := -2 \log(\alpha_g) + (\mathbf{x} - \bar{\mathbf{x}}_g) \mathbf{S}^{-1} (\mathbf{x} - \bar{\mathbf{x}}_g)^T, \tag{3.1}$$

with discriminant rule

$$\hat{d}^L(\mathbf{x}) = \min \left\{ \hat{d}_g^L(\mathbf{x});\ g = 1, \ldots, G \right\}. \tag{3.2}$$

However, if we are unable to assume the equality of covariance matrices, then the sample quadratic discriminant criterion function of group g for observation $\mathbf{x}$ is

$$\hat{d}_g^Q(\mathbf{x}) := \log |\mathbf{S}_g| - 2 \log(\alpha_g) + (\mathbf{x} - \bar{\mathbf{x}}_g) \mathbf{S}_g^{-1} (\mathbf{x} - \bar{\mathbf{x}}_g)^T, \tag{3.3}$$

with discriminant rule

$$\hat{d}^Q(\mathbf{x}) = \min \left\{ \hat{d}_g^Q(\mathbf{x});\ g = 1, \ldots, G \right\}. \tag{3.4}$$

3.2.2 Matrix Decompositions

To mitigate the effects of the curse of dimensionality, we can use linear dimension reduction to take linear combinations of all features, and then discard features according to some criterion. Often, this criterion is the relative magnitude of the eigen- or singular values. Let $n_1, n_2, \ldots, n_G$ and $m_1, m_2, \ldots, m_G$ denote the per-group training and test sample sizes, respectively, where

$$n := \sum_{g=1}^{G} n_g \ ; \quad m := \sum_{g=1}^{G} m_g;$$

so that $N = n + m$. Because $n < p$, the $\text{rank}(\mathbf{X}) \leq n$. Therefore, we discuss only the reduced dimensions $q \leq n < p$. The intuition for this approach is given in Hastie, Tibshirani, & Friedman (2001), who remark that "just like two points in three-dimensional space always lie on a line, N points in p-dimensional space lie in an $(N-1)$-dimensional affine subspace" (p. 660). This computational linear dimension reduction result is valid for any model linear in its parameters and under a quadratic penalty (p. 661).

3.2.2.1 *SVD.* The singular value decomposition decomposes the sample-mean centered data matrix $\mathbf{X}$ into three components,

$$\mathbf{X} := \mathbf{U}\mathbf{D}\mathbf{V}^T.$$

The matrix $\mathbf{U}$ is the $n \times n$ orthogonal matrix of the left singular vectors, $\mathbf{D} = \text{diag}(\delta_1, \ldots, \delta_n)$ is the $n \times n$ diagonal matrix of singular values which measure the variance of each singular vector, and $\mathbf{V}$ is the $p \times n$ matrix of right singular vectors. Let the projection $\mathbf{V}_q$ from p to $q \leq n \ll p$ dimensions be the q columns of $\mathbf{V}$ which correspond to the q largest singular values in $\mathbf{D}$. One can reduce the dimension of observation $\mathbf{x}_i$ by taking

$$\mathbf{y}_i := \mathbf{x}_i \mathbf{V}_q \in \mathbb{R}_{1 \times q},$$

for $i = 1, 2, \ldots, n$.

3.2.2.2 *PCA.* The principal component decomposition of the sample covariance matrix, $\mathbf{S}$, is

$$\mathbf{S} := \mathbf{P}\mathbf{\Lambda}\mathbf{P}^T,$$

where $\mathbf{P}$ is the orthonormal matrix of eigenvectors and $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues, such that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq \lambda_{n+1} = \cdots = \lambda_p = 0$ for $n \ll p$. Let the projection $\mathbf{P}_q$ from p to $q \ll p$ dimensions be the q columns of $\mathbf{P}$ which correspond to the q largest eigenvalues in $\mathbf{\Lambda}$. One can reduce the dimension of observation $\mathbf{x}_i$ by taking

$$\mathbf{y}_i := \mathbf{x}_i\mathbf{P}_q \in \mathbb{R}_{1\times q},$$

for $i = 1, 2, \ldots, n$.

3.2.2.3 *Relating PCA to the SVD.* We now demonstrate a direct relationship between PCA and the SVD.

Theorem. *Let the unbiased estimator of the pooled covariance matrix $\mathbf{\Sigma}$ of the mean-centered data matrix $\boldsymbol{X} \in \mathbb{R}_{n\times p}$ be given by*

$$\boldsymbol{S} = \frac{1}{n-1}\boldsymbol{X}^T\boldsymbol{X} \in \mathbb{R}_p^{\geq},$$

where $\mathbb{R}_p^{\geq}$ denotes a $p \times p$ positive semi-definite matrix. Further, let $n < p$. Then the principal axes of $\boldsymbol{S}$ are the right singular vectors of $\boldsymbol{X}$, and the eigenvalues of $\boldsymbol{S}$ are proportional to the squared singular values of $\boldsymbol{X}$; that is, for $i = 1, \ldots, n$,

$$\lambda_i := \frac{1}{n-1}\delta_i^2.$$

Proof. Let $\mathbb{E}[\mathbf{X}] = \mathbf{0}$ without loss of generality. The SVD of $\mathbf{X}$ is

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T,$$

where $\mathbf{U}$ is the orthonormal matrix of left singular vectors, $\mathbf{D} = \operatorname{diag}(\delta_1, \ldots, \delta_n)$ are the singular values, and $\mathbf{V}$ is the orthonormal matrix of right singular vectors. Thus,

the symmetric Gram matrix of $\mathbf{X}$ is decomposed as

$$\mathbf{X}^T\mathbf{X} = \left(\mathbf{U}\mathbf{D}\mathbf{V}^T\right)^T\left(\mathbf{U}\mathbf{D}\mathbf{V}^T\right) = \mathbf{V}\mathbf{D}^2\mathbf{V}^T$$
$$\implies \mathbf{X}^T\mathbf{X}\mathbf{v}_i = \delta_i^2\mathbf{v}_i,$$

for $i = 1, \ldots, n$. Therefore, $\mathbf{V}$, the right singular vectors of $\mathbf{X}$ are the eigenvectors of $\mathbf{X}^T\mathbf{X}$ with corresponding eigenvalues δ_i^2, noting that $\delta_{n+1}, \ldots, \delta_p = 0$. Further, because $\mathbf{X}$ is mean-centered, the pooled sample covariance matrix of $\mathbf{X}$ is

$$\mathbf{S} = \frac{1}{n-1}\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$$
$$\implies \mathbf{S}\mathbf{v}_i = \lambda_i\mathbf{v}_i,$$

where $\lambda_i\mathbf{v}_i$ is the i^{th} principal component, $\lambda_i = \frac{1}{n-1}\delta_i^2$, and $i = 1, \ldots, p$. Thus, theoretically, the principal axes of $\mathbf{S}$ are the right singular vectors of $\mathbf{X}$, and the eigenvalues of $\mathbf{S}$ are proportional to the squared singular values of $\mathbf{X}$. □

3.2.3 Ill-Posed Discriminant Functions

Without loss of generality, let $n_g < n_j$ for each $j \neq g$. Assume that we reduce the original feature dimension p to some arbitrary dimension $q < p$ via PCA or the SVD and recalculate the maximum likelihood estimate for $\boldsymbol{\mu}_g$ and the unbiased estimates for $\mathbf{\Sigma}_g$ and $\mathbf{\Sigma}$ in the q-dimensional subspace. For cases when $n_g \leq q < p$, the group covariance matrix for group g is singular, so (3.3) is said to be ill-posed. For cases when $n \leq q < p$, the pooled covariance matrix is singular, implying that (3.1) is also ill-posed.

We replace the inverse of $\mathbf{S}_g$ with the Moore-Penrose (MP) pseudo-inverse (Penrose, 1955). The MP pseudo-inverse $\mathbf{S}^+$ simplifies to $\mathbf{S}^{-1}$ when $q < n$. This modification yields the q-dimensional sample linear discriminant function

$$\tilde{d}_g^L(\mathbf{x}) := -2\log\left(\alpha_g\right) + \left(\mathbf{x} - \bar{\mathbf{x}}_g\right)\mathbf{S}^+\left(\mathbf{x} - \bar{\mathbf{x}}_g\right)^T. \tag{3.5}$$

Also, $\mathbf{S}_g^+$ reduces to $\mathbf{S}_g^{-1}$ when the reduced dimension $q < n_g$. Therefore, for the class-specific quadratic criterion function, $\mathbf{S}_1^{-1}, \ldots, \mathbf{S}_G^{-1}$ are replaced with $\mathbf{S}_1^+, \ldots, \mathbf{S}_G^+$,

respectively. Further, because the determinant of $\mathbf{S}_g = 0$ when $q \geq n_g$, the $\log|\mathbf{S}_g|$ tends without bound toward $-\infty$. Therefore, in (3.3) we replaced

$$\log|\mathbf{S}_g| = \sum_{i=1}^{p} \log \lambda_i$$

with

$$\log\left(\mathrm{tr}(\mathbf{S}_g)\right) = \log \sum_{j=1}^{k} \lambda_j,$$

where $k = \mathrm{rank}(\mathbf{S}_g)$. These modifications yielded the q-dimensional sample quadratic criterion function

$$\tilde{d}_g^Q(\mathbf{x}) := \log\left(\mathrm{tr}(\mathbf{S}_g)\right) - 2\log\left(\alpha_g\right) + \left(\mathbf{x} - \bar{\mathbf{x}}_g\right)\mathbf{S}_g^{+}\left(\mathbf{x} - \bar{\mathbf{x}}_g\right)^T. \tag{3.6}$$

We replace (3.1) with (3.5) for LDA or (3.3) with (3.6) for QDA, and we assign an observation $\mathbf{x}$ to the group g which minimizes the appropriate linear or quadratic discriminant rule.

3.3 Principal Direction Ordering

Often, the projection matrix from p to $q < p$ dimensions is constructed from the q eigenvectors corresponding to the largest q eigen- or singular values. However, Chang (1983) showed that reducing multivariate observations from p dimensions to q dimensions by decreasing variance order does not always guarantee that the most useful discriminatory information is preserved. That is, we may improve our low-rank classifier for $\mathbf{X}$ by projecting the observations onto some judiciously chosen subspace spanned by the q eigenvectors. Because the rank of the training-data matrix is at most n, we have only n linearly independent eigenvectors to consider as a subspace basis. Hence $q \leq n$. We examine three such alternative ordering schemes: the canonical variate criterion (CVA) of Krzanowski (1992), a heteroscedastic modification of CVA (HCVA), and the univariate Bayes quadratic error rate criterion discussed in Young, Turner, & Marco (1987b).

3.3.1 Canonical Variates

Song, Ren, & Yan (2009) have applied CVA to eigenvector ordering to improve classification of high-dimensional observations. The CVA ordering scheme seeks to find principal component which maximize the ratio of the between- to within-sums-of-squares. More precisely, let

$$\mathbf{S}_W := \sum_{g=1}^{G} \alpha_g \mathbf{S}_g \tag{3.7}$$

and

$$\mathbf{S}_B := \sum_{g=1}^{G} (\bar{\mathbf{x}}_g - \bar{\bar{\mathbf{x}}})^T (\bar{\mathbf{x}}_g - \bar{\bar{\mathbf{x}}}) = \sum_{g=1}^{G} \bar{\mathbf{x}}_g^T \bar{\mathbf{x}}_g \tag{3.8}$$

be the within group and between group sum-of-squares matrices, respectively. Calculate the *canonical variate* score of the eigenvector $\mathbf{e}_i$ as

$$\mathrm{CV}_i := \frac{\mathbf{e}_i^T \mathbf{S}_B \mathbf{e}_i}{\mathbf{e}_i^T \mathbf{S}_W \mathbf{e}_i}, \ i = 1, \ldots, n, \tag{3.9}$$

then order the eigenvectors by their CV_i scores using (3.9). A projection matrix constructed of these selected eigenvectors reduces the observations to the lower-dimensional subspace containing some portion of among-group discriminatory information.

3.3.2 Heteroscedastic Canonical Variates

In many cases the assumption that the group covariance matrices are equal is violated. Therefore, we have modified the canonical variate score in (3.9) so that the estimates of the group covariance matrices retain their individual influence. Thus, the *heteroscedastic canonical variate* score is

$$\mathrm{HCV}_i := \sum_{g=1}^{G} \alpha_g \frac{\mathbf{e}_i^T \mathbf{S}_B \mathbf{e}_i}{\mathbf{e}_i^T \mathbf{S}_g \mathbf{e}_i}, \ i = 1, \ldots, n. \tag{3.10}$$

Similarly to the CVA ordering described in Section 3.3.1, a projection matrix composed of the selected eigenvectors from their corresponding HCV_i scores will reduce the observations to a subspace containing a portion of group-specific discriminatory information, while partially accounting for unequal group covariance matrices.

3.3.3 Univariate Quadratic Parametric Error Rate

Young et al. (1987b) have discussed the two-population parametric error rate for the QDF in one dimension under the assumption of normality when the parameters are known. The univariate sample QDF is given by

$$\hat{d}_g(x) := -2\log(\alpha_g) + \frac{1}{s_g^2}(x - \bar{x}_g)^2. \tag{3.11}$$

We assign an observation x to the group g which minimizes (3.11) over g.

The steps necessary to determine this error rate are given below. First, partition the data by group, and select groups 1 and 2. Then, transform each observation by each of the eigenvectors $\mathbf{e}_i,\ i = 1, \ldots, n$, and calculate the univariate leave-one-out quadratic error rate estimate (the details are discussed in the next paragraph). Complete this same error rate calculation in a pairwise fashion for all groups, and then take a sample-size-weighted average of the pairwise error rates.

Once all the observations from groups j and k ($j \neq k$) have been transformed by the $p \times 1$ principal component $\mathbf{e}_i$, calculate $\bar{x}_j,\ \bar{x}_k,\ s_j^2$, and s_k^2. Further, assume $s_j^2 > s_k^2$ without loss of generality. The non-centrality parameters for two non-central χ^2 distributions are $\lambda_j = as_j^2$ and $\lambda_k = as_k^2$, where

$$a := \frac{(\bar{x}_j - \bar{x}_k)^2}{(s_j^2 - s_k^2)^2}. \tag{3.12}$$

The two critical values are $\chi_j^* = bs_k^2$ and $\chi_k^* = bs_j^2$, where

$$b := \frac{1}{s_j^2 - s_k^2}\left[\log\left(\frac{s_j^2}{s_k^2}\right) + \frac{(\bar{x}_j - \bar{x}_k)^2}{s_j^2 - s_k^2}\right]. \tag{3.13}$$

Therefore, the estimated probability of classifying an observation $\mathbf{x}$ drawn from group j incorrectly into group k after projection onto eigenvector $\mathbf{e}_i$ is

$$P\left[\hat{d}_k(\mathbf{x}\mathbf{e}_i) < \hat{d}_j(\mathbf{x}\mathbf{e}_i) \mid \mathbf{x} \in \Pi_j\right] \approx \int_0^{\chi_j^*} \chi^2(1, \lambda_j),$$

while the estimated probability of classifying $\mathbf{x}$ drawn from group k incorrectly into group j after projection onto $\mathbf{e}_i$ is

$$P\left[\hat{d}_j(\mathbf{x}\mathbf{e}_i) < \hat{d}_k(\mathbf{x}\mathbf{e}_i) \mid \mathbf{x} \in \Pi_k\right] \approx \int_{\chi_k^*}^{\infty} \chi^2(1, \lambda_k),$$

where $\hat{d}$ is given in (3.11). We calculate these weighted misclassification probabilities for all pairs of j and k, and then average these probabilities to render the final estimated error-rate score for each principal component. One then orders the eigenvectors from the smallest to the largest using the weighted average pairwise estimated error.

3.4 Bootstrap Design

For each of the six real-data cases in Section 3.6, we held half of the observations as training data at random, and kept the other half for test data. The test data set was centered on the mean of the training data, and the dimension was reduced via the eigenvectors from the training data. This step ensured that the test data were not used in any way for training purposes. We completed a non-parametric bootstrap replication on each data set 1,000 times for data sets with $N \times p^2 <= 10^9$ and 100 times for data sets with $N \times p^2 > 10^9$, where $N \times p^2$ is shown in Table 3.1.

Within each replicate, we

(1) Draw, at random, half of the observations for training.

(2) Mean-center the train data, and then center the test data with the training-sample mean.

(3) Calculate the full $p \times n$ matrix of eigenvectors:

   (a) For PCA, construct the sample pooled covariance matrix, and decompose this covariance as

$$\mathbf{S} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^T.$$

   Take the n columns of $\mathbf{P}$ associated with the n non-zero eigenvalues of $\mathbf{\Lambda}$ as a dimension reduction matrix.

   (b) For the SVD, factor a training data matrix as

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T.$$

Take the n columns of $\mathbf{V}$ associated with the Nn non-zero eigenvalues of $\mathbf{D}$ as the dimension reduction matrix.

(4) Separately order the eigenvectors by each of the following criterion:

   (a) Eigenvalue magnitude (no change in order).

   (b) Canonical variate ordering (CVA) of Section 3.3.1.

   (c) Heteroscedastic canonical variates ordering (HCVA) of Section 3.3.2.

   (d) Univariate quadratic parametric error rate ordering (QPER) of Section 3.3.3.

(5) For each $q \in 1, \ldots, n$, map the training and test data sets from p to q dimensions.

(6) Classify the test observations in the q-dimensional reduced-feature subspace, and record the conditional error rate for that sample.

Repeat this process 100 or 1,000 times, as described in the previous paragraph, and average the estimated conditional error rates and store these EEERs for each combination of LDR method, eigenvector ordering scheme, and statistical classifier considered.

3.5 Computational Costs and Numerical Stability

3.5.1 Precision Error Conditions

Let $\mathbf{z} \in \mathbb{R}_{p\times 1}$. Recall the ℓ_2 norm of a matrix $\mathbf{A} \in \mathbb{R}_{r\times p}$,

$$||\mathbf{A}||_2 := \max_{\mathbf{z}\neq\mathbf{0}} \left(\frac{||\mathbf{A}\mathbf{z}||_2}{||\mathbf{z}||_2} \right),$$

where $||\mathbf{z}||_2 = \sqrt{z_1^2 + z_2^2 + \cdots + z_p^2}$. The condition number of $\mathbf{A}$ is defined to be

$$\kappa(\mathbf{A}) := \frac{\max\{\sigma(\mathbf{A})\}}{\min\{\sigma(\mathbf{A})\}},$$

where $\max \sigma(\mathbf{A})$ and $\min \sigma(\mathbf{A})$ are the largest and smallest strictly positive singular values of $\mathbf{A}$, respectively. As discussed in Golub & Van Loan (1996), the condition

number of a matrix measures the "rate of change" of the estimator of the matrix with respect to small error (p. 80). The larger the condition number, the more inaccurate calculations using $\mathbf{A}$ will be.

Now recall the relationship between the SVD and PCA of a mean-centered data matrix, $\mathbf{X} \in \mathbb{R}_{N \times p}$. The SVD factors the data matrix as

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T,$$

while PCA factors the scatter matrix,

$$\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{D}\mathbf{U}^T\mathbf{U}\mathbf{D}\mathbf{V}^T = \mathbf{V}\mathbf{D}^2\mathbf{V}^T,$$

yielding positive singular values $\sigma(\mathbf{X}^T\mathbf{X}) = [\sigma(\mathbf{X})]^2$. Therefore,

$$\kappa\left(\mathbf{X}^T\mathbf{X}\right) = \frac{\max\left\{\sigma(\mathbf{X}^T\mathbf{X})\right\}}{\min\left\{\sigma(\mathbf{X}^T\mathbf{X})\right\}} = \frac{\max\left\{[\sigma(\mathbf{X})]^2\right\}}{\min\left\{[\sigma(\mathbf{X})]^2\right\}} = \left[\frac{\max\left\{[\sigma(\mathbf{X})]\right\}}{\min\left\{[\sigma(\mathbf{X})]\right\}}\right]^2 = \kappa^2\left(\mathbf{X}\right).$$

That is, the computational error associated with estimating the PCA of a scatter matrix is the square of the computational error yielded by estimating the SVD of the original data. Any computational error in estimating the reduced-dimensional data, $\mathbf{w} = \mathbf{A}\mathbf{z}$, using the SVD would be squared if PCA were used instead.

3.5.2 Complexity Costs

For PCA, one must compute the orthonormal matrix $\mathbf{P} \in \mathbb{R}_{p \times p}$ and the diagonal matrix $\mathbf{\Lambda} \in \mathbb{R}_p^{\geq}$, while for the SVD one must compute the orthogonal matrix $\mathbf{U} \in \mathbb{R}_{N \times N}$, the diagonal matrix $\mathbf{D} \in \mathbb{R}_N^{\geq}$, and the matrix of orthogonal vectors $\mathbf{V} \in \mathbb{R}_{p \times N}$. According to Section 5.5.9 of Golub & Van Loan (1996), the time complexity for the SVD of $\mathbf{X} \in \mathbb{R}_{N \times p}$ is $\mathrm{O}(Np^2)$, while the time complexity for the PCA of $\mathbf{X}^T\mathbf{X} \in \mathbb{R}_p^{\geq}$ is $\mathrm{O}(p^3)$. Furthermore, the initial space required is Np for PCA and SVD, and the non-zero entries of the diagonal matrices $\mathbf{\Lambda}$ and $\mathbf{D}$ only require N to store. So the additional space complexity of PCA is $N + 2p^2 \in \mathrm{O}(p^2)$, while the additional space complexity of SVD is only $N + N^2 + Np \in \mathrm{O}(Np)$. For $N \ll p$, these differences in

time and space complexity can be large. However, for the majority of practitioners in other branches of statistics, p is not prohibitively large, so PCA is less computationally expensive than SVD. We speculate that this fact explains why PCA and its many incarnations and modifications are the "de facto" linear dimension reduction routines for most of the sciences.

Unfortunately, when $N < p$ we still find researchers attempting to employ PCA despite its prohibitive $\mathrm{O}(p^2)$ space and $\mathrm{O}(p^3)$ time costs (Song et al., 2009). A common technique in the high-dimensional setting is the use of pre-process feature selection, wherein features are retained through some thresholding mechanism such as multiple t-tests, correlation tests, and other methods. However, many of these techniques retain redundant information. For a thorough discussion of these approaches, see Liu & Motoda (1998). Alternatively, the SVD is well known to preserve *all* of the energy of $\mathbf{X}$ in an $(N-1)$-dimensional subspace, as discussed here in Section 3.2.2.

The time and space complexity summaries listed in Table 3.1 demonstrate how much more computationally expensive PCA can be in comparison to the SVD. Recall that PCA additionally requires the computation of $\mathbf{X}^T\mathbf{X}$, which is also $\mathrm{O}(p^3)$, but the calculation of the eigen-decomposition of this scatter matrix can be made more efficient by operating on the lower triangular of this symmetric matrix rather than the full $p \times p$ matrix. The machine we used for these benchmark comparisons was a Dell Optiplex 9020 with 64-bit OS with an Intel Core i7-4790 3.60 GHz processor and 16 GB of RAM. For computation costs on the moderately-sized data set shown in row 4 of Table 3.1 (labeled "Khan"), we observe that the calculation time to multiply $\mathbf{X}^T\mathbf{X}$ and perform the symmetric eigendecomposition is 12.12 seconds and requires 2.4×10^8 total bytes of memory. In contrast, the SVD does not require the multiplication of $\mathbf{X}^T\mathbf{X}$. Further, the calculation time and space for the same data set are reduced considerably to 0.01 seconds and only 6.0×10^6 bytes of memory, respectively. The

SVD calculations for the "Khan" data set are over 1,200 times faster and nearly 40 times more memory efficient than their PCA counterparts.

Table 3.1: Computation times and memory allocated for PCA and SVD on some real high-dimensional data sets. Data sets are described in their respective subsections in Section 3.6.

	Dimensions			Time (seconds)			Memory (bytes)		
Data Set	N	p	$N \times p^2$	PCA	SVD	Ratio	PCA	SVD	Ratio
Alon	62	2000	2.5×10^8	8.25	0.01	825	1.8×10^8	5.1×10^6	34.5
Golub	72	7129	3.7×10^9	336.70	0.07	4810	2.2×10^9	2.1×10^7	107.9
Gravier	168	2905	1.4×10^9	24.43	0.14	175	3.7×10^8	2.1×10^7	18.0
Khan	63	2308	3.4×10^8	12.12	0.01	1212	2.4×10^8	6.0×10^6	39.3
Sorlie	85	456	1.8×10^7	0.10	0.02	5	7.7×10^6	1.8×10^6	4.1
van't Veer	77	250	4.8×10^6	0.02	0.01	2	2.9×10^6	1.0×10^6	2.8

3.6 Results for Six Real Data Sets

For Figures 3.1 – 3.6, each point represents the EEER for each reduced dimension q for one of the 16 possible classifiers we have discussed: the fully-factorial combination of discriminant functions (LDF or QDF), projection methods (PCA or the SVD), and one of the four eigenvector ordering schemes described in Section 3.3. The left column of graphs shows EEER results for dimension reduction with PCA, while the right column shows EEER results for dimension reduction with the SVD. The top row of graphs shows EEER results for classification with the LDF, while the graphs on the bottom row show EEER results for classification with the QDF. Because of the heteroscedastic assumption of the QDF, we expect the behavior of error rates for the reduced dimensions $q < \min\{n_g\}$ to be different from the EEER behavior for the reduced dimensions $q \geq \min\{n_g\}$ in the bottom row of figures. Finally, the plot colors and shapes distinguish the four dimension ordering schemes discussed in Section 3.3.

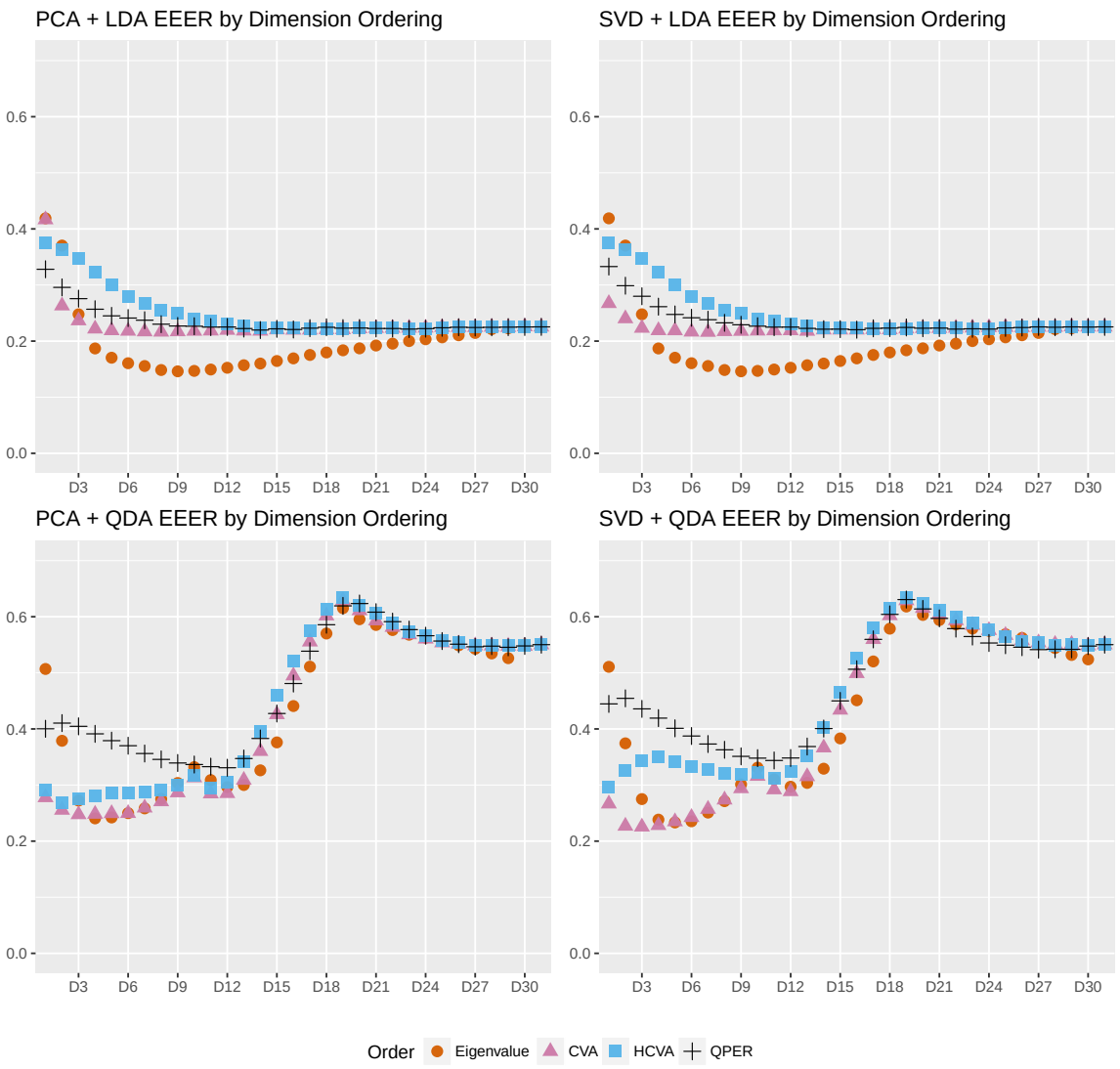


Figure 3.1: EEER by reduction method, discriminant function, and eigenvector order for the colon cancer data set of Alon et al. (1999).

3.6.1 The Alon et al. Data Set

The data set of Alon, Barkai, Notterman, Gish, Ybarra, Mack, & Levine (1999) has 62 observations in two groups with 2,000 measured features. The observations consist of 40 cancerous and 22 non-cancerous biopsies of colon tissue, while the features are gene expression levels from an oligonucleotide array. We randomly chose $n_1 = 20$ cases and $n_2 = 11$ controls as training data, and the remaining observations were used as test data. This bootstrap sampling sampling process was repeated 1,000 times to determine 1,000 conditional error rates (CERs) for each supervised classifier, which were then averaged to yield the EEERs.

Table 3.2: Minimum EEERs by reduction method, discriminant function, and eigenvector ordering scheme for the colon cancer data set of Alon et al. (1999).

		PCA		SVD	
Discrim	Order	Mean	Dim	Mean	Dim
LDA	Eigenvalue	0.146	9	0.146	9
LDA	CVA	0.216	8	0.216	7
LDA	HCVA	0.221	19	0.221	19
LDA	QPER	0.220	14	0.220	16
QDA	Eigenvalue	0.241	4	0.233	5
QDA	CVA	0.247	3	0.225	3
QDA	HCVA	0.268	2	0.296	1
QDA	QPER	0.331	12	0.344	11

As one can see in the top row of Figure 3.1, the EEERs for PCA and SVD were identical when we classified test observations with the LDF. Similarly, the EEERs for PCA and SVD were almost identical when we classified test observations with the QDF, with the slight exception of the HCVA ordering approach for the reduced dimensions $2 \leq q \leq 9$. Also, eigenvector ordering had a greater effect as q decreased. As the number of principal components decreased, the EEERs became more sensitive to the choice of principal components. This result explains the convergence of EEERs for the four eigenvector ordering methods as q increased. For classification with the

LDF, ordering by the eigenvalues yielded the smallest EEERs. For classification with the QDF, the CVA eigenvector ordering process yielded EEERs roughly equivalent to traditional eigenvalue ordering for the reduced dimensions $q \geq 4$, and yielded smaller EEERs for $1 \leq q \leq 3$. Recall that for the QDF, the ordered class training-sample sizes were $n_2 = 11$ and $n_1 = 20$, so we see different EEER behavior over three intervals: $1 \leq q \leq 10$, $11 \leq q \leq 19$, and $q \geq 20$. For classification with the QDF, we observed that ordering the eigenvectors by some criterion which incorporated group information yielded smaller EEERs for the QDF in the lowest-dimensional subspaces when compared to default eigenvalue-based eigenvector ordering approach. We remark that the choice of principal components had the strongest effect when fewer principal components were selected; that is, the ordering scheme of the eigenvectors mattered more for LDA when $q \leq 13$ and QDA when $q \leq 10$.

As one can see in Table 3.2, after we blocked on the type of discriminant function and eigenvector ordering techniques, the overall minimum EEERs were very similar for the PCA and SVD dimension-reduction methods. For example, the best error rate for this colon cancer classification data set was given by LDA coupled eigenvector ordering by decreasing eigenvalue. This error rate was 0.141 when the reduced dimension was $q = 10$ for PCA and 0.146 when $q = 9$ for the SVD. For this data set, dimension reduction with the SVD yielded non-inferior EEERs to dimension reduction with PCA and completed calculations 825 times faster while requiring 34.5 times less memory. For the data set of Alon et al. (1999), we concluded that classifier performance was essentially equivalent when the computationally-superior SVD was used in comparison to PCA for feature-dimension reduction.

3.6.2 The Golub et al. Data Set

The data set of Golub, Slonim, Tamayo, Huard, Gaasenbeek, Mesirov, Coller, Loh, Downing, Caligiuri, Bloomfield, & Lander (1999) has two groups with 7,129 features

Figure 3.2: EEER by reduction method, discriminant function, and eigenvector order for the leukemia data set of Golub et al. (1999).

recorded on 72 subjects. The observations consist of 47 acute myeloid leukemia (AML) and 25 acute lymphoblastic leukemia (ALL) samples, while the features are gene expression levels from a microarray. We randomly chose $n_1 = 24$ AML and $n_2 = 13$ ALL samples as training data, and the remaining observations were used as test data. Because $Np^2 > 10^9$, this bootstrap sampling process was repeated 100 times to determine 100 CERs for each supervised classifier, which were then averaged to yield the EEERs.

Table 3.3: Minimum EEERs by reduction method, discriminant function, and eigenvector ordering scheme for the leukemia data set of Golub et al. (1999).

		PCA		SVD	
Discrim	Order	Mean	Dim	Mean	Dim
LDA	Eigenvalue	0.048	23	0.048	23
LDA	CVA	0.051	27	0.051	26
LDA	HCVA	0.052	29	0.052	31
LDA	QPER	0.053	32	0.053	31
QDA	Eigenvalue	0.071	7	0.085	8
QDA	CVA	0.056	8	0.066	6
QDA	HCVA	0.057	7	0.068	6
QDA	QPER	0.087	16	0.093	16

As one can see in the top row of Figure 3.2, the EEERs for PCA and SVD were nearly identical when we classified test observations with the LDF, except for the CVA eigenvector ordering strategy when $q = 1$. Similarly, the EEERs for PCA and SVD were very similar when we classified test observations with the QDF. The EEER for eigenvalue-based ordering was comparatively smaller when $q = 1$ and larger when $q = 2$. Also, the EEERs under the QPER ordering approach were slightly smaller for the reduced dimensions $25 \leq q \leq 31$ when reducing the dimension with the SVD. Once again, the EEERs were more dispersed for the four eigenvector ordering methods, in this case for $q \leq 23$. For classification with the QDF, the CVA and HCVA ordering schemes yielded EEERs roughly equivalent to eigenvalue-based ordering for

the reduced dimensions $1 \leq q \leq 12$ and $26 \leq q \leq 37$. Recall that for the QDF, the ordered class training-sample sizes were $n_2 = 13$ and $n_1 = 24$, so we see different EEER behavior over three intervals: $1 \leq q \leq 12$, $13 \leq q \leq 23$, and $q \geq 24$. For $4 \leq q \leq 14$, we remark that, for this data set, employing the QPER eigenvector ordering technique to choose principal components in conjunction with the QDF yielded larger EEERs in comparison to the other ordering schemes.

As one can see in Table 3.3, after we blocked on the type of discriminant function and eigenvector ordering methods, the overall minimum EEERs were nearly identical between the PCA and SVD dimension-reduction methods for the LDF, which differed slightly in the reduced dimension yielding the minimum EEER. Further, these minimum EEERs were within a range of 0.015 for the QDF. Additionally, using eigenvalue-based eigenvector ordering in conjunction with the LDF yielded the smallest EEER, but the other ordering method EEERs were within a range of 0.005. For this data set, dimension reduction with the SVD yielded non-inferior EEERs to dimension reduction with PCA, while completing calculations 4,810 times faster and requiring 107.9 times less memory. For the data set of Golub et al. (1999), we concluded that classifier performance was essentially equivalent between the computationally-superior SVD and PCA for feature-dimension reduction.

3.6.3 The Gravier et al. Data Set

The data set of Gravier, Pierron, Vincent-Salomon, Gruel, Raynal, Savignoni, De Rycke, Pierga, Lucchesi, Reyal, Fourquet, Roman-Roman, Radvanyi, Sastre-Garau, Asselain, & Delattre (2010) has two groups with 2,905 features recorded on 168 subjects. The observations consist of 111 patients labeled "good" and 57 patients labeled "poor", and all patients have been diagnosed with T1T2 node-negative breast cancer. A "good" label represents that the patient has had no metastasizing in the five years after a positive diagnosis while a "poor" label represents that the patient has experienced

metastasis after diagnosis. The features are gene expression levels from a comparative genomic hybridization array. We randomly chose $n_1 = 50$ "good" and $n_2 = 28$ "poor" samples as training data, and the remaining observations were used as test data. Because $Np^2 > 10^9$, this bootstrap sampling process was repeated 100 times to determine 100 CERs for each supervised classifier, which were then averaged to yield the EEERs.

Table 3.4: Minimum EEERs by reduction method, discriminant function, and eigenvector ordering scheme for the breast cancer prognosis data set of Gravier et al. (2010).

		PCA		SVD	
Discrim	Order	Mean	Dim	Mean	Dim
LDA	Eigenvalue	0.254	24	0.254	24
LDA	CVA	0.256	49	0.256	49
LDA	HCVA	0.256	71	0.256	71
LDA	QPER	0.255	67	0.255	67
QDA	Eigenvalue	0.313	26	0.313	28
QDA	CVA	0.279	16	0.280	17
QDA	HCVA	0.285	15	0.289	14
QDA	QPER	0.309	38	0.313	38

As one can see in the top row of Figure 3.3, the EEERs for PCA and SVD were identical when we classified test observations with the LDF. Similarly, the EEERs for PCA and SVD were nearly identical when we classified test observations with the QDF, with the only difference in the reduced dimensions $q \leq 2$ for the eigenvalue-based eigenvector ordering approach. Once again, the EEERs were more disperse for the four ordering methods, for $q \leq 39$ in this case. For classification with the QDF, the CVA and HCVA eigenvector ordering techniques yielded EEERs roughly equivalent to eigenvalue-based ordering for all reduced dimensions and much smaller EEERs when $q \leq 49$. Recall that for the QDF, the ordered class training-sample sizes were $n_2 = 28$ and $n_1 = 50$, so we see different EEER behavior over three intervals:

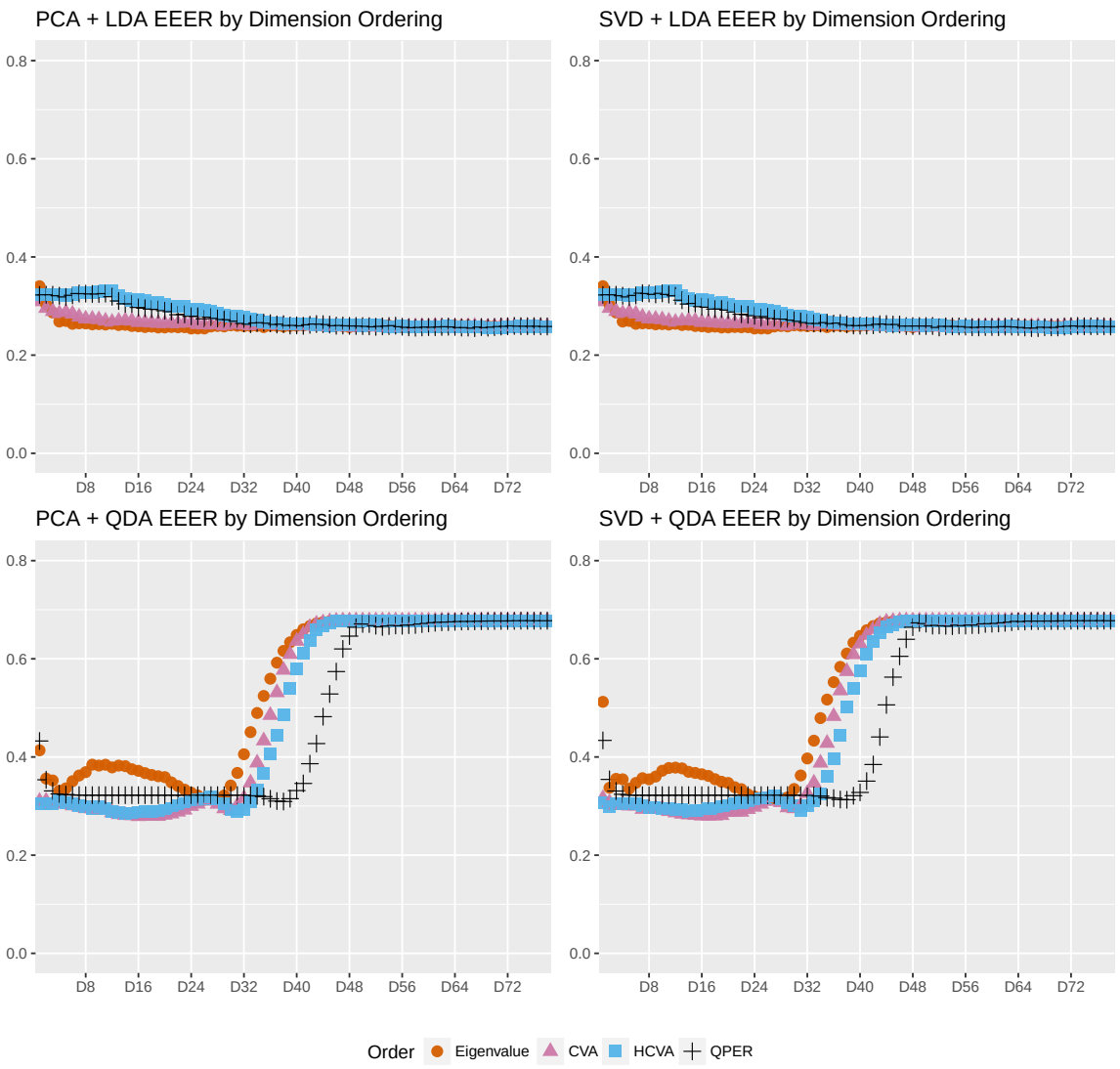


Figure 3.3: EEER by reduction method, discriminant function, and eigenvector order for the breast cancer prognosis data set of Gravier et al. (2010).

$1 \leq q < 27$, $28 \leq q \leq 49$, and $q \geq 50$. For $q \leq 27$, we remark that, for this data set, employing the eigenvalue-based eigenvector ordering strategy to choose principal components in conjunction with the QDF yielded larger EEERs in comparison to the other eigenvector ordering schemes.

As one can see in Table 3.4, after we blocked on the type of discriminant function and eigenvector ordering processes, the overall minimum EEERs were identical between the PCA and SVD dimension-reduction methods for the LDF and were within a range of 0.004 for the QDF. Additionally, all ordering schemes with the LDF yielded the smallest EEER, but eigenvector ordering methods had an effect on EEER for the QDF. For this data set, dimension reduction with the SVD yielded non-inferior EEERs to dimension reduction with PCA, while completing calculations 175 times faster and requiring 18 times less memory. For the data set of Gravier et al. (2010), we concluded that classifier performance was essentially equivalent when the computationally-superior SVD was used compared to PCA for feature-dimension reduction.

3.6.4 The Khan et al. Data Set

The data set of Khan, Wei, Ringner, Saal, Ladanyi, Westermann, Berthold, Schwab, Antonescu, Peterson, & Meltzer (2001) has 63 observations in four groups with 2,308 recorded features. The observations consist of four different types of small, round blue-cell tumors: neuroblastoma (NB), rhabdomyosarcoma (RMS), non-Hodgkin lymphoma (NHL) and the Ewing family of tumors (EWS). The data set contains 12 NB, 20 RMS, 8 NHL, and 23 EWS observations, while the features are gene expression signatures from a complementary-DNA microarray. We randomly chose $n_1 = 6$ NB, $n_2 = 10$ RMS, $n_3 = 4$ NHL, and $n_4 = 11$ EWS samples as training data, and the remaining observations were used as test data. This bootstrap sampling process was

repeated 1,000 times to determine 1,000 CERs for each supervised classifier, which were then averaged to yield the EEERs.

Table 3.5: Minimum EEERs by reduction method, discriminant function, and eigenvector ordering scheme for the small, round blue-cell cancer data set of Khan et al. (2001).

		PCA		SVD	
Discrim	Order	Mean	Dim	Mean	Dim
LDA	Eigenvalue	0.017	30	0.014	30
LDA	CVA	0.017	30	0.017	30
LDA	HCVA	0.017	30	0.017	30
LDA	QPER	0.017	31	0.017	31
QDA	Eigenvalue	0.276	4	0.324	6
QDA	CVA	0.225	4	0.217	4
QDA	HCVA	0.225	4	0.217	4
QDA	QPER	0.364	4	0.353	4

As one can see in the top row of Figure 3.4, the EEERs for PCA and SVD were essentially identical when we classified test observations with the LDF. Similarly, the EEERs for PCA and SVD were nearly identical when we classified test observations with the QDF, with the only difference in the reduced dimensions $q \leq 5$ for eigenvalue-based ordering approach. We again observed a diverging of the EEERs for the four eigenvector ordering procedures for $q \leq 19$, but these differences were smaller in comparison to the EEER divergences seen in the previous data sets. For classification with the QDF, the CVA and HCVA ordering strategies yielded EEERs roughly equivalent to eigenvalue-based eigenvector ordering for all reduced dimensions and much smaller EEERs when $q \leq 10$. Recall that for the QDF, the ordered class training-sample sizes were $n_3 = 4$, $n_1 = 6$, $n_2 = 10$, and $n_4 = 11$, so we see different EEER behavior over five interval sets: $1 \leq q \leq 3$, $4 \leq q \leq 5$, $6 \leq q \leq 9$, $q = 10$, and $q \geq 11$. Further, we remark that for this data set, employing the eigenvalue-based

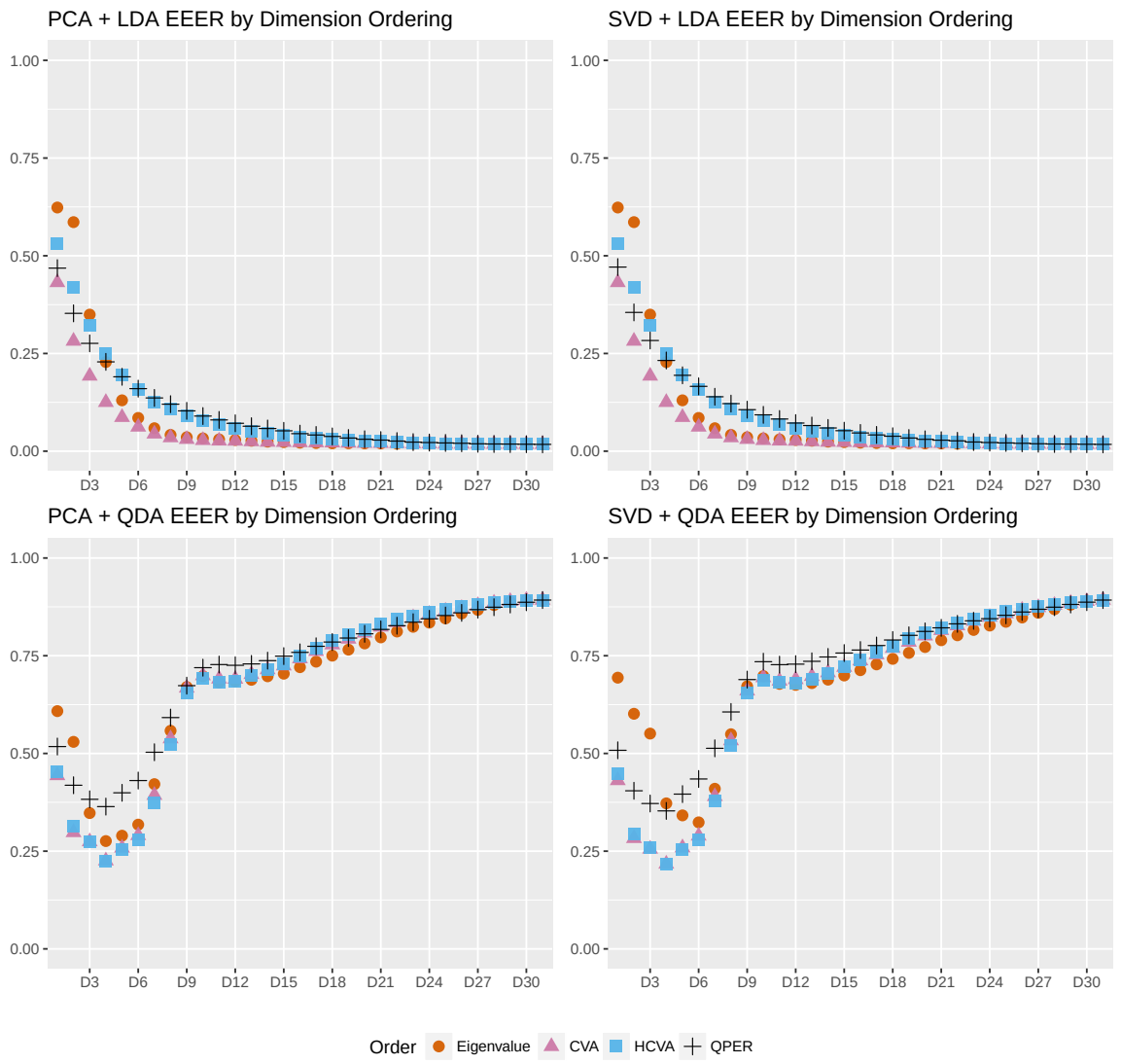


Figure 3.4: EEER by reduction method, discriminant function, and eigenvector order for the small, round blue-cell cancer data set of Khan et al. (2001).

eigenvector ordering approach to choose principal components in conjunction with the QDF yielded larger EEERs in comparison to the other ordering schemes for $q \leq 3$.

As one can see in Table 3.5, after we blocked on the type of discriminant function and eigenvector ordering strategies, the overall minimum EEERs were nearly identical between the PCA and SVD dimension-reduction methods for the LDF, while eigenvalue-based ordering in conjunction with the SVD had a slightly smaller EEER than for PCA. For the QDF, the EEERs were also slightly smaller for all eigenvector ordering schemes, other than the eigenvalue-based ordering scheme, but the EEERs were within a range of 0.01 for the other three ordering processes. Additionally, all ordering schemes with the LDF yielded the smallest EEER for both the SVD and PCA, but eigenvector ordering methods had an effect on EEER for the QDF. For this data set, dimension reduction with the SVD yielded non-inferior EEERs to dimension reduction with PCA, while completing calculations 1,212 times faster and requiring 39.3 times less memory. For the data set of Khan et al. (2001), we concluded that classifier performance was essentially equivalent between the computationally-superior SVD and PCA for feature-dimension reduction.

3.6.5 The Sorlie et al. Data Set

The data set of Sorlie, Perou, Tibshirani, Aas, Geisler, Johnsen, Hastie, Eisen, Rijn, Jeffrey, Thorsen, Quist, Matese, Brown, Botstein, Lonning, & Borresen-Dale (2001) has 85 observations in five groups with 456 recorded features. The observations consist of five different subtypes of breast cancer tumors: basal-like (14 observations), ErbB2+ (11 observations), normal (13 observations), luminal C or B (15 observations), and luminal A (32 observations). The features are gene expression signatures from a complementary-DNA microarray. We randomly chose $n_1 = 7$ basal-like, $n_2 = 5$ ErbB2+, $n_3 = 6$ normal, $n_4 = 7$ luminal C / B, and $n_5 = 16$ luminal A samples as training data, and the remaining observations were used as test data. This bootstrap

sampling process was repeated 1,000 times times to determine 1,000 CERs for each supervised classifier, which were then averaged to yield the EEERs.

Table 3.6: Minimum EEERs by reduction method, discriminant function, and eigenvector ordering scheme for the breast cancer data set of Sorlie et al. (2001).

		PCA		SVD	
Discrim	Order	Mean	Dim	Mean	Dim
LDA	Eigenvalue	0.177	21	0.177	21
LDA	CVA	0.185	11	0.185	11
LDA	HCVA	0.203	33	0.203	33
LDA	QPER	0.205	41	0.205	41
QDA	Eigenvalue	0.246	2	0.247	2
QDA	CVA	0.256	2	0.257	2
QDA	HCVA	0.273	3	0.276	3
QDA	QPER	0.374	3	0.370	3

As one can see in the top row of Figure 3.5, the EEERs for PCA and SVD were identical. While grouping on the four eigenvector ordering schemes, the among-group variability of the EEERs increased with the linear and quadratic classifiers as q decreased. For classification with the LDF, the CVA and eigenvalue-based ordering strategies yielded smaller EEERs than the HCVA and QPER eigenvector ordering approaches. For classification with the QDF, the three ordering procedures, other than QPER, yielded nearly equivalent EEERs. Recall that for the QDF, the ordered class training-sample sizes were $n_2 = 5$, $n_3 = 6$, $n_1 = 7$, $n_4 = 7$, and $n_5 = 16$, so we see different EEER behavior over five interval sets: $1 \leq q \leq 4$, $q = 5$, $q = 6$, $7 \leq q \leq 15$, and $q \geq 16$. For $q \leq 7$, we remark that, for this data set, employing the QPER eigenvector ordering method to choose principal components in conjunction with the QDF yielded larger EEERs in comparison to the other eigenvector ordering schemes.

As one can see in Table 3.6, after we blocked on the type of discriminant function and eigenvector ordering approaches, the overall minimum EEERs were identical be-

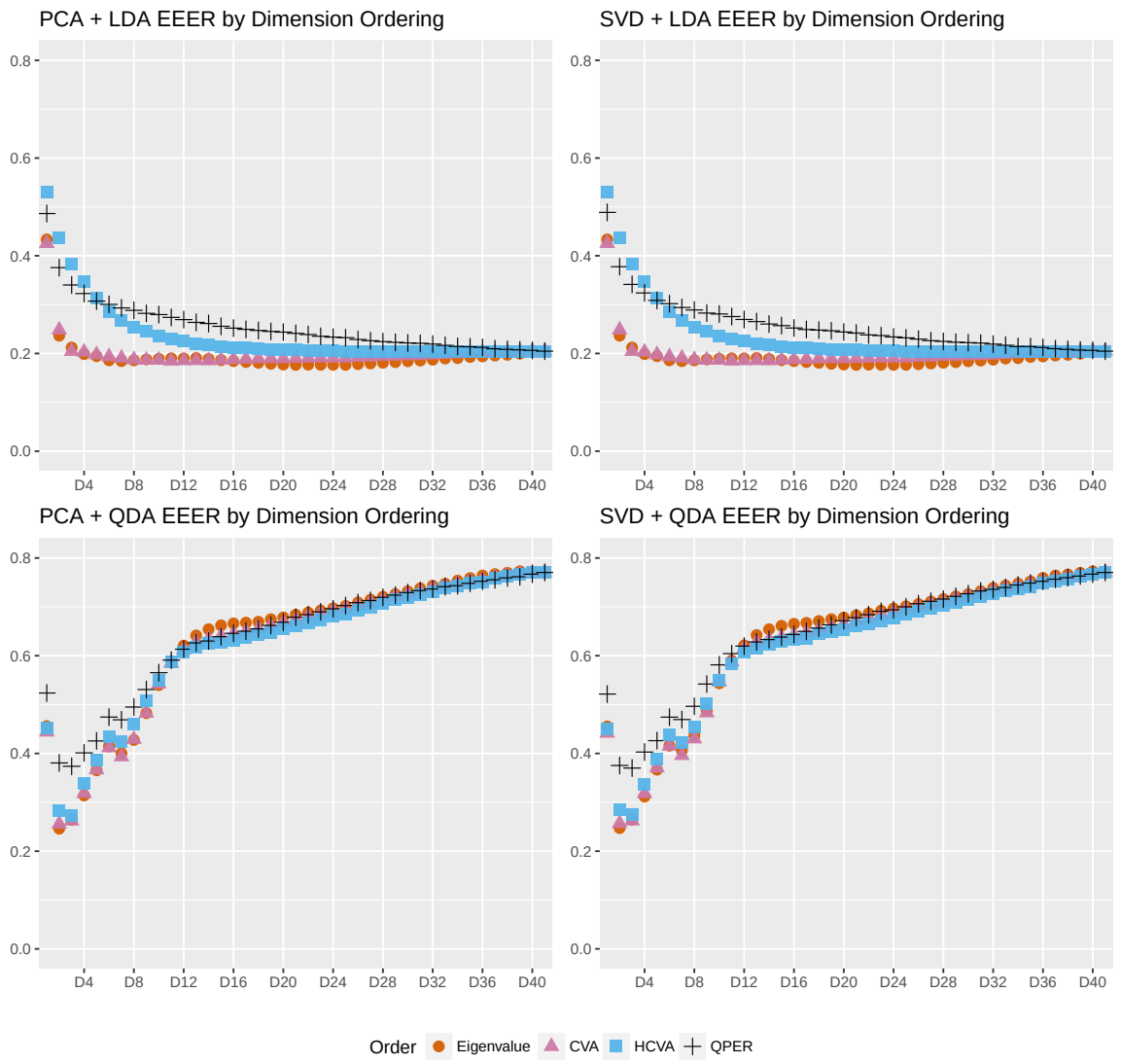


Figure 3.5: EEER by reduction method, discriminant function, and eigenvector order for the breast cancer data set of Sorlie et al. (2001).

tween the PCA and SVD dimension-reduction methods for the LDF and were within a range of 0.004 for the QDF. Also, the eigenvalue-based ordering scheme yielded the smallest EEER for both LDA and QDA. Overall, the EEERs of the eigenvector ordering schemes with the LDF were within a range of 0.03 for both the SVD and PCA, but ordering methods had a stronger effect on EEER for the QDF. For this data set, dimension reduction with the SVD yielded non-inferior EEERs to dimension reduction with PCA, while completing calculations five times faster and requiring 4.1 times less memory. For the data set of Sorlie et al. (2001), we concluded that classifier performance was essentially equivalent when the computationally-superior SVD was used compared to PCA for feature-dimension reduction.

3.6.6 The van't Veer et al. Data Set

The data set of van't Veer, Dai, van de Vijver, He, Hart, Mao, Peterse, van der Kooy, Marton, Witteveen, Schreiber, Kerkhoven, Roberts, Linsley, Bernards, & Friend (2002) has 78 subjects in two groups with 250 features independently selected by biologists. One case was removed due to majority missing feature values. The observations consist of 44 patients labeled "good" and 33 patients labeled "poor", and all patients have been diagnosed with lymph node negative breast cancer. Similarly to Section 3.6.3, a "good" label represents the patient has had no metastasizing in the five years after a positive diagnosis while a "poor" label represents that the patient has experienced metastasis after diagnosis. The features are gene expression levels from a DNA microarray. We randomly chose $n_1 = 22$ "good" and $n_2 = 16$ "poor" subjects as training data, and the remaining observations were used as test data. This bootstrap sampling process was repeated 1,000 times times to determine 1,000 CERs for each supervised classifier, which were then averaged to yield the EEERs.

As one can see in the top row of Figure 3.6, the EEERs for PCA and SVD were identical when we classified test observations with the LDF and nearly iden-

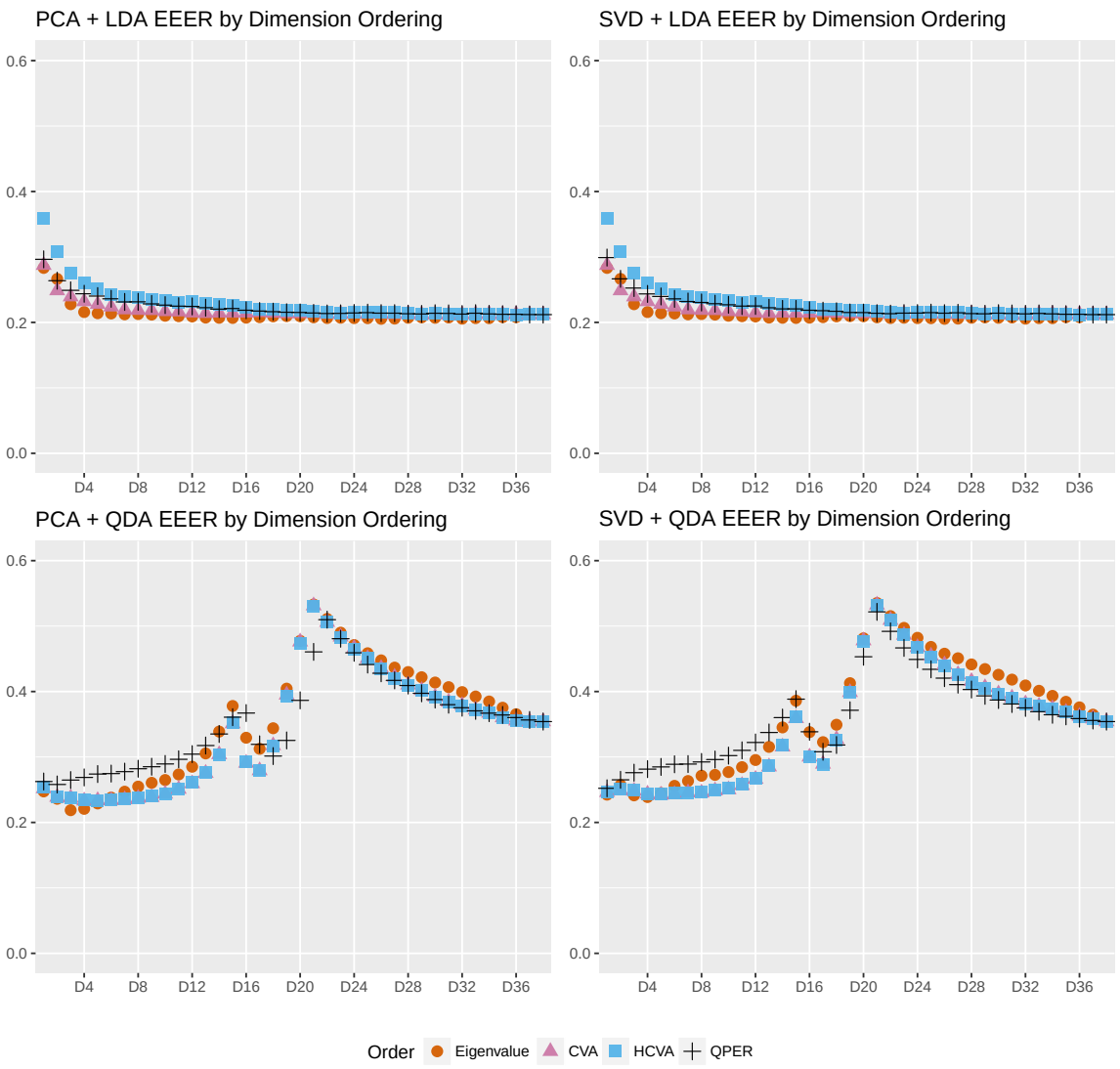


Figure 3.6: EEER by reduction method, discriminant function, and eigenvector order for the breast cancer prognosis data set of van't Veer et al. (2002).

Table 3.7: Minimum EEERs by reduction method, discriminant function, and eigenvector ordering scheme for the breast cancer prognosis data set of van't Veer et al. (2002).

		PCA		SVD	
Discrim	Order	Mean	Dim	Mean	Dim
LDA	Eigenvalue	0.205	26	0.205	26
LDA	CVA	0.212	31	0.212	31
LDA	HCVA	0.212	36	0.212	36
LDA	QPER	0.212	38	0.212	38
QDA	Eigenvalue	0.219	3	0.239	4
QDA	CVA	0.233	4	0.242	5
QDA	HCVA	0.234	5	0.244	5
QDA	QPER	0.258	2	0.252	1

tical with the QDF. The EEERs were slightly more dispersed with the linear and quadratic classifiers as q decreased. For classification with the LDF, the eigenvalue-based eigenvector ordering strategy yielded slightly smaller EEERs than the three other ordering techniques. For classification with the QDF, the eigenvector ordering schemes (other than QPER for the reduced dimensions $2 \leq q \leq 15$) yielded nearly equivalent EEERs. Recall that for the QDF, the ordered class training-sample sizes were $n_2 = 16$ and $n_1 = 22$, so we see different EEER behavior over three intervals: $1 \leq q \leq 15$, $16 \leq q \leq 21$, and $q \geq 22$. For $q \leq 15$, we remark that, for this data set, employing the QPER eigenvector ordering procedure to choose principal components in conjunction with the QDF yielded slightly larger EEERs in comparison to the other ordering processes.

As one can see in Table 3.7, after we blocked on the type of discriminant function and eigenvector ordering techniques, the overall minimum EEERs were identical between the PCA and SVD dimension-reduction methods for the LDF. For the QDF, however, we saw a small performance decrease for the SVD instead of PCA; that is, the minimum EEER for ordering the principal components via eigenvectors was 0.02

less for PCA than for the SVD. However, we remark that the EEERs for the QDF were larger without exception than the corresponding EEERs for the LDF. Also, the eigenvalue-based ordering scheme yielded the smallest EEER for both LDA and QDA. Overall, the EEERs of the eigenvalue ordering schemes with the LDF were within a range of 0.007 for both the SVD and PCA, but ordering methods had a stronger effect on EEER for the QDF. For this data set, dimension reduction with the SVD yielded arguably non-inferior EEERs to dimension reduction with PCA, while completing calculations 2 times faster and requiring 2.8 times less memory. For the data set of van't Veer et al. (2002), we concluded that classifier performance was essentially equivalent between the computationally-superior SVD and PCA for feature-dimension reduction.

3.7 Discussion

In this paper, we have established notation for the data sets, have given definitions for the linear and quadratic discriminant functions, and have defined the PCA and SVD dimension-reduction methods. Further, we have described three alternatives to ordering the eigenvectors by their eigenvalues, such as canonical variates ordering. Additionally, we have discussed the condition numbers of PCA and SVD and have also presented some benchmark results on the time and memory costs of these two routines, thoroughly demonstrating the increased computational cost associated with PCA when compared to the SVD. We have also presented the EEERs for six real data cases and have discussed their behavior under four eigenvector ordering schemes and two statistical discriminant functions.

Additionally, we have shown that the EEER behavior for the linear and quadratic discriminant functions are essentially equivalent when reducing the data dimension via PCA or the SVD for the six high-dimensional data sets we considered. While the EEERs for these two dimension reduction methods are essentially equal, the computational costs and allocated memory associated with PCA makes this linear

dimension-reduction technique prohibitively expensive and even impossible for certain high-dimensional data sets. Overall, we recommend the use of the SVD to reduce the feature space of high-dimensional data sets before classification to decrease computational costs while preserving misclassification rates, instead of using dimension reduction with PCA or a combination of PCA with gene selection.

Furthermore, we have shown that while eigenvector ordering schemes take into account group information, their positive effects on the EEERs are often minimal at best, and negligible at worst. Eigenvector reordering methods based on the canonical variates or heteroscedastic canonical variates criteria perform similarly to ordering eigenvectors by their eigenvalues alone for the six real data sets considered in this paper. However, both of these ordering schemes require the calculation of one or more group covariance or sums-of-squares matrices, which significantly increases the computational cost. Because of these considerations, we can not recommend employing eigenvector reordering techniques for the high-dimensional cases we considered.

CHAPTER FOUR

Multi-State Multivariate Statistical Process Control

ABSTRACT

Even though principal component analysis (PCA) is not optimal for autocorrelated, nonlinear, and non-stationary data, adaptive-dynamic PCA (AD-PCA) has been shown to do as well as or better than nonlinear dimension reduction methods in flagging outliers in such environments. In some engineered systems, additional designed features create a known multi-state switching scheme among multiple autocorrelated, non-linear, and non-stationary processes, and incorporating this additional known information into AD-PCA can further improve it. In simulations with one of three types of faults introduced, we compare accounting for the states versus ignoring them. We find that multi-state AD-PCA reduces the proportion of false alarms and reduces the average time to fault detection. Conversely, we also investigate the impact of assuming multiple states when only one exists, and find that as long as the number of observations is sufficient, this misspecification is not detrimental. We then apply multi-state AD-PCA to real-world data collected from a decentralized wastewater treatment (WWT) system during in control (IC) and out of control (OoC) conditions. Multi-state AD-PCA flags a strong system fault earlier and more consistently than its single-state competitor. Furthermore, accounting for the physical switching system does not increase the number of false alarms when the process is IC and may ultimately assist with fault attribution.

4.1 Introduction

In the past two decades, online system monitoring and statistical process control (SPC) of industrial systems has expanded rapidly due to increased personnel and en-

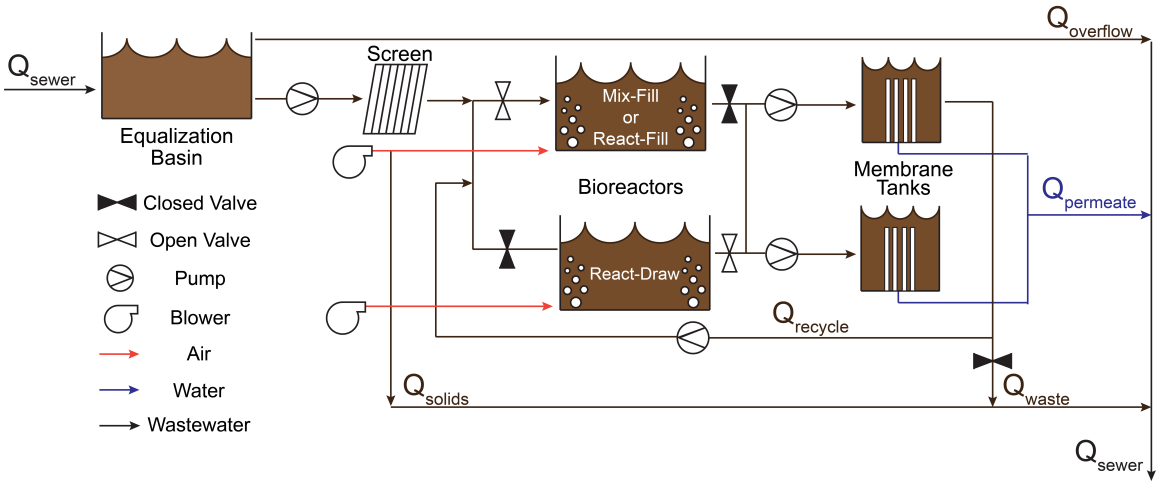
ergy costs and questions of environmental impact (Stavropoulos, Chantzis, Doukas, Papacharalampopoulos, & Chryssolouris, 2013). One such area of application is in the monitoring of centralized and decentralized WWT operations. Centralized treatment plants are often placed at the geographical location that maximizes gravity wastewater conveyance, but most water treatment systems still require extensive pumping operations (EPA, 2000). Decentralized plants are growing in popularity because they can reduce the high costs necessary for water-system infrastructure (Leverenz & Asano, 2011). Additionally, smaller, decentralized facilities can more flexibly respond to local demand for water reuse as well as potentially reduce energy demand within the region (Gikas & Tchobanoglous, 2009; Vuono, Henkel, Benecke, Cath, Reid, Johnson, & Drewes, 2013).

Because of the decentralized locations of these treatment plants, online SPC from a central, off-site location or by on-call staff is often the only cost-effective strategy to ensure that satellite facilities operate within established parameters. These systems generate data that are non-linear, non-stationary, autocorrelated, and multivariate. In an initial monitoring study, Kazor, Holloway, Cath, & Hering (2016) compared an adaptive-dynamic modification of principal component analysis (PCA) (Wold, Esbensen, & Geladi, 1987), abbreviated AD-PCA, with non-linear dimension reduction approaches. They found that AD-PCA yielded either similar or better monitoring results than adaptive, dynamic versions of kernel PCA (Choi & Lee, 2004; Chouaib, Mohamed-Faouzi, & Messaoud, 2013; Scholkopf, Smola, & Muller, 1998) and locally linear embedding (Miao, Song, Ge, Zhou, & Wen, 2013; Zhang, Liu, Qi, & Jiang, 2013). Additional comparisons could still be made between AD-PCA and nonlinear dimension reduction techniques such as *IsoMap* (Tenenbaum, de Silva, & Langford, 2000), Laplacian eigenmaps (Belkin & Niyogi, 2003), ICA (Lee, Yoo, & Lee, 2004), semidefinite embedding (Weinberger & Saul, 2006), spectral embedding (Bengio, Delalleau, Le Roux, Paiement, Vincent, & Ouimet, 2004), multidimensional

scaling (Torgerson, 1958), or spectral multidimensional scaling (Aflalo & Kimmel, 2013). However, a PCA-based approach remains an interpretable and accessible solution that can be implemented with minimal computational cost.

Many adaptations to PCA have been proposed. Gertler, Li, Huang, & McAvoy (1999) discusses monitoring PCA residuals, and Kano, Nagao, Hasebe, Hashimoto, Ohno, Strauss, & Bakshi (2000) compare Moving PCA, wavelet-based Multi-scale PCA, and a dissimilarity scale (DISSIM). Dynamic PCA was employed by Garcia-Munoz, Kourti, & MacGregor (2004) to detect process faults by comparing process observations to their predictions, and Wang & He (2010) augment PCA with statistical pattern analysis.

Figure 4.1: Process flow diagram of the decentralized SB-MBR WWT pilot at Mines Park. Influent from the municipal sanitary sewer is diverted to a 2,500 gallon equalization tank and filtered through a 2-mm drum screen to remove large solids before addition to the bioreactors. The bioreactors are dosed sequentially: the first bioreactor is fed influent while the second recirculates through the membrane tanks. The membranes are hollow-fiber, ultrafiltration membranes with a nominal pore size of 0.04 μm and a total surface area of 74 m^2.



However, none of these authors applied these techniques to SPC for WWT. One of the early applications of PCA to WWT process monitoring was by Wise, Veltkamp, Davis, Ricker, & Kowalski (1988), and PCA was also applied to the similar task

of chemical systems monitoring by Kresta, MacGregor, & Marlin (1991). Baggiani & Marsili-Libelli (2009) discuss implementing adaptive PCA to detect faults in a large-scale, centralized WWT plant by monitoring three process variables. Sanchez-Fernandez, Fuente, & Sainz-Palmero (2015) dissected the fault detection problem in a large-scale, centralized WWT plant by employment of distributed PCA to monitor segments within the plant as a sum of parts rather than the whole. Kazor et al. (2016) focused exclusively on a decentralized WWT facility monitoring nearly 30 variables.

4.1.1 Motivating Example

In this paper, we study multi-state PCA in which PCA or one of its variations is independently applied to subsets of observations. These states can occur in many processes when an operator controls, for example, a physical mechanism in the system. While one could consider any number of states, we illustrate the concept with three states. Figure 4.1 shows a schematic of a decentralized WWT facility with a hybrid sequencing batch reactor (SBR) and membrane bioreactor (MBR), known as a sequencing batch membrane bioreactor (SB-MBR). The system is partitioned, and a blower for each MBR mixes the pre-screened wastewater. The operation of a blower in MBR 1 has no direct effect on MBR 2, and vice versa. At any given time, the system is in one of three states: neither blower on ($\mathcal{S}_0$); blower 1 one on ($\mathcal{S}_1$); and blower 2 on ($\mathcal{S}_2$). The operator can control the operation and duration of each blower. Figure 4.2 shows the values of a few monitored variables over time, and it is clear that a blower being off or on will change the dissolved oxygen and blower flow rates dramatically. In addition, two heat maps are shown in Figure 4.3 where the differences in correlation matrices between states ($\mathcal{S}_1$) versus ($\mathcal{S}_1$) and ($\mathcal{S}_0$) versus ($\mathcal{S}_2$) are given. The pairwise correlation between variables can also change substantially between different states.

As a simple example to show why blocking the multivariate process model on these known system states is important, assume that blower 1 malfunctions. Because

Figure 4.2: Each panel shows a time series of a monitored variable. The values of each variable change drastically with blower operation.

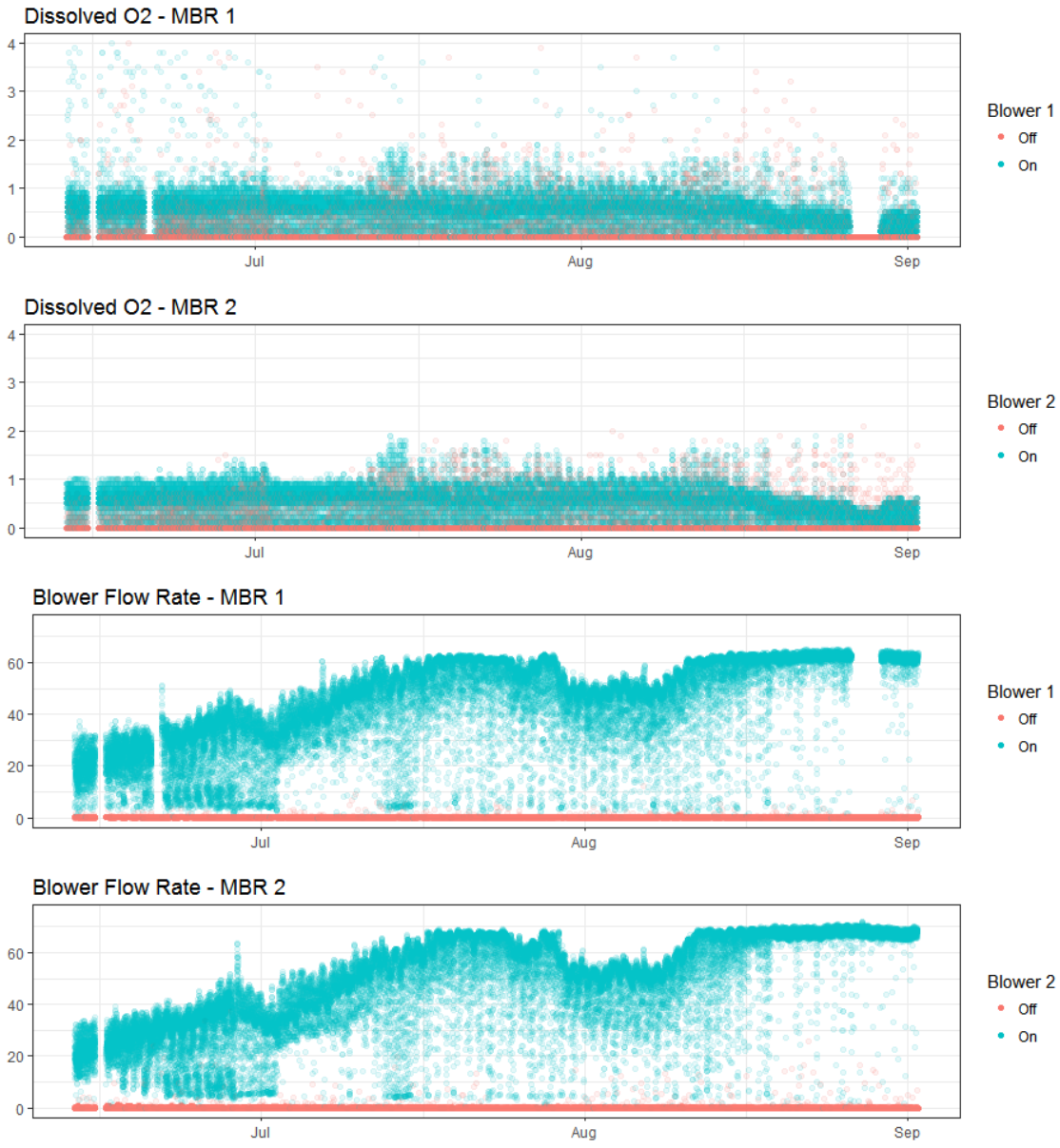
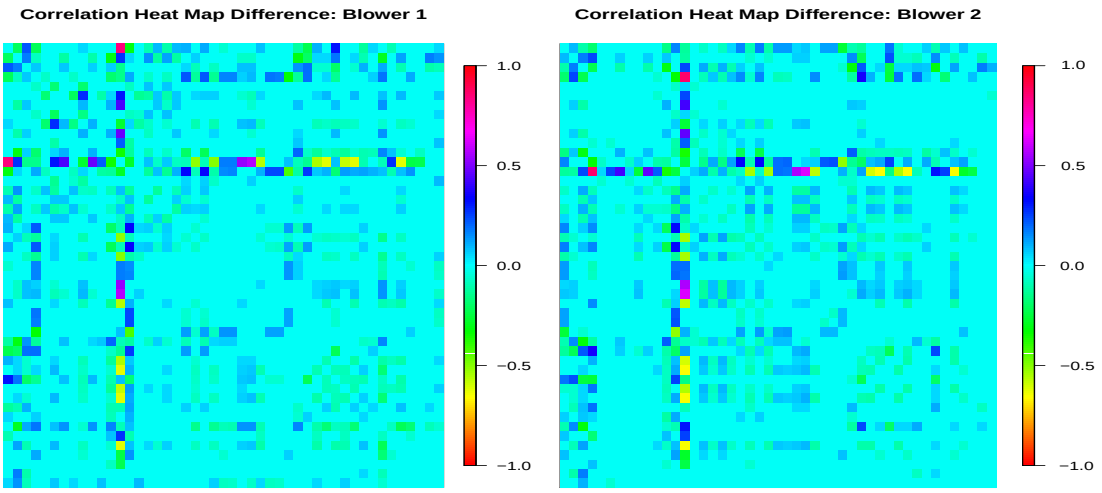


Figure 4.3: Differences in correlation matrices between $\mathcal{S}_0$ and $\mathcal{S}_1$ (*left*) and $\mathcal{S}_0$ and $\mathcal{S}_2$ (*right*).



of this malfunction, the dissolved oxygen values in this state would fall towards 0. If state information is ignored, these lower values would not be considered unusual because the dissolved oxygen values are often 0 when both blowers are off. However, with state information, a multi-state monitoring method could correctly identify that the blower is, in fact, malfunctioning.

4.1.2 Multi-State Versus Multi-Stage Monitoring

A similar approach to multi-*state* SPC is multi-*stage* SPC, which accounts for different stops along an ordered manufacturing process or assembly line. Jearkpaporn, Borror, Runger, & Montgomery (2007) accounted for the differences in feature means across different stages in a multivariate production process, where each stage directly influences the following stages. Asadzadeh, Aghaie, & Yang (2008) built on this with their work on Cause Selecting Control Charts, but these methods do not account for changes in variances of features or correlation among features across stages. For a discussion of multivariate SPC (MVSPC) over sequential stages, see Li & Tsung (2012) and the references therein. A restriction of the multi-stage assumption is that

the process must follow the same steps in order, and stages never deviate from the prescribed order. However, states in multi-state monitoring do not have a required order and often depend entirely on operator input.

In this paper, we build on the work of Kazor et al. (2016), by constructing a modification to AD-PCA that accounts for different known and controlled process states. We will show that monitoring non-linear, non-stationary, and serially auto-correlated multi-state multivariate processes by pairing the Squared Prediction Error (SPE) and Hotelling's T^2 (T^2) monitoring statistics after dimension reduction via our new multi-state adaptive-dynamic principal components analysis (MSAD-PCA) is superior to AD-PCA. We show in a simulation study, and verify with a real case study on a decentralized WWT plant, that the SPE monitoring statistic after MSAD-PCA dimension reduction offers the best combination of sensitivity and high detection proportion while the T^2 monitoring statistic after MSAD-PCA dimension reduction offers the best combination of specificity and high detection proportion.

This paper is organized as follows: in Section 4.2, the basic notation for PCA is presented along with a description of MSAD-PCA. In Section 4.3, a simulation study to test the utility and reliability of the new method is described. Section 4.4 presents the simulation results with average false alarm rates, detection probabilities, and detection times for nine different faults introduced into the multi-state simulation design. In Section 4.5, we return to the real scenario that motivates the need for multi-state monitoring and demonstrate the added benefit to correctly monitoring a multi-state process with multi-state AD-PCA. In Section 4.6, we offer concluding remarks, discuss future research directions, and mention useful software for applying multi-state or single-state AD-PCA.

First, let $\mathcal{P}$ represent a real p-dimensional multivariate process. Let $s = 1, \ldots, n$ be the observation index of process $\mathcal{P}$, and further let $X_i(s)$ be the observed value for feature $i = 1, \ldots, p < n$ at index s. Then, $\mathbf{X}_s = (X_1(s), X_2(s), \ldots, X_p(s)) \in \mathbb{R}_{1\times p}$ is a standardized realization from the multivariate stochastic process $\mathcal{P}$ at index s.

When the process dimension p is large, variables are often noisy and dependent. One widely accepted technique to reduce the data dimension and produce linearly independent variables is PCA (Johnson & Wichern, 2002). Briefly, PCA is an orthogonal linear transformation mapping the existing data matrix, $\mathbf{X}$, defined as

$$\mathbf{X} := [\mathbf{X}_1 \vdots \mathbf{X}_2 \vdots \cdots \vdots \mathbf{X}_n]^T \in \mathbb{R}_{n\times p}, \tag{4.1}$$

onto a new coordinate system wherein the greatest source of the variability of $\mathbf{X}$ lies in the first coordinate of the new system, the second-greatest source lies in the second coordinate, and so forth.

To do this, the *singular value decomposition* (Eckart & Young, 1936) can be applied to the mean-centered data matrix

$$\underset{n\times p}{\mathbf{X}} := \underset{n\times n}{\mathbf{U}} \times \underset{n\times p}{\mathbf{D}} \times \underset{p\times p}{\mathbf{P}}^T, \tag{4.2}$$

where $\mathbf{U}$ is the matrix of n orthonormal basis vectors $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$, or left singular vectors, and $\mathbf{P}$ is the $p\times p$ orthogonal matrix of right singular vectors, $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_p\}$. These singular vectors correspond to the eigenvalues in the rectangular block matrix $\mathbf{D}$, defined as

$$\mathbf{D}_{n\times p} := \begin{bmatrix} \mathbf{\Lambda}_{p\times p} \\ \mathbf{0}_{(n-p)\times p} \end{bmatrix},$$

where $\mathbf{\Lambda}$ is the diagonal matrix of the p eigenvalues.

To reduce the data dimension from p to d, for $d < p$, let

$$\mathbf{P}_d := [\mathbf{v}_1 \vdots \mathbf{v}_2 \vdots \cdots \vdots \mathbf{v}_d] \tag{4.3}$$

be the $p \times d$ projection matrix comprised of the d eigenvectors associated with the d largest eigenvalues. Then, post-multiply $\mathbf{X}$ by $\mathbf{P}_d$ to obtain the data matrix approximated in its d-dimensional principal component subspace, as follows:

$$\mathbf{Y}_d = \mathbf{UD} \approx \mathbf{UDP}^T\mathbf{P}_d = \mathbf{XP}_d. \tag{4.4}$$

Because $\mathbf{P}$ is orthogonal,

$$\lim_{d \to p} ||\mathbf{UDP}^T\mathbf{P}_d - \mathbf{UD}|| = 0.$$

4.2.1 Adaptive-Dynamic PCA

Kazor et al. (2016) applied dimension reduction prior to performing multivariate statistical process control to reduce variability of the process monitoring statistics. An adaptive-dynamic variation of PCA was found to account for inherent non-linearity, non-stationarity, and autocorrelation in simulated data. For a good survey of accounting for non-linearity, non-stationarity, or autocorrelation in process data, see Ge & Song (2012). AD-PCA is the concurrent application of both the adaptive and dynamic modifications discussed below to the PCA of process data.

4.2.1.1 *Adaptive PCA.* The adaptive modification to PCA creates a rolling training window over which to estimate a reduced data matrix $\mathbf{Y}_d$, in order to mitigate the effects of non-linearity and non-stationarity. Let $\mathbf{X}$, as defined in (4.1), be a realization from a p-dimensional non-linear, non-stationary stochastic process $\mathcal{P}$. Further, let $\mathbf{X}$ be locally linear for a w-width discrete-time neighborhood of observation s. We assume that within this w-width neighborhood, the process $\mathbf{X}$ can be approximated by a d-dimensional linear process, for $d < p$. Thus, we estimate the $p \times d$ projection matrix $\mathbf{P}_d$ by taking the SVD of w observations,

$$[\mathbf{X}_{s-w+1} \vdots \mathbf{X}_{s-w+2} \vdots \cdots \vdots \mathbf{X}_{s-1} \vdots \mathbf{X}_s]^T,$$

rather than of the full data matrix $\mathbf{X}$. We then project the next n_u $1 \times p$ observation from p to d dimensions by right-multiplying each by $\mathbf{P}_d$, yielding $1 \times d$ observations $\mathbf{Y}_{s+1}, \mathbf{Y}_{s+2}, \ldots, \mathbf{Y}_{s+n_u}$, where n_u is the number of observations to monitor before moving the window forward and re-estimating $\mathbf{P}_d$. Once these reduced-dimension observations are constructed, any number of tests can be performed in this reduced-dimension feature space.

After the next set of observations, $\mathbf{X}_{s+n_u+1}, \ldots, \mathbf{X}_{s+2n_u}$, is recorded, the oldest n_u observations, $\mathbf{X}_{s-w+1}, \mathbf{X}_{s-w+2}, \ldots, \mathbf{X}_{s-w+n_u}$, are removed from the training set; the newest n_u observations, $\mathbf{X}_{s+1}, \mathbf{X}_{s+2}, \ldots, \mathbf{X}_{s+n_u}$, are added to the training set; and the above dimension reduction process is repeated. At each step in the adaptive process, the number of training observations is fixed at w.

As an example, consider a multivariate process, $\mathbf{X}_s$, recorded hourly, that is known to be non-linear or non-stationary. For short subsets of consecutive observations, $\mathbf{X}$ may be locally linear. Thus, we could estimate the $p \times d$ projection matrix $\mathbf{P}_d$ using a week's worth of data points, setting $w = 24 \times 7 = 168$. From this first week of observations, we reduce the data dimension for the eighth day of the process, and we perform tests on the set of 24 reduced observations $\{\mathbf{Y}_{169}, \mathbf{Y}_{170}, \ldots, \mathbf{Y}_{192}\}$. After we have finished testing or classifying the reduced forms of $\{\mathbf{X}_{169}, \mathbf{X}_{170}, \ldots, \mathbf{X}_{192}\}$, we remove the first 24 observations from $\mathbf{X}$, replace them with the last 24, and estimate a new projection matrix $\mathbf{P}_d$ to reduce the data dimension for observations from day nine. This process is then repeated until the end of the data is reached.

4.2.1.2 *Dynamic PCA.* If $\mathbf{X}$ defined in (4.1) is a realization from a p-dimensional autocorrelated stochastic process, then a dynamic modification to PCA can account for autocorrelation in this process. Assume a feature $X_i(s)$ depends upon $X_i(s-\ell)$, for some lag $\ell \in \mathbb{N}$. These lagged features are column-concatenated onto the data matrix $\mathbf{X}$. In the most general case, all lags up to lag ℓ for each feature would be

included. Row s, a $1 \times (p + p\ell)$-dimensional vector, of this data matrix is then defined as

$$\begin{aligned} \mathrm{L}(\mathbf{X}_s) := [&X_1(s), X_1(s-1), \ldots, X_1(s-\ell), \\ &X_2(s), X_2(s-1), \ldots, X_2(s-\ell), \\ &\cdots, \\ &X_p(s), X_p(s-1), \ldots, X_p(s-\ell)]. \end{aligned} \tag{4.5}$$

Combining $n - \ell$ rows, the lagged data matrix is formed as

$$\mathbf{X} = [\mathrm{L}(\mathbf{X}_\ell) \vdots \mathrm{L}(\mathbf{X}_{\ell+1}) \vdots \cdots \vdots \mathrm{L}(\mathbf{X}_n)]^T \in \mathbb{R}_{(n-\ell)\times(p+p\ell)}. \tag{4.6}$$

To reduce the dimension of observation $\mathrm{L}(\mathbf{X}_{s+1})$, we estimate the $(p + p\ell) \times d$ projection matrix $\mathbf{P}_d$ by taking the SVD of (4.6). To curb the rapid dimensional growth possible in (4.6), Rato & Reis (2013) advocate including only a few lagged features per original feature. To this end, Kazor et al. (2016) define

$$\mathrm{L}^{\boldsymbol{\ell}}(\mathbf{X}_s) := [X_1(s), X_1(s-\ell_1), X_2(s), X_2(s-\ell_2), \ldots, X_p(s), X_p(s-\ell_p)], \tag{4.7}$$

where $\boldsymbol{\ell} = (\ell_1, \ell_2, \ldots, \ell_p)$ is the vector of maximally autocorrelated lags for each feature $i = 1, \ldots, p$. This lag feature selection criterion yields a data matrix $\mathbf{X}$ with $n - \max\{\boldsymbol{\ell}\}$ rows and $2p$ columns. Thus, our projection matrix $\mathbf{P}_d$ has dimension $2p \times d$. This doubles the number of features and consequently the number of rows of the projection matrix $\mathbf{P}_d$. However, these new features are maximally correlated with the existing features, so the number of retained principle components, d, is not expected to double.

4.2.2 Multi-State Adaptive-Dynamic PCA

We now consider how known dichotomous states affect system behavior. MSAD-PCA accounts for the multi-state structure of the process while still incorporating the adaptive and dynamic PCA modification. When the relationships among features change

as the state changes, the projection matrix will have different linear combinations of features under different states.

Let $\mathbf{X}_s = (X_1(s), X_2(s), \ldots, X_p(s))$ be a standardized realization from some unknown real-valued p-dimensional multivariate stochastic process recorded over index s, where $X_i(s)$ denotes the realization of the i^{th} feature at index s. Given a set, $\mathcal{S}$, of K mutually exclusive process states, $\mathcal{S}_1, \ldots, \mathcal{S}_K$, we assign corresponding pairwise disjoint sets of indices $\mathcal{T}_1, \ldots, \mathcal{T}_K$ such that $\bigcup_{k=1}^{K} \mathcal{T}_k = \{1, \ldots, n\}$ and $n_k = \mathcal{C}(\mathcal{T}_k)$, where $\mathcal{C}(\cdot)$ denotes the cardinality of a set. More specifically, $s^* \in \mathcal{T}_k \iff \mathbf{X}_{s^*}$ was observed under state $\mathcal{S}_k$.

We now define an observation from a multi-state p-dimensional stochastic process. Let $f_k : \mathbb{R}_p \to \mathbb{R}_p$ be some measurable function describing the process under state k with mean $\boldsymbol{\mu}_k$ and finite covariance matrix $\boldsymbol{\Sigma}_k$, for $k = 1, \ldots, K$. Then

$$\mathbf{X}_{s^*} \sim f_k(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \iff s^* \in \mathcal{T}_k. \tag{4.8}$$

Naturally, in order to find the feature subspace maximizing the principal component loadings, the observations from each process state should be treated differently. Therefore, for states $\mathcal{S}_1, \ldots, \mathcal{S}_K$, we partition $\mathbf{X}$ as $\left\{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(K)}\right\}$, where

$$\mathbf{X}^{(k)} := \left[\mathbf{X}_{\min(\mathcal{T}_k)}, \ldots, \mathbf{X}_{\max(\mathcal{T}_k)}\right]^T$$

is the set of all observations from the process under state k, ordered in s. Appropriate lags can then be included and state-specific projection matrices calculated along a rolling window for a full MSAD-PCA approach.

In summary, MSAD-PCA splits the observations by state and estimates state-specific projection matrices that update over a rolling window of training observations. However, one should exercise caution when including state information. For K equally-sized states, MSAD-PCA will partition the observations into K groups, effectively dividing the total sample size by K. Therefore, the number of states should be kept as small as possible. We return to this point at the end of Section 4.4.1.

4.2.3 Process Monitoring Statistics

After we have calculated the $2p \times d$ linear projection matrix $\mathbf{P}_d$, trained from the observations $\max_i\{\ell\}, \ldots, s$, we calculate two process monitoring statistics of the new observation $\mathbf{X}_{s+1}$. The statistics we choose are the Hotelling's T^2 and squared prediction error (*SPE*). We recognize the uses of the Apley & Shi (1999) generalized likelihood ratio test (GLRT) statistic as applied to SPC of a univariate autocorrelated process, but extensions of this test to MVSPC do not seem to be used frequently in practice, so we do not employ the GLRT statistic at this time.

4.2.3.1 *Hotelling's T^2.* The T^2 statistic is calculated as

$$T^2_{s+1} = \mathbf{Y}_{s+1}\mathbf{\Lambda}_d^{-1}\mathbf{Y}_{s+1}^T, \tag{4.9}$$

where $\mathbf{\Lambda}_d = \text{diag}\,(\lambda_1, \lambda_2, \ldots, \lambda_d)$, and λ_i is the i^{th} eigenvalue. Thus, T^2 is the Mahalanobis distance of the mapped value $\mathbf{Y}_{s+1}$ from the original space into the PCA-subspace, and it measures deviations in the lower d-dimensional subspace. Because this subspace should characterize the major components of the process under IC conditions, outliers identified by large T^2 values are often indicative of OoC conditions.

4.2.3.2 *Squared Prediction Error.* The *SPE* statistic is calculated as

$$\text{SPE}_{s+1} = \left(\mathbf{X}_{s+1} - \mathbf{Y}_{s+1}\mathbf{P}_d^T\right)\left(\mathbf{X}_{s+1} - \mathbf{Y}_{s+1}\mathbf{P}_d^T\right)^T. \tag{4.10}$$

The *SPE* statistic measures the squared distance between the original and reduced-dimension vectors, measuring the goodness-of-fit of the d-dimensional model to the p-dimensional process observations, or how well $\mathbf{P}_d$ approximates $\mathbf{P}$. Abnormal values of *SPE* can indicate either an unusual observation or that the lower-dimensional model does not account for an important component of process variability.

4.2.3.3 *Threshold estimation.* It is commonplace to use parametric distributions, such as the χ^2 or F-distribution, to compute thresholds for the above process

monitoring statistics, but when the underlying distribution of the data is not normal, these thresholds are not reliable (Qiu, 2013). To avoid making any distributional assumptions about T^2, Kazor et al. (2016) calculated non-parametric thresholds based on observations in the IC training window for the *SPE* and T^2 monitoring statistics. We follow their work and compute a non-parametric threshold for T^2 from the values of T^2 computed in the training data via kernel density estimation with a Sheather-Jones bandwidth (Sheather, 2009) and a Gaussian kernel. A threshold for *SPE* is computed similarly. For further discussion of non-parametric SPC, see Qiu & Li (2012). For non-parametric MVSPC, see Chen, Zi, & Zou (2016). Additional modifications for multivariate autocorrelated observations could be extended from univariate approaches (Apley & Lee, 2012; Apley & Tsung, 2002; Lee & Apley, 2011).

4.3 Simulation Design

A simulation study is constructed to illustrate the fault-detection behavior of MSAD-PCA compared to ordinary AD-PCA. Specifically, performance is measured on three scales: (1) the average time-to-detection for true system faults, (2) the proportion of faults detected, and (3) the average false alarm rates for IC conditions. Lower values of (1) and (3) and higher values of (2) are desirable.

4.3.1 Simulation Notation

Consider an $n \times p$ multivariate process indexed over s as defined in Section 4.2.2. In order to account for state differences, X_i is piecewise-defined over the state boundaries and smooth within these boundaries. To simulate non-linearity and non-stationary in each feature, let $X_i(s) := X_i(t_s)$ be a function of a latent feature t, where $t := g(s)$. Further, restrict g to be a smooth and bounded function with $g'(s) \neq 0$ for some s. To induce autocorrelation within t, draw the errors of t, ϵ_t, from an autoregressive process over s. Combining these conditions, we have the form of a univariate, non-stationary,

and autocorrelated latent variable

$$t_s := g(s) + \epsilon_s, \quad \epsilon_s = \varphi\epsilon_{s-1} + \varepsilon,$$

where $\varepsilon \sim N(0, \sigma_\varepsilon)$. The full non-linear, autocorrelated, non-stationary process at index s is denoted

$$\mathbf{X}_s := \langle X_1(t_s), X_2(t_s), \ldots, X_p(t_s) \rangle .$$

4.3.2 Data Generation

Based on the study of Dong & McAvoy (1996), we set $p = 3$ so that

$$\mathbf{X}_s = \langle x(t_s), y(t_s), z(t_s) \rangle .$$

Also, let the cycle length $\omega = 60 * 24 * 7 = 10,080$, and define the observation index $s = 1, \ldots, \omega$, which corresponds to one observation per minute for one week. Further, we set the autocorrelation dependence parameter φ to 0.75. The data is generated as follows:

(1) Initialize the first innovation

$$\varepsilon_1 \sim \mathcal{N}\left(\frac{1}{2}(a+b)(1-\varphi), \frac{b-a}{12}(1-\varphi^2)\right), \tag{4.11}$$

where $a = 0.01$ and $b = 2$. These mean and variance multipliers are from the mean and variance of the uniform(a, b) distribution.

(2) Define the first-order autoregressive error structure

$$\epsilon_s := \varphi\epsilon_{s-1} + (1-\varphi)\varepsilon, \quad s = 2, 3, \ldots, \tag{4.12}$$

where ε is defined in (4.11).

(3) Define the latent-feature vector

$$t_s := -\cos\left(\frac{2\pi}{\omega}s\right) + \varepsilon_s, \quad s = 1, \ldots, \omega \tag{4.13}$$

where ε_s is the autoregressive process defined in (4.12). Notice that the mean of this latent feature depends upon the index, which allows the process to be non-stationary.

(4) Scale the latent feature vector t_s by

$$\tilde{t}_s := \frac{(b-a)(t_s - \min(t_s))}{\max(t_s) - \min(t_s)} + a,$$

where a and b are defined in step 1.

(5) Draw three machine-error vectors, each of length ω: $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \overset{i.i.d.}{\sim} \mathcal{N}(0, 0.01)$.

(6) The three features are constructed as follows:

$$\begin{aligned} x(t_s) &:= t_s + \mathbf{e}_1(s) \\ y(t_s) &:= t_s^2 - 3t_s + \mathbf{e}_2(s) \\ z(t_s) &:= -t_s^3 + 3t_s^2 + \mathbf{e}_3(s). \end{aligned} \tag{4.14}$$

(7) **Optional:** Induce a state-independent fault (Fault 1A, 1B, 2A, 2B, 3A, or 3B) as described in Section 4.3.3. (Note: apply this step only if you will not apply Step 10.)

(8) The system states are defined as:

$\mathcal{S}_1$: $\mathbf{X}(t_s) := \langle x(t_s), y(t_s), z(t_s) \rangle$.

$\mathcal{S}_2$: $\mathbf{X}(t_s) := \langle x(t_s), y(t_s), z(t_s) \rangle \cdot \mathbf{P}_1 \mathbf{\Lambda}_1$, where

$$\mathbf{P}_1 = \begin{bmatrix} 0 & 0.50 & -0.87 \\ 0 & 0.87 & 0.50 \\ 1 & 0 & 0 \end{bmatrix}$$

is the orthogonal rotation matrix for a yaw, pitch, and roll degree change of $\langle 0°, 90°, 30° \rangle$, and $\mathbf{\Lambda}_1 = \text{diag}(1, 0.5, 2)$ is a diagonal scaling matrix.

$\mathcal{S}_3$: $\mathbf{X}(t_s) := \langle x(t_s), y(t_s), z(t_s) \rangle \cdot \mathbf{P}_2 \mathbf{\Lambda}_2$, where

$$\mathbf{P}_2 = \begin{bmatrix} 0 & 0.87 & -0.50 \\ -1 & 0 & 0 \\ 0 & 0.50 & 0.87 \end{bmatrix}$$

is the orthogonal rotation matrix for a yaw, pitch, and roll degree change of $\langle 90°, 0°, -30° \rangle$, and $\mathbf{\Lambda}_2 = \text{diag}(0.25, 0.1, 0.75)$ is a diagonal scaling matrix.

The rotation matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ turn the states in three-dimensional space so that the states are at right angles to each other in at least one dimension, and the scaling matrices $\mathbf{\Lambda}_1$ and $\mathbf{\Lambda}_2$ inflate or deflate the feature variances along each principal component.

(9) Single-State or Multi-State options:

(a) For data generation under a single-state model, generate all observations from $\mathcal{S}_1$.

(b) For data generation under a multi-state model, define the state-switching scheme corresponding to a *known* hourly process schedule as follows:

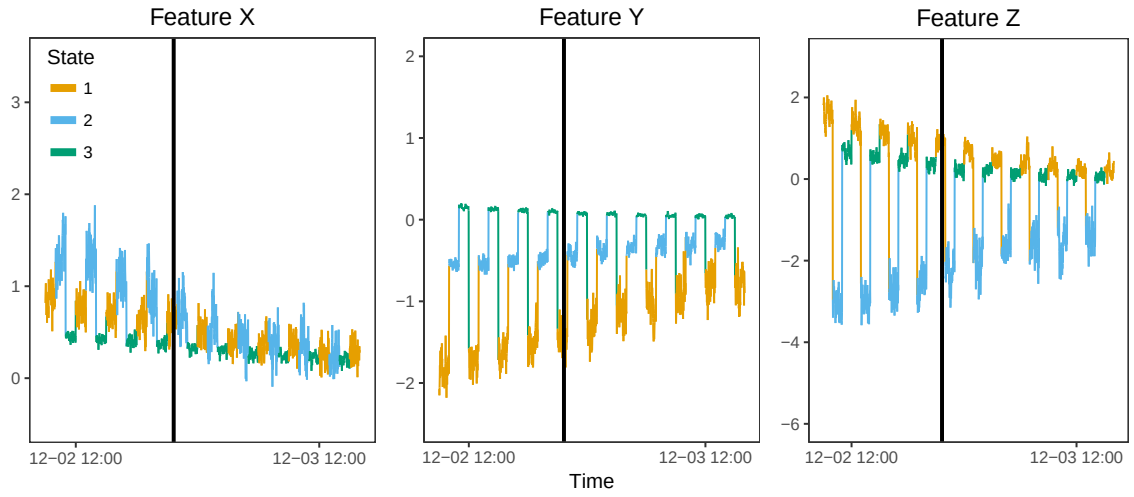
$$f(s) := \begin{cases} \mathcal{S}_1, & s \bmod (60 * 3) = 0, \\ \mathcal{S}_2, & s \bmod (60 * 3) = 1, \\ \mathcal{S}_3, & s \bmod (60 * 3) = 2. \end{cases}$$

The state switches each hour from $\mathcal{S}_1$ to $\mathcal{S}_2$ to $\mathcal{S}_3$ in a three-hour cycle before returning to $\mathcal{S}_1$.

(10) **Optional:** Induce a state-specific fault (Fault 1C, 2C, or 3C) as described in Section 4.3.3. (Note: apply this step only if you have not applied Step 7.)

Figure 4.4 shows the time series of a single random draw of the IC multi-state multivariate process data generated via this simulation scheme. Note that the

Figure 4.4: Multivariate process feature time series under ICC. The vertical black line (21:40 on 2 December) marks the time at which a fault would be induced.



features x, y, and z are represented by the labels X, Y, and Z in Figure 4.4. Time series observations were generated over 1 week, with the arbitrarily chosen time index running from 27 November at 00:00 to 4 December at 00:00. As we can see in this figure, the means and variances for each of the process features change as the state changes. For each feature window, the limits of the vertical axis are constant across each of the generated faults. Moreover, the vertical black line at 21:40 on 2 December marks when a fault is to be introduced. The process responses to the right of the black line show the observations *as they should have been*, if a fault had not been introduced. Finally, unless otherwise specified, faults are introduced *before* the observations are partitioned into the state projections (Step 8).

4.3.3 Fault Scenarios

We introduce one of the following system faults at $s = 8500$, or roughly 85% through the cycle. Table 4.1 shows the types of faults and how these faults affect states and features. The columns show the general type of fault introduced, while the rows show how these faults affect each process feature. The specific faults are as follows:

Fault 1A: $\mathbf{X}^*(\tilde{t}_s) = \mathbf{X}(\tilde{t}_s) + 2,\ s \geq 8500$. This fault is a shift for all features in all states.

Fault 1B: $x^*(\tilde{t}_s) = x^*(\tilde{t}_s) + 2,\ s \geq 8500$. This fault is a shift for one feature in all states.

Fault 1C: $x^*(\tilde{t}_s) = x^*(\tilde{t}_s) + 0.5,\ z^*(\tilde{t}_s) = z^*(\tilde{t}_s) + 0.5,\ s \geq 8500$. This fault simulates a shift in two of the process monitoring features in $\mathcal{S}_3$ only.

Fault 2A: $\mathbf{X}^*(\tilde{t}_s) = \mathbf{X}(\tilde{t}_s) + (s - 8500) \times 10^{-3},\ s > 8500$. This fault simulates a positive drift across all the process monitoring features.

Fault 2B: $y^*(\tilde{t}_s) = y(\tilde{t}_s) + (s - 8500) \times 10^{-3}$ and $z^*(\tilde{t}_s) = z(\tilde{t}_s) + (s - 8500) \times 10^{-3}$, for $s > 8500$. This fault simulates a positive drift in two of the process monitoring features.

Fault 2C: $y^*(\tilde{t}_s) = y(\tilde{t}_s) - 1.5 \times \frac{s - 8500}{10080 - 8500}$, for $s > 8500$. This fault simulates a negative drift in one of the process monitoring features in $\mathcal{S}_2$ only.

Fault 3A: $\mathbf{X}^*(\tilde{t}_s) = \mathbf{X}(\tilde{t}_s^*)$, where $\tilde{t}_s^* = \left[\frac{\delta(s - 8500)}{\omega - 8500} + 1\right] \tilde{t}_s$. For $s > 8500$, this fault will amplify the underlying latent effect for all features. The maximum latent drift of this fault will be $\delta + 1$.

Fault 3B: $z^*(\tilde{t}_s) = z(\tilde{t}_s^*)$, where $\tilde{t}_s^* = \ln|\tilde{t}_s|,\ s \geq 8500$. This fault will dampen the underlying latent effect for x_3 if $\tilde{t}_s > 1$ and amplify this effect if $\tilde{t}_s < 1$.

Fault 3C: $y^*(\tilde{t}_s) = y(\tilde{t}_s) + 2 * \mathbf{e}_2(s) - 0.25$, for $s > 8500$. This fault simulates an error increase and negative shift, but only applied to one feature in $\mathcal{S}_2$.

In summary, the "1" faults are a shift to all features in all states ("A"), one feature in all states ("B"), and two features in one state ("C"). The "2" faults are a slow drift to all features in all states ("A"), two features in all states ("B"), and one feature in one state ("C"). Fault 3A is a mutation of the latent t vector, affecting all features in all states. Fault 3B is a different mutation of the latent t vector, but it affects only one feature in all states. Fault 3C is a shift and amplitude increase to the

Table 4.1: Fault types by how each fault affects different sets of features.

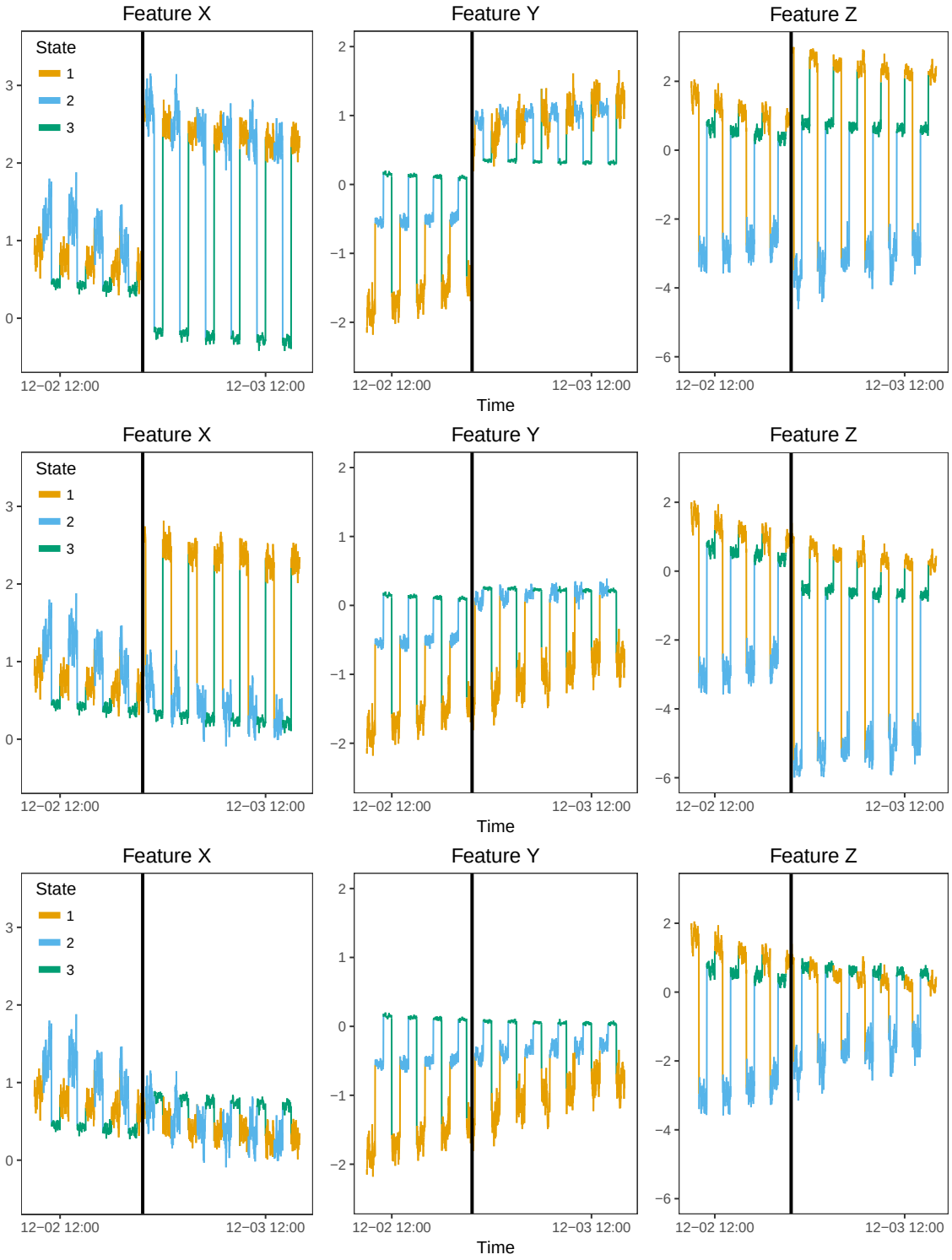
	Type of Fault		
Features Affected	Shift Fault	Drift Fault	Latent / Error Faults
All Features Equally	Fault 1A $(x,\ y,\ z)$	Fault 2A $(x,\ y,\ z)$	Fault 3A $(x,\ y,\ z)$
Each Feature Differently	Fault 1B (x)	Fault 2B $(y,\ z)$	Fault 3B (z)
Each State Differently	Fault 1C $(x,\ z) \in \mathcal{S}_3$	Fault 2C $y \in \mathcal{S}_2$	Fault 3C $y \in \mathcal{S}_2$

error vector of one feature in one state. The "C" faults should showcase differences between AD-PCA and MSAD-PCA.

4.3.3.1 *Shift faults: 1A, 1B, and 1C.* Figure 4.5 shows the effects of the "Shift Faults" column in Table 4.1. The first row exhibits a realization of each of the three features after the introduction of Fault 1A. This fault is simply adding 2 to each feature *before* state projections. This adds a further layer of complexity to the fault as certain projections will affect the positive shift differently. For example, the projection associated with $\mathcal{S}_3$ actually makes this shift *negative* for feature X, and similarly the projection associated with $\mathcal{S}_2$ makes this shift negative for feature Z as well. Overall, however, this positive shift affects each state within each feature, and we expect this shift to be relatively trivial for each of the monitoring statistics of both methods to detect.

The second row of Figure 4.5 exhibits a realization of each of the three features after the introduction of Fault 1B. By definition, this fault is a positive shift to the X feature only, but the state projections will allow this fault to affect the other features as well. Specifically, we see this shift clearly in $\mathcal{S}_1$ of feature X and $\mathcal{S}_2$ of feature Z,

Figure 4.5: Multivariate process feature time series before and after shift faults. Fault 1A (top), Fault 1B (middle), and Fault 1C (bottom) are shown after the vertical black line (21:40 on 2 December).

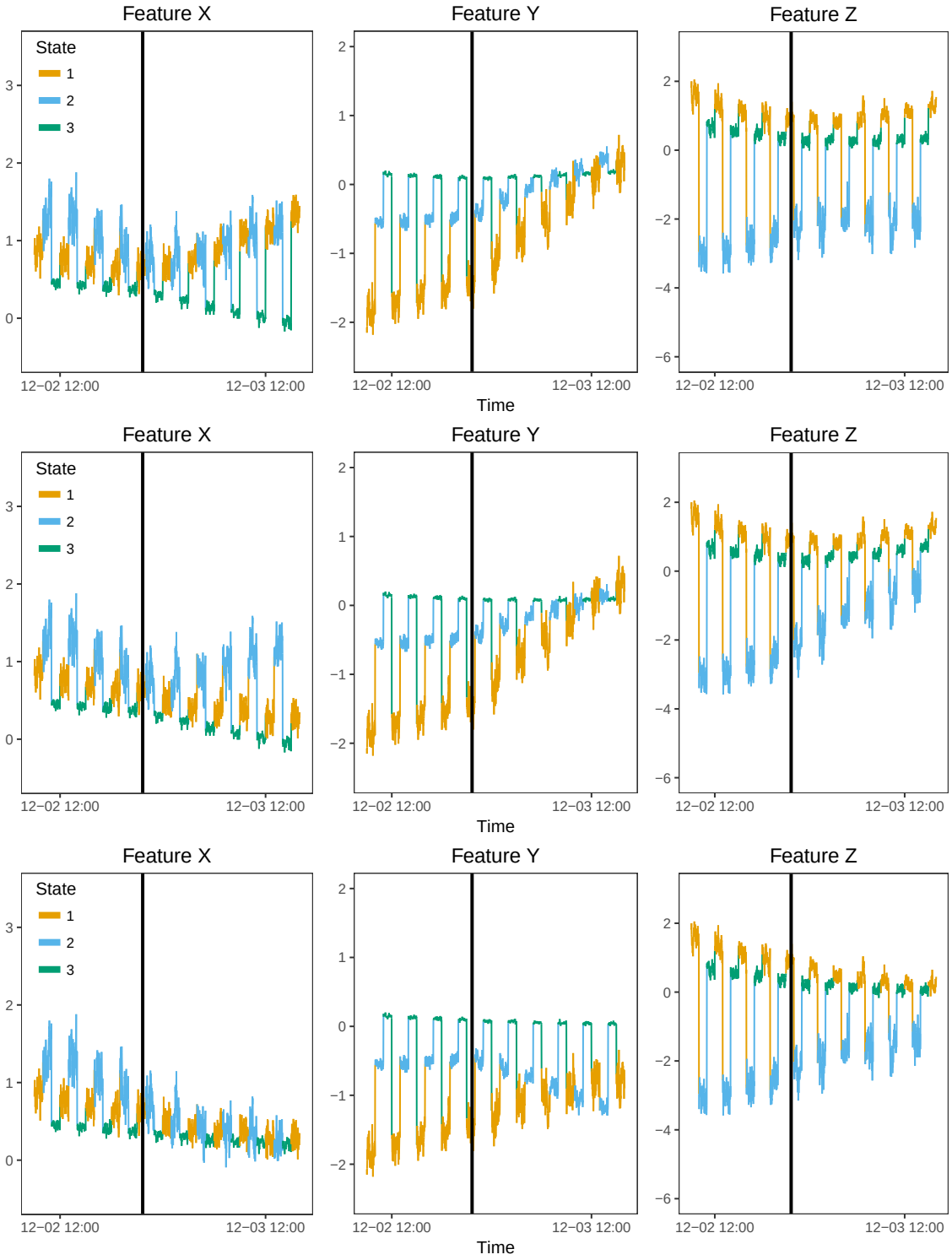


and we see this fault's effect in $\mathcal{S}_2$ of feature Y to a lesser extent. We consider it worth noting that shifting one of these features in a closed but correlated system will have ramifications throughout the entire system, as demonstrated by Fault 1B. Because of the cross-feature fault infection, we also expect Fault 1B to be relatively easy to detect by all four monitoring statistics, but not as easy to detect as Fault 1A.

The last row of Figure 4.5 depicts a realization of each of the three features after the introduction of Fault 1C. By definition, this fault is a positive shift to the X and Z features only, but we constrain this fault to occur only in $\mathcal{S}_3$. In this case, the fault was applied *after* the state partitioning and projection, so the other feature realizations within $\mathcal{S}_1$ and $\mathcal{S}_2$ remain unaffected. We remark that the $\mathcal{S}_3$ realizations of X and Z remain relatively "hidden" within the noise of the other states, so we expect Fault 1C to be the most difficult to detect of the three shift faults, but it should be easier to detect for MSAD-PCA than AD-PCA.

4.3.3.2 *Drift faults: 2A, 2B, and 2C.* Figure 4.6 shows the effects of the "Drift Faults" column in Table 4.1. The first row exhibits a realization of each of the three features after the introduction of Fault 2A. This fault adds a slowly-building drift effect to each feature *before* state projections. The drift value starts at 0 and increases to its maximum value of 1.58 over the course of around 1,500 observations. Again, certain projections will affect the positive drift differently. For example, the projection associated with $\mathcal{S}_3$ actually makes this drift *negative* for feature X, while this positive drift negates the expected negative drift for feature Z in all states. We expect this drift to be moderately difficult for each of the monitoring statistics to detect until the drift has been in effect for a few hundred steps. For example, after 300 minutes, the drift has reached $+0.3$, which may be large enough to detect with reasonable accuracy.

Figure 4.6: Multivariate process feature time series before and after drift faults. Fault 2A (top), Fault 2B (middle), and Fault 2C (bottom) are shown after the vertical black line (21:40 on 2 December).

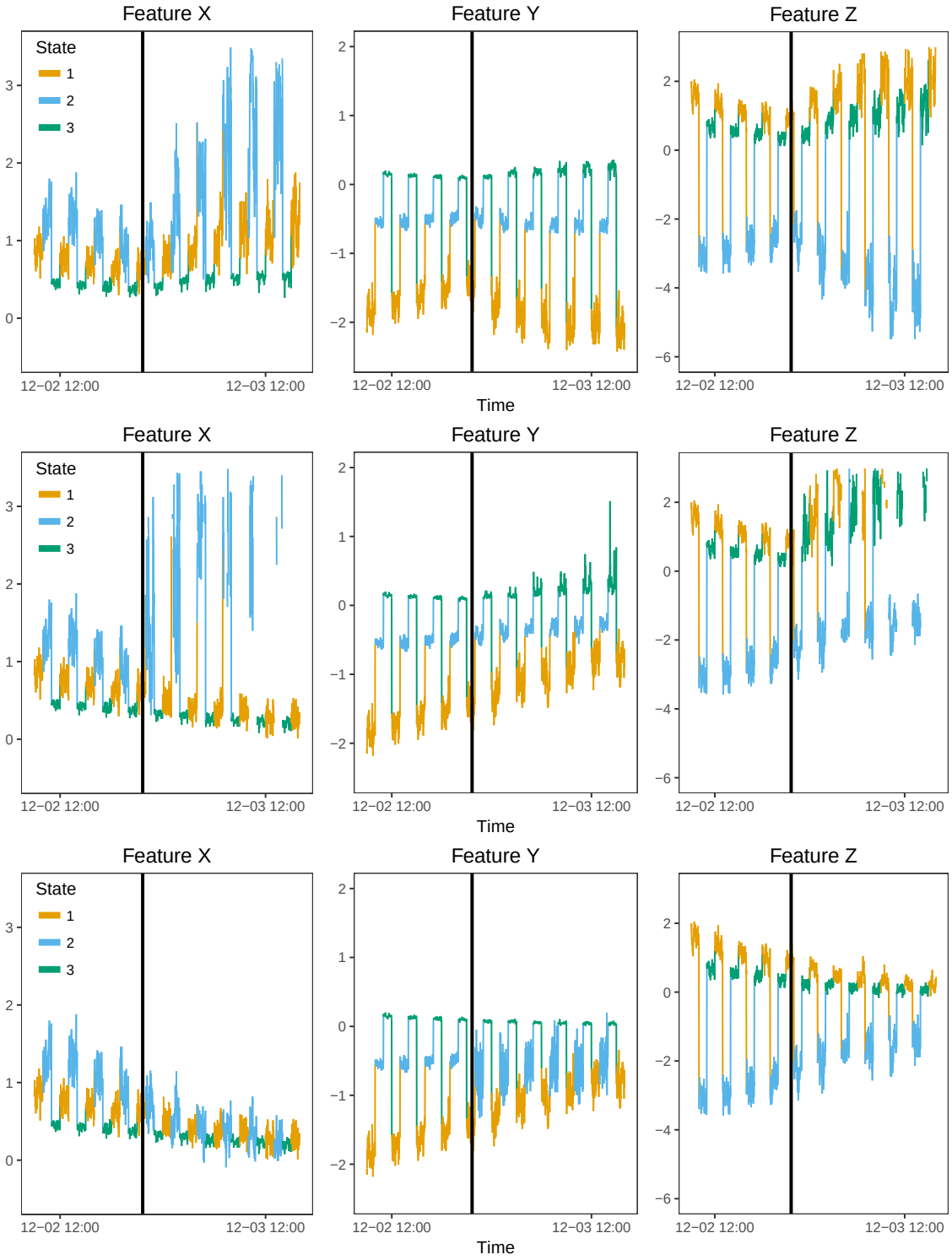


The second row of Figure 4.6 exhibits a realization of each of the three features after the introduction of Fault 2B. By definition, this fault is a positive drift of the X and Z features, but the state projections will allow this fault to affect feature Y as well. Specifically, we see this drift clearly in $\mathcal{S}_2$ of feature X and $\mathcal{S}_1$ of feature Y, and we see a minor effect across each state of feature Z. Because slow drifts can be difficult to detect in the first place, and because this drift does not affect all states and features equally, we expect Fault 2B to be just as or more difficult to detect than Fault 2A.

The last row of Figure 4.6 exhibits a realization of each of the three features after the introduction of Fault 2C. By definition, this fault is a negative drift to feature Y only, and we further constrain this fault to only affect $\mathcal{S}_2$. In this case, the fault was applied *after* the state partitioning and projection, so the other features' realizations remain unaffected, as does the realization for feature Y within $\mathcal{S}_1$ and $\mathcal{S}_3$. At maximum drift, feature Y will be decreased by 1.5. We expect Fault 2C to be the most difficult to detect of the three drift faults.

4.3.3.3 *Latent / error faults: 3A, 3B, and 3C.* Figure 4.7 shows the effects of the "Latent / Error Faults" column in Table 4.1. The first row exhibits a realization of each of the three features after the introduction of Fault 3A. This fault is created by drifting the underlying latent variable t in each feature *before* state projections, and the maximum drift ($\delta+1$) will be $6*t$. As with the other faults, certain projections will affect this drift differently. For example, the projection associated with $\mathcal{S}_3$ actually dampens this latent effect for features X and Y, while the projection associated with $\mathcal{S}_2$ amplifies this latent drift for feature X. Overall, this latent drift amplifies the within-feature, between-state variance. However, Fault 3A has less of an effect on the feature means (other than within $\mathcal{S}_2$ of feature X), so we expect this latent drift to be more difficult to detect.

Figure 4.7: Multivariate process feature time series before and after latent / error faults. Fault 3A (top), Fault 3B (middle), and Fault 3C (bottom) are shown after the vertical black line (21:40 on 2 December).



The second row of Figure 4.7 exhibits a realization of each of the three features after the introduction of Fault 3B. This fault is created by mutating the underlying latent variable t only in feature Z *before* state projections by taking its natural logarithm. Recall that $t \in (0.01, 2)$, so this mutation amplifies the latent effect $\forall t < 1$, but dampens it for $\forall t > 1$. Once again, the state projections will allow this fault to affect the other features as well. Specifically, we see this mutation clearly in $\mathcal{S}_2$ of feature X and $\mathcal{S}_3$ of feature Y, and we see a pronounced effect across $\mathcal{S}_1$ and $\mathcal{S}_3$ of feature Z. We expect Fault 3B to be moderately simple to detect, though perhaps not quite as easy as Fault 1A, for instance.

The last row of Figure 4.7 exhibits a realization of each of the three features after the introduction of Fault 3C. By definition, this fault is a variance increase and negative shift to the errors of feature Y in $\mathcal{S}_2$ only. In this case, the fault was applied *after* the state partitioning and projection, so the other features' realizations remain unaffected, as does the realization for feature Y within $\mathcal{S}_1$ and $\mathcal{S}_3$. We expect Fault 3C to be the most difficult to detect of all the faults we have constructed.

4.3.4 Simulation Design

Based on the data generation process described in Section 4.3.2, we outline the simulation steps.

(1) Draw a set of IC observations with associated fault introduced under $\mathcal{S}_1$ (single-state) or under $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ alternating (multi-state).

(2) Train the fault-detection system on 75% of the observations (7,560 observations for AD-PCA). For MSAD-PCA, note that the number of training observations per class is $7560/3 = 2520$.

(3) Apply AD-PCA and MSAD-PCA to the training and test data sets for each data set under both the single-state and multi-state assumptions.

(4) Measure the following:

   (a) False Alarm Rate: when IC, we expect to see 0 alarms, so the *SPE* and T^2 statistic alarm rates on the IC data are their false alarm rates.

   (b) Fault Detection: for *SPE* and T^2, this is a 1 if the monitoring statistic detected a fault and 0 otherwise.

   (c) Detection Time: for *SPE* and T^2, this is the time in minutes after a fault is introduced until a statistic raises an alarm. Note that the earliest the methods can detect a fault is within three minutes because we set the number of sequential faults necessary to trigger an alarm to be equal to 3.

(5) Repeat Steps 1 – 4 1,000 times.

4.4 Simulation Results

In this section, we present and discuss the time-to-detection results for each of the four monitoring statistics (MSAD-PCA *SPE* and T^2, and AD-PCA *SPE* and T^2) over 1,000 simulation replicates. In Table 4.2, we present the false alarm rates, expected number of false alarms per day, and detection percentages for each of the four monitoring statistics within the multi- or single-state data generation processes. To demonstrate parsing Table 4.2, note that for observations generated under a multi-state process, the AD-PCA T^2 column along the "False Alarm" rows show that the false alarm rate for this statistic is 1.25%, yielding 18.1 false alarms per day on average.

In Table 4.3, the rows are the mean and the 95^{th} percentile ("Upper Tail") of the time until fault detection (in minutes) for each of the four monitoring statistics within the multi- or single-state data generation processes, broken out by each of the nine faults. Cells of Table 4.3 are shaded green if those cells have a detection percentage of 100% as shown in Table 4.2. To demonstrate parsing Table 4.3, for observations

generated under a multi-state process, the AD-PCA T^2 rows under the "1C" column show the following:

- The AD-PCA T^2 statistic detected Fault 1C after 122 minutes on average.
- When the AD-PCA T^2 statistic detected Fault 1C, it detected this fault within 490 minutes in 95% of the simulation replicates.
- The cell is not shaded green because there were simulation replicates where this statistic did not detect a fault at all.

Table 4.2: False alarm rates and detection probabilities. Note that Faults 1C, 2C, and 3C are state-specific and are cannot be applied to observations generated under a single state.

		Multi-State Process				Single-State Process			
		MSAD-PCA		AD-PCA		MSAD-PCA		AD-PCA	
		SPE	T^2	*SPE*	T^2	*SPE*	T^2	*SPE*	T^2
False Alarms	%	0.15	0	0	1.25	0.10	11.5	0.12	14.5
	/Day	2.3	0.1	0.1	18.1	1.6	165.7	1.9	208.9
Shift Faults	1A	100.0	100.0	1.0	99.8	100.0	100.0	99.3	100.0
	1B	100.0	100.0	0.0	97.7	100.0	100.0	100.0	100.0
	1C	100.0	32.0	0.0	96.5	NA	NA	NA	NA
Drift Faults	2A	100.0	100.0	0.0	97.7	25.8	100.0	20.3	100.0
	2B	100.0	100.0	0.0	7.7	100.0	100.0	100.0	100.0
	2C	100.0	100.0	0.0	97.0	NA	NA	NA	NA
Latent / Error Faults	3A	100.0	16.2	3.0	30.6	92.9	98.5	93.0	98.8
	3B	100.0	99.7	98.3	99.0	100.0	100.0	100.0	100.0
	3C	100.0	14.0	0.0	97.0	NA	NA	NA	NA

4.4.1 False Alarm Rates

In Table 4.2, we present the false alarm rates under both the single- and multi-state models for each of the monitoring statistics. These values were calculated by applying the four process-monitoring statistics to IC data for a full week (10,080 observations): three days for training and four days for testing with the model updating every 24

hours. We repeated this data generation, training, and testing 1,000 times with $\alpha = 0.001$. We also remark that the median detection percentage for all four monitoring statistics under these faults is 100%.

We begin by discussing the results from the multi-state data generating process. As we can see in Table 4.2, AD-PCA *SPE* yielded effectively no false alarms on average, but this is not unexpected as this statistic is not very sensitive. In fact, AD-PCA *SPE* only detects Fault 3B with any passable accuracy (98.3%) and fails completely for the other eight faults. In contrast, MSAD-PCA T^2 also maintained a false alarm rate of effectively 0, but was still sensitive enough to detect most faults. Faults 1C, 3A, and 3C were the most difficult for this statistic to detect, detecting only 32%, 16.2%, and 14%, respectively, of these faults. Alternatively, AD-PCA T^2 is hyper-sensitive and issues over 18 false alarms per day on average. This seriously degrades any positive consideration we have for this statistic in terms of time to fault detection. Further, this statistic has difficulty detecting Faults 2B (7.7% detection) and 3A (30.6% detection). In the "sweet spot", MSAD-PCA *SPE* recorded an average 2.3 false alarms per day, while also detecting all nine faults in 100% of the simulation replicates. Overall, note that the AD-PCA T^2 statistic often has better detection probabilities than MSAD-PCA T^2, but at the cost of much higher false alarm rates.

Table 4.2 also shows that all four methods' false alarm rates increase for the single-state model, particularly the T^2 statistics. However, AD-PCA *SPE* increases slightly in false alarm rates, while MSAD-PCA *SPE* actually has fewer false alarms. The increased false alarm rates for MSAD-PCA T^2 for observations from a single-state process illustrate a possible consequence when making a model overly complex by incorrectly assuming a multi-state model. We therefore recommend a thorough exploratory data analysis to justify the necessity of a multi-state modification to an existing fault-detection system before application.

Table 4.3: Detection times for shift, drift, and latent / error faults. Fault 3A has a maximum latent drift of $\delta + 1 = 6$ under Multi-State and $\delta + 1 = 3$ under Single-State. Cells shaded in green have perfect fault detection (see Table 4.2).

				Shift Faults			Drift Faults			Latent / Error Faults		
				1A	1B	1C	2A	2B	2C	3A	3B	3C
Multi-State	MSAD-PCA	*SPE*	Mean	3.0	3.0	171.1	369.9	281.7	699.3	869.4	16.7	34.1
			Upper Tail	3.0	3.0	1032.0	621.1	384.0	953.0	1225.0	54.0	191.0
		T^2	Mean	3.0	97.0	392.2	492.6	570.6	935.8	1351.0	679.0	1251.0
			Upper Tail	3.0	233.0	1354.0	533.0	692.0	1084.0	1518.0	999.6	1432.0
	AD-PCA	*SPE*	Mean	18.0	∞	∞	∞	∞	∞	1372.0	738.2	∞
			Upper Tail	18.0	∞	∞	∞	∞	∞	1518.0	1116.0	∞
		T^2	Mean	5.2	26.0	122.0	1114.0	191.7	81.8	660.9	87.5	9.7
			Upper Tail	5.0	26.0	490.0	1406.0	1514.0	195.0	1459.0	258.0	23.0
Single-State	MSAD-PCA	*SPE*	Mean	9.9	3.0	NA	1171.0	304.7	NA	1060.0	17.5	NA
			Upper Tail	32.0	3.0	NA	1568.0	401.8	NA	1481.0	48.0	NA
		T^2	Mean	3.0	4.1	NA	85.8	93.4	NA	435.4	95.2	NA
			Upper Tail	3.0	5.0	NA	156.5	173.0	NA	942.3	220.8	NA
	AD-PCA	*SPE*	Mean	12.4	3.0	NA	1141.0	312.3	NA	1055.0	17.9	NA
			Upper Tail	46.0	3.0	NA	1562.0	405.8	NA	1476.0	50.8	NA
		T^2	Mean	3.0	4.1	NA	83.2	90.1	NA	412.5	91.2	NA
			Upper Tail	3.0	5.0	NA	152.8	168.0	NA	929.9	222.8	NA

4.4.2 Detection Times

Table 4.3 shows that Fault 1A for a multi-state process is detected by three of the methods almost instantly, except for AD-PCA *SPE*. The AD-PCA *SPE* monitoring statistic does not detect a fault until minute 18 on average. Fault 1B is similar, but slightly more difficult to detect, as expected. Also as expected, Fault 1C is the most difficult of the shift faults to detect; comparing AD-PCA T^2 with MSAD-PCA *SPE*, the average time to fault detection is 122 minutes (AD-PCA T^2) or 171 minutes (MSAD-PCA *SPE*). However, the 95^{th} percentile of MSAD-PCA *SPE* takes over twice as long as AD-PCA T^2. The reason for this is in the detection probability. As shown in Table 4.2, MSAD-PCA *SPE* detects 100% of these faults (increasing the time to fault detection in the upper tail), while AD-PCA T^2 detects Fault 1C in only 96.5% of the replicates.

For the multi-state drift faults (Fault 2A, 2B, and 2C), our expectation that the drift fault would take a few hours to detect seems to hold true. Interestingly, MSAD-PCA *SPE* detected Fault 2B faster than Fault 2A, while MSAD-PCA T^2 detected Fault 2A faster than Fault 2B. Also, AD-PCA T^2 detected Fault 2B on average hours before either MSAD-PCA *SPE* or T^2, but Table 4.2 shows it only detected Fault 2B in 7.7% of replicates. Fault 2C began in $\mathcal{S}_2$, so the earliest a method could *correctly* detect a fault would be 21 minutes after the fault start index. As expected, Fault 2C was the most difficult drift fault to detect for MSAD-PCA *SPE* and MSAD-PCA T^2, but was uncharacteristically easy to detect by AD-PCA T^2 in comparison. We attribute this quick fault detection to the fact that AD-PCA T^2 is overly sensitive, which is shown by the high false alarm rate (over 18 per day) for this statistic.

Of the two latent-feature faults (3A and 3B) and the error fault (3C), Fault 3A was the most difficult fault for all four methods to detect, as shown by the low detection proportions for MSAD-PCA T^2 and AD-PCA *SPE* and T^2. Recall that this fault was a drift in the underlying variable t, changing it to $(\delta + 1)t$ at its maximum

drift. Fault 3B was markedly easier to detect by comparison, and MSAD-PCA *SPE* detected this fault quickly and consistently. Fault 3B is also the only fault under the multi-state data generation model in which AD-PCA *SPE* performs comparably to the other monitoring statistics. However, Fault 3C (a perturbation of the error structure) shows AD-PCA *SPE* is again ineffective. This fault is especially difficult even for MSAD-PCA T^2 to detect, detecting a fault in only 14% of cases. However, MSAD-PCA *SPE* and AD-PCA T^2 detect this fault relatively quickly.

The fault detection times for single-state data are also included in Table 4.2 to show that the multi-state methods detect faults as expected, even in the absence of discernible state information. This table shows that when the data come from from a single-state process, MSAD-PCA is not significantly different from AD-PCA in terms of false alarm rates, detection percentage, or mean and upper tail detection times. These results show that MSAD-PCA does not suffer a loss of power in well-posed multivariate process scenarios, even when the process is erroneously assumed to have the multi-state property.

Overall, the best performing monitoring statistics by average time to fault detection are MSAD-PCA *SPE* and AD-PCA T^2. The best performing monitoring statistics by detection percent are MSAD-PCA *SPE*, MSAD-PCA T^2, and AD-PCA T^2. The best performing monitoring statistics by false alarm rates are MSAD-PCA *SPE*, MSAD-PCA T^2, and AD-PCA *SPE*. In summary, we consider the MSAD-PCA *SPE* monitoring statistic to be the best combination of high detection probability, short time to detection, and low false alarm rates. While the AD-PCA T^2 monitoring statistic does perform very well in some scenarios, it also performs very poorly in others. Combine this with its high false alarm rate, and we consider this statistic to be too erratic to consider in practice.

To summarize these results, when considering false alarm rates, fault detection probabilities, and time to fault detection for data generated under a single state, the

MSAD-PCA test statistics perform almost identically to their AD-PCA counterparts. When data are generated from multiple states, the MSAD-PCA test statistics have the valuable combination of lower false alarm rates and higher detection probabilities. Furthermore, their times to fault detection are superior (especially after considering their detection probabilities). Overall, the MSAD-PCA *SPE* monitoring statistic exhibits excellent detection probabilities and good detection times, at the cost of a few false alarms per day. In comparison, the MSAD-PCA T^2 monitoring statistic, while slower to detect faults than its AD-PCA counterpart, makes up for this lack of sensitivity by perfect detection in over half of the faults and perfect specificity of the test (within simulation error). For these reasons, we believe that pairing the more sensitive MSAD-PCA *SPE* monitoring statistic with the specific MSAD-PCA T^2 monitoring statistic is the best course of action.

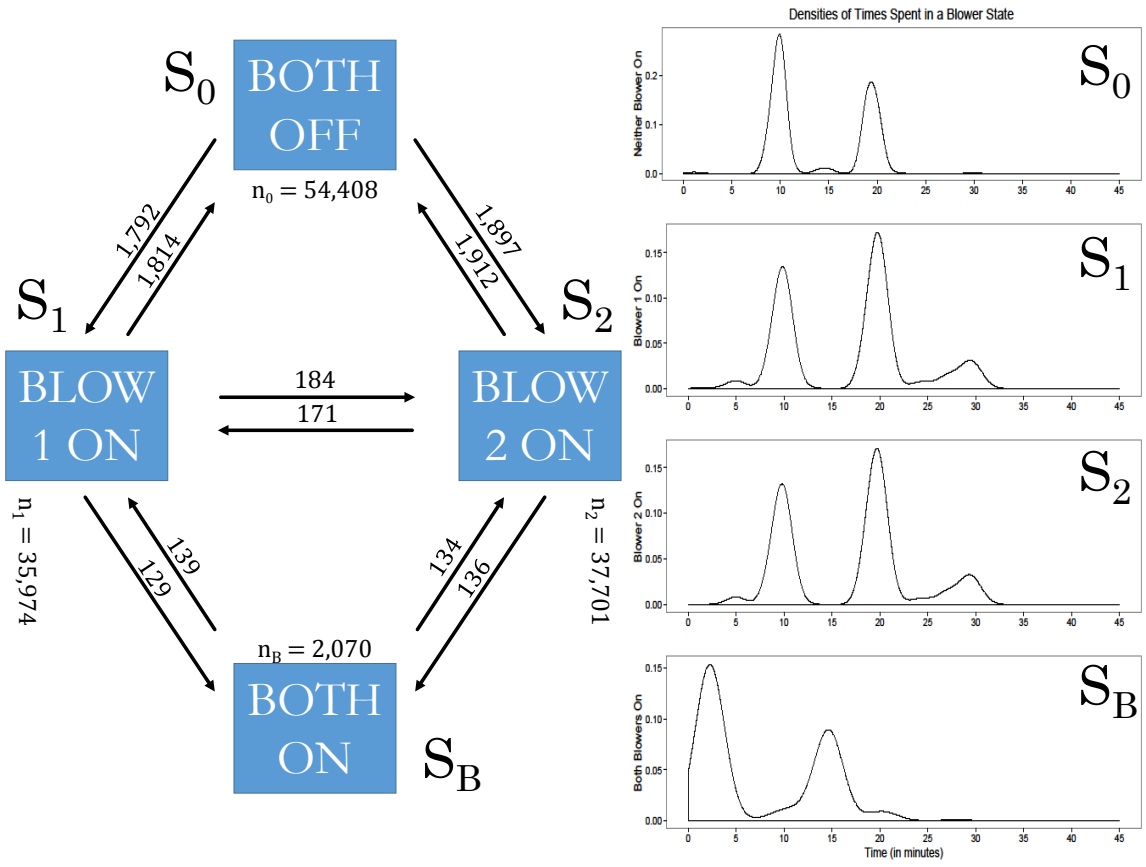
4.5 Case Study

To illustrate these methods in practice, we have data from a decentralized WWT plant in Golden, CO recorded at three different temporal frequencies: 20 features at the five-second-level, 20 at the one-minute-level, and 15 at the 10-minute-level. We are interested in detecting process faults remotely. That is, can the MSAD-PCA model be applied without human intervention to detect when the WWT process may be operating out of control? We build on the work of Kazor et al. (2016) by hypothesizing that the underlying multivariate process may change based on the operation of a set of two blowers. However, blower power (on / off) is recorded at the one-minute time scale. We need to perform three major tasks: preliminary data cleaning, data aggregation and downscaling to the one-minute frequency, and secondary data cleaning and joining. After importing the data into `R`, we first remove gross outliers identified by visual inspection of the univariate time series of each feature.

4.5.1 Specific System Considerations

The plant has two separate blowers used to mix the suspended solids mixture (see the figures and explanation in Section 4.1.1 for more detail). When either of these blowers are in operation, some process features change dramatically. The blowers have independent controls, so we have four states: $\mathcal{S}_0$ = both blowers off; $\mathcal{S}_1$ = only blower one on; $\mathcal{S}_2$ = only blower two on; and $\mathcal{S}_B$ = both blowers on. See Figure 4.8 for reference. We observe 130,153 sequential one-minute-level IC observations at this facility. The operation of the blowers is known and controlled by the process engineer(s), so we investigate the lengths of stay in each blower state.

Figure 4.8: The blower process flow chart shows the number of times the process changes states and in which directions (left). The lengths of times spent in each state are bi-modal, as shown by their densities (right).



When both blowers are off ($\mathcal{S}_0$), $n_0 = 54,408$. This is the most common state. The process changes from $\mathcal{S}_0$ to $\mathcal{S}_1$ or $\mathcal{S}_2$ with similar frequency, but never changes directly to $\mathcal{S}_B$. As we can see from the top three densities of Figure 4.8 (right), the modal lengths of time to remain in $\mathcal{S}_0$, $\mathcal{S}_1$, and $\mathcal{S}_2$ are 10 or 20 minutes. If neither blower is on, we surmise that the water treatment protocol is to leave both blowers off for either 10 or 20 minutes, then turn one of the blowers on. $\mathcal{S}_1$ (only blower one on) and $\mathcal{S}_2$ (only blower two on) also occur frequently ($n_1 = 35,974,\ n_2 = 37,701$). Based on the diagram in Figure 4.8 (left), if a blower is on, we surmise that the most common water treatment protocol is to leave that blower on for 10 or 20 minutes, then turn that blower off. The last state, $\mathcal{S}_B$ (both blowers on), is comparatively rare, with $n_B = 2,070$ observations recorded (less than 2% of the observations). Thus, $\mathcal{S}_B$ is purely a transition state between $\mathcal{S}_1$ and $\mathcal{S}_2$, and we do not believe that accounting for this rare state will increase model accuracy. We randomly assigned the $< 2\%$ of observations in $\mathcal{S}_B$ to either $\mathcal{S}_1$ or $\mathcal{S}_2$ with equal probability.

4.5.2 Data Cleaning

Since the blower operation indicator is recorded at the one-minute-level, and the effects of the blower are almost immediate for features affected by blower status, we need to down-scale all of the 10-minute observations to the one-minute level as follows:

(1) Interpolate nine observations between the 10-minute recorded observations. For continuous features, we use linear interpolation. For discrete features, we use "last object carried forward" (LOCF) interpolation.

(2) Estimate the Median Absolute Deviation (MAD) of each feature via an appropriate method:

   (a) For stationary features, estimate one MAD for the entire process (Figure 4.9, top left).

(b) For processes with a trend and constant variance, fit a smoothing spline to the feature over time, and estimate a single MAD of the residuals (Figure 4.9, top right).

(c) For processes with relatively constant mean but non-constant variance, calculate a piecewise estimate of the MAD over empirically chosen blocks of time (Figure 4.9, bottom left).

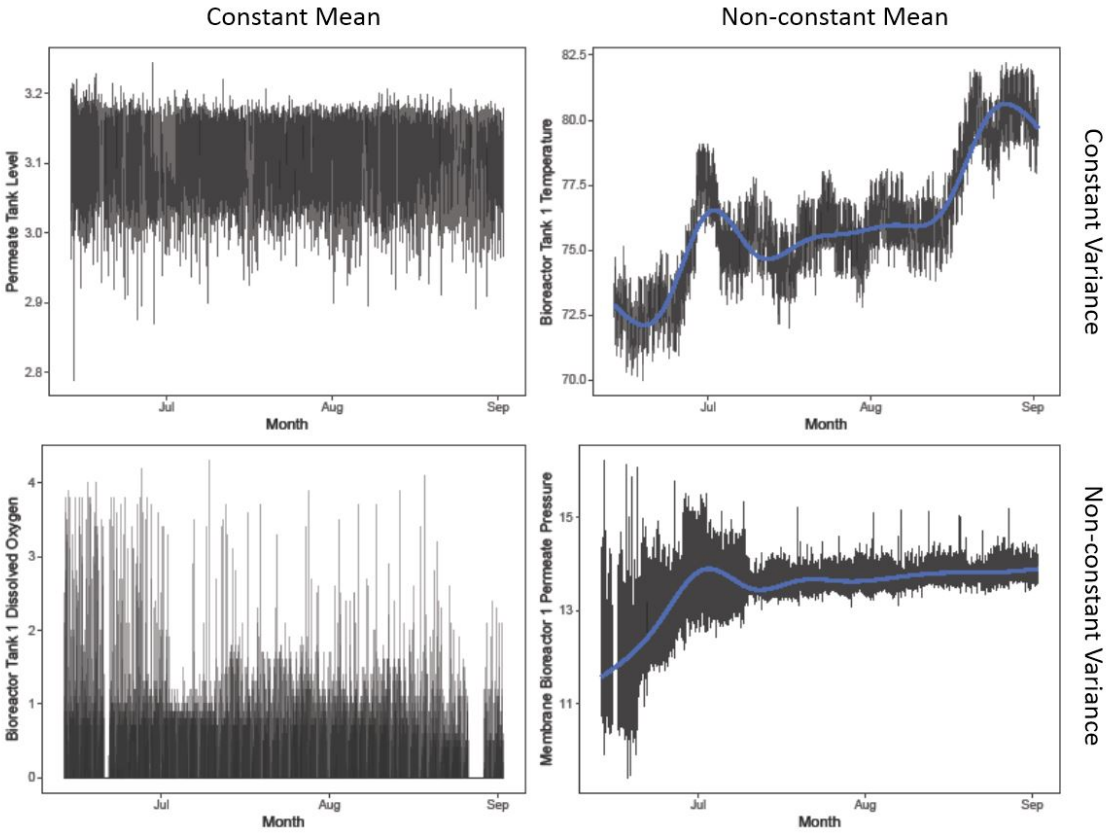
(d) For non-stationary processes in both mean and variance, fit a smoothing spline to the feature over time, and calculate a piecewise estimate of the MAD of the residuals (Figure 4.9, bottom right).

(3) To avoid underestimating process variance, we add a white noise component to each linearly interpolated value with standard deviation equal to the feature- and time-specific MAD values divided by $\sqrt{2/\pi}$ (Geary, 1935).

We fit cubic splines via the function `smooth.spline` in `R`, which uses a knot at each observation. For a thorough discussion of this function, see Venables & Ripley (1999). For processes with variable mean but constant variance, we fit this cubic spline and estimate the MAD of the process from the residuals of the spline fit. For processes with variable mean and variance, we also fit a cubic spline and consider the residuals. However, some residuals require piecewise MAD estimation, and some require MAD estimation via a linear trend. For process features exhibiting heteroscedasticity, we transform the feature values via the natural logarithm. After downscaling and jittering the 10-minute-level observations, we round all observation time indices to the nearest minute. Because of the difficulty of analyzing features recorded at different frequencies, all features at the WWT facility are now recorded on the same one-minute scale.

With all observations on the one-minute scale, we can see some of the differences among the variable values across states in Figure 4.2 and in $\mathcal{S}_1$ and $\mathcal{S}_2$ versus $\mathcal{S}_0$ via 49×49 correlation heatmaps in Figure 4.3. As discussed in Section 4.1.1, both the feature values themselves and the relationships between the features change based

Figure 4.9: Example processes for four different MAD estimation techniques: (*top left*) a stationary process, (*top right*) a process with a trend and constant variance with overlaid general additive smoother, (*bottom left*) a process with constant mean and non-constant variance, and (*bottom right*) a process with non-stationary mean and variance with overlaid general additive smoother.



upon the multivariate process state. Changing from $\mathcal{S}_0$ (neither blower on) to $\mathcal{S}_1$ (blower one on) has a profound effect on the recorded process values for MBR 1 (row 13 of Figure 4.3, *left*), and for other features (such as raw, influent, and waste flow values). We see a similar effect on MBR 2 (row 14 of Figure 4.3, *right*) when the state changes from $\mathcal{S}_0$ (neither blower on) to $\mathcal{S}_2$ (blower two on). Therefore, the PCA decomposition will change from one state to another. Accounting for these process changes should improve our ability to correctly detect system faults.

4.5.3 Real-Data Results

We now discuss the performance of MSAD-PCA on some real data.

4.5.3.1 *False alarm rates.* We first applied our method to the IC data set from 1 June – 1 September 2013 described in Kazor et al. (2016), monitoring the false alarm rates using the first 40 days of the 81-day data set. They showed that the false alarm rates for the AD-PCA projection system were 4.4% for *SPE* and 2.2% for T^2 when monitoring ICC for 10-minute-averaged observations. They required three flagged observations in a row to issue an alarm.

For the same observations interpolated to the 1-minute level, the AD-PCA false alarm rates were 0.3% for both *SPE* and T^2. Similarly, the MSAD-PCA rates were 0.3% for *SPE* and 0.1% for T^2. For the 1-minute-level observations, our requirement for an alarm was five flagged observations in a row instead of three (which is different from the simulation described in Section 4.3.4), and this increase is justified in the following paragraph. This slight decrease in the false alarm rate of the T^2 monitoring statistic agrees with the simulation results showing that the MSAD-PCA T^2 has higher specificity than the single-state versions.

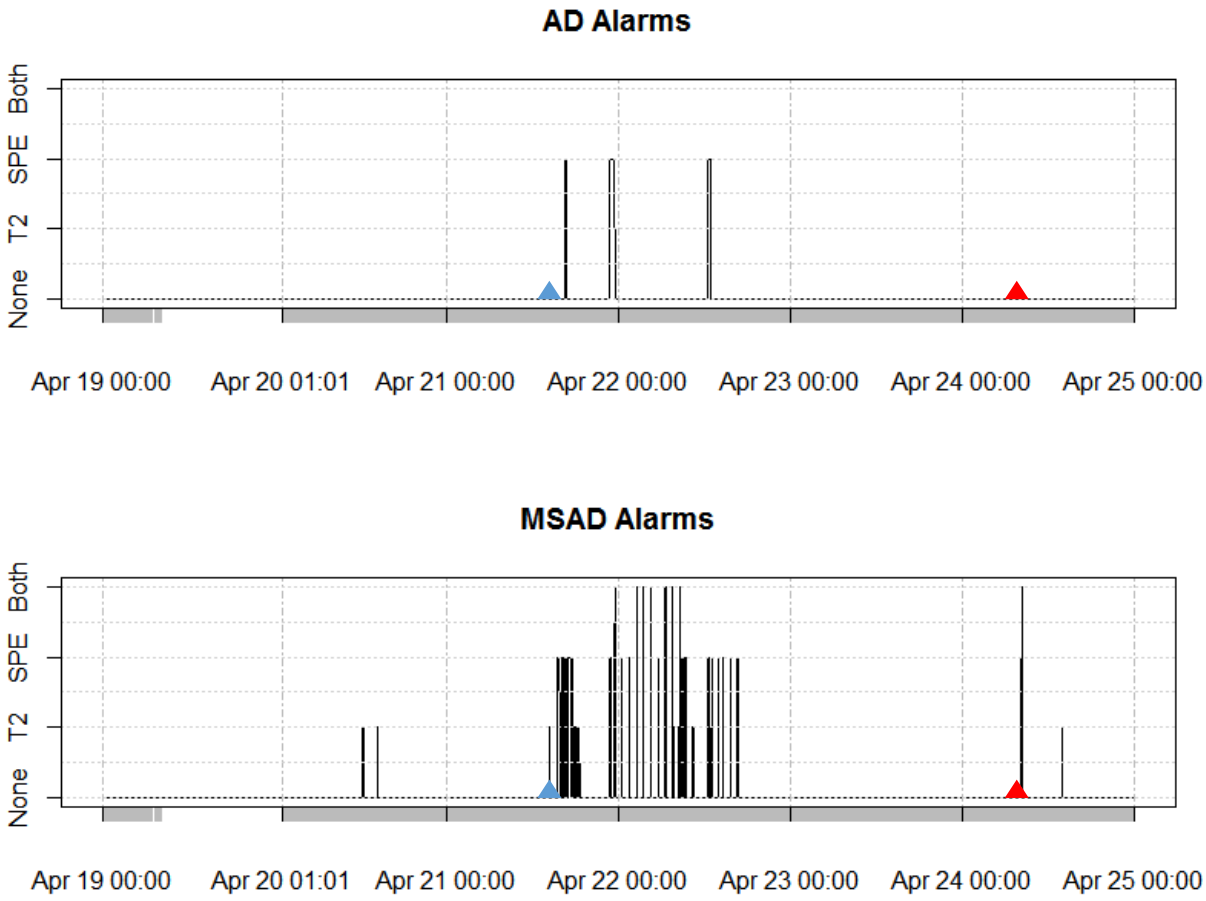
However, the false alarm rates for both AD-PCA and MSAD-PCA are higher than the specified α threshold of 0.1%. These higher false alarm rates are likely due to

observations being serially autocorrelated. Therefore, if one observation is flagged as beyond the threshold for its *SPE* or T^2 monitoring statistics, the monitoring statistics for the next 2 to 5 observations also have a higher probability of being beyond these same thresholds. Because of this, we increased the number of sequential flags necessary to trigger an alarm from 3 to 5. Of note, in the Kazor et al. (2016) paper, 3 flagged observations in a row corresponded to 30 minutes of data. Here, 5 consecutive flags correspond to 5 minutes. One reason for this change is that serial autocorrelation will have a greater impact on 1-minute averaged observations than it will on 10-minute observations.

4.5.3.2 *Time to fault detection.* We also applied our method to a fault analyzed by Kazor et al. (2016). To keep similar specifications as their paper, we train on the observations from 10 April at 01:10 to 19 April at 00:00, for a total of 12,854 observation recorded at the 1 minute level. Operators detected a fault at 10:00 on 24 April, 2010. After downscaling the observations to the 1-minute level, the AD-PCA method first detects a fault at 16:31 on 21 April, and triggers a few alarms between then and the point at which the operators detected the fault. This is shown in Figure 4.10 (*top*). Furthermore, none of the observations projected with the AD-PCA method trigger *both* the *SPE* and T^2 alarms.

After applying the MSAD-PCA method, the first alarms were issued around noon on 20 April, but after inspecting the feature process graphs, we believe that these first alarms were triggered by the observations directly following a period of missing data on the morning of the 20^{th} and not by the onset of a system fault. However, Figure 4.10 (*bottom*) shows that observations projected with the MSAD-PCA method trigger alarms starting at 14:15 on 21 April, two hours before AD-PCA. Moreover, these alarms are persistent for longer than a 24-hour period (from 14:15 on 21 April to 17:00 on 22 April), and some of these observations trigger both the *SPE*

Figure 4.10: Observation alarms issued after AD-PCA (*top*) and MSAD-PCA (*bottom*) projections during a real system fault that occurred between 21 and 24 April, 2010. The y-axis represents the categories for when an observation triggers no alarms, a T^2 alarm, an *SPE* alarm, or both alarms. The blue triangles are at 14:15 on 21 April, when the MSAD-PCA monitoring method first detects a fault. The red triangle at 10:00 on 24 April is when the human operators detected a fault.



and T^2 alarms. Based on these results from this real data case, we believe that our new projection method will warn system operators of faults earlier, and it will more consistently detect faults as they occur in real time.

4.6 Conclusion

In conclusion, we have illustrated how incorporating state information when monitoring non-linear, non-stationary, autocorrelated, and non-Gaussian multivariate process data can improve fault detection results. We have crafted a simulation study to test the utility and reliability of this new method and have described the data generated within this study. Additionally, we have shown average false alarm rates, detection times, and detection probabilities for nine different faults, some of which occur only in one of the states. Further, we have discussed the real data case that motivated the multi-state monitoring paradigm, and we have briefly described the data exploration process that lead to using a multi-state framework. We have concluded by comparing the false alarm rates and time to fault detection with and without accounting for multiple process states for a real fault in the decentralized WWT plant in Golden, CO. Finally, we have demonstrated the added benefit of correctly monitoring a true multi-state process with a multi-state model.

When choosing which states to include in a model, we recommend having at least $p^2/2$ IC observations per state for training, because multi-state monitoring with MSAD-PCA estimates state-specific covariance matrices. Consequently, we also recommend that the frequency of membership for each state not be drastically dissimilar. For example, five states with membership probabilities of $(0.2, 0.2, 0.2, 0.2, 0.2)$, $(0.25, 0.11, 0.07, 0.36, 0.21)$, or $(0.6, 0.1, 0.1, 0.1, 0.1)$ would probably be acceptable, but a membership probability of $(0.96, 0.01, 0.01, 0.01, 0.01)$ would most likely cause problems. Furthermore, we recommend considering states in which the process distribution, mean vector, and / or covariance matrix change significantly between states.

Finally, in the interest of model parsimony, we do not recommend including too many states. In short, block on states meeting these criteria: states that have different vector or matrix moments; states with each training sample size at least $p^2/2$; mutually exclusive states that contain all system observations; and states with similar frequencies.

Further, we recommend monitoring multi-state multivariate processes that exhibit non-stationarity and serial autocorrelation by pairing the SPE and T^2 monitoring statistics after dimension reduction via multi-state adaptive-dynamic PCA. We have shown in our simulation study and the case study that the MSAD-PCA *SPE* monitoring statistic offers an excellent combination of sensitivity and high detection probability, while the MSAD-PCA T^2 monitoring statistic offers an excellent combination of specificity and high detection probability. Once implemented, this approach can save engineers months of work reconstructing the stable biological environment necessary for water treatment by warning of impending faults well in advance. Accounting for mutually exclusive states can also help to identify and attribute causes of faults when they occur in a specific state and is the next step in process monitoring (Choi, Lee, Lee, Park, & Lee, 2005; Zheng & Zhang, 2013; Zou, Jiang, & Tsung, 2011). To implement the MSAD-PCA process in `R`, consider using our package `mvMonitoring`, discussed in the next chapter.

Acknowledgments

This work in multivariate statistical process monitoring is motivated by our partnership with the Colorado School of Mines and is supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) under Award No: OSR-2015-CRG4-2582 and by the National Science Foundation PFI:BIC Award No: 1632227.

CHAPTER FIVE

Multivariate Statistical Process Control with `mvMonitoring`

ABSTRACT

The task of automating a decentralized waste-water treatment process relies heavily on accurate process monitoring. This vignette describes the motivation and design details of the `R` package `mvMonitoring` and its utility when applied to multivariate, state-switching, non-stationary, non-linear, and non-Gaussian stochastic process data. This document serves as a tutorial to introduce engineers to the four main functions within this package: `mspProcessData`, `mspTrain`, `mspMonitor`, and `mspWarning`. Further, this vignette also describes the workflow necessary in order to analyze multivariate process monitoring data via this package.

5.1 Introduction

This is the accompanying package to the research published by Kazor et al. (2016) and the previous chapter. The `mvMonitoring` package is designed to make simulation of multi-state multivariate process monitoring statistics easy and straightforward, as well as streamlining the statistical process monitoring component. This package can be downloaded from GitHub by running the following:

```
devtools::install_github("gabrielodom/mvMonitoring",
auth_token = "tokenHere")
```

where you create the value of "tokenHere" by generating a personal access token (PAT) at `https://github.com/settings/tokens` and copying the quoted string to this argument.

The outline of this vignette is as follows. Section 5.2 discusses the motivation for creating this package and briefly explains the utility of multi-state process monitoring. Section 5.3 offers a thorough explination of how to generate the synthetic data employed to test and compare the new multi-state modification of the process monitoring setup with the `mspProcessData` function. Section 5.4 describes the necessary information and data formatting to effectively train the fault detection algorithm using the `mspTrain` function. Then, Section 5.5 lists and explains the necessary inputs to the `mspMonitor` and `mspWarning` functions and examines their outputs. Section 5.6 shows a step-by-step walkthrough of how to implement the four `mvMonitoring` functions in practice. Some concluding remarks are offered in Section 5.7.

5.2 Motivation

5.2.1 Why Use `mvMonitoring`?

The `mvMonitoring` package can be used to detect outliers in a correlated, non-Gaussian multivariate process with non-linear, non-stationary, or autocorrelated feature behavior. These process outliers are often indicative of system fault. The naive (but unfortunately common) approach to multivariate process monitoring is to use expert opinion to identify a few important features to monitor visually, and raise an alarm if these features travel outside pre-defined normal operating boundaries.

However, this split univariate approach fails to account for the correlated nature of the process features, so some engineers have taken to monitoring the system as a single correlated multivariate process rather than a collection of independent univariate processes. In the literature review of the motivating papers, the authors cite how this approach has benefited the science of process monitoring as a whole. Unfortunately, this approach has its own shortcomings. One such complication is that monitoring a multivariate process in its original feature space can lead to exorbitant

computational costs, as discussed in Huang, Kong, & Huang (2014). Additionally, correlated multivariate processes over many features may be noisy and often retain redundant information. To combat these issues, *principal components analysis* (PCA) and its many modifications have been employed to assist in monitoring multivariate processes as a whole. This package is an implementation of one such PCA modification.

5.2.2 Multi- or Single-State AD-PCA

Adaptive-Dynamic PCA (AD-PCA), thoroughly discussed in Kazor et al. (2016) as well as in the previous chapter, accounts for non-linearity, non-stationarity, and autocorrelation in non-Gaussian multivariate processes. As an additional layer of complexity within this model, consider such a process with multiple known system states. *Multi-State* monitoring is thus a modification to PCA which accounts for multiple process states and models each one separately. States can be any mutually exclusive blocking factor, and states do not necessarily follow a strict order. Then, Multi-State AD-PCA (MSAD-PCA) allows process engineers to account for distinct process states. Specifically, this modification should be used when features under different process states have different means, correlations, variances, or some combination of these three.

5.3 Simulating Data with `mspProcessData`

The `mspProcessData` function generates three-dimensional multi-state or single-state non-linear, non-stationary, and autocorrelated process observations. We follow the original work of Dong & McAvoy (1996) for generation of the foundational stationary and independent features.

5.3.1 Latent Feature Creation

So that the simulated features have non-zero correlations, Dong and McAvoy created their three features as polynomial functions of a single latent variable t_s, where $s = 1, \ldots, \omega$ is the observational index of the process.

5.3.1.1 *Autocorrelated and non-stationary errors.* The `mspProcesData` function induces autocorrelation in t through its errors, ε_s, where

$$\varepsilon_1 \sim \mathcal{N}\left(\frac{1}{2}(a+b)(1-\varphi), \frac{b-a}{12}(1-\varphi^2)\right),$$

where $a = 0.01$ and $b = 2$. Now, we define the first-order autoregressive process on ε_s by

$$\varepsilon_s = \varphi\varepsilon_{s-1} + (1-\varphi)\varepsilon,$$

where ε is a random innovation drawn from the Normal distribution defined in the previous expression, and the autocorrelation component $\varphi = 0.75$. The mean and variance multipliers are the mean and variance of a random variable from the uniform$_{[a,b]}$ distribution.

5.3.1.2 *The non-linear latent process.* This t vector will be sinusoidal with period $\omega = 7 * 24 * 60$ (signifying a weekly period in minute-level observations). We then synthesize a t by taking

$$t_s^* = -\cos\left(\frac{2\pi}{\omega}s\right) + \epsilon_s,$$

and then scaling t^* to

$$t = \frac{(b-a)(t_s^* - \min(t_s^*))}{\max(t_s^*) - \min(t_s^*)} + a.$$

Finally then, the t vector will lie entirely in $[a, b]$.

5.3.2 Single-State and Multi-State Features

5.3.2.1 *Single-state features.* First `mspProcessData` simulates three features, with each feature operating under k different states. Let $\langle x_k(t_s), y_k(t_s), z_k(t_s)\rangle$ be the process evaluated at t_s within State k. These are the three features under State 1 (in control, or IC) as three functions of t:

$$x(\mathbf{t}) \equiv \mathbf{t} + \boldsymbol{\varepsilon}_1, \tag{5.1}$$

$$y(\mathbf{t}) \equiv \mathbf{t}^2 - 3 * \mathbf{t} + \boldsymbol{\varepsilon}_2, \tag{5.2}$$

$$z(\mathbf{t}) \equiv -\mathbf{t}^3 + 3 * \mathbf{t}^2 + \boldsymbol{\varepsilon}_3, \tag{5.3}$$

where $\varepsilon_i \sim N(0, 0.01),\ 1 = 1, 2, 3$. The `mspProcessData` function calls the internal `processNOCdata` function to generate IC single-state observations.

5.3.2.2 *Multi-state features.* The multi-state feature expression is induced by rotation and scaling of certain sets of observations. To induce a three-state, hourly switching process (the default), the `mspProcessData` function will create a label column that switches from "1" to "2" to "3" every hour. State "1" will be the features generated under the single-state assumption, while State "2" and State "3" are generated as follows. These states will be scaled rotations of the current $\langle x, y, z\rangle$ set. The second state is yaw, pitch, and roll rotated by (0, 90, 30) degrees, and the scales are multiplied by (1, 0.5, 2). The third state is yaw, pitch, and roll rotated by (90, 0, -30) degrees, and the scales are multiplied by (0.25, 0.1, 0.75). That is,

(1) $\mathcal{S}_1$: $\mathbf{X}(t_s) := \langle x(t_s), y(t_s), z(t_s)\rangle$.

(2) $\mathcal{S}_2$: $\mathbf{X}(t_s) := \langle x(t_s), y(t_s), z(t_s)\rangle \cdot \mathbf{P}_1\boldsymbol{\Lambda}_1$, where

$$\mathbf{P}_1 = \begin{bmatrix} 0 & 0.50 & -0.87 \\ 0 & 0.87 & 0.50 \\ 1 & 0 & 0 \end{bmatrix}$$

is the orthogonal rotation matrix for a yaw, pitch and roll degree change of $\langle 0°, 90°, 30° \rangle$, and $\mathbf{\Lambda}_1 = \text{diag}(1, 0.5, 2)$ is a diagonal scaling matrix.

(3) $\mathcal{S}_3$: $\mathbf{X}(t_s) := \langle x(t_s), y(t_s), z(t_s) \rangle \cdot \mathbf{P}_2 \mathbf{\Lambda}_2$, where

$$\mathbf{P}_2 = \begin{bmatrix} 0 & 0.87 & -0.50 \\ -1 & 0 & 0 \\ 0 & 0.50 & 0.87 \end{bmatrix}$$

is the orthogonal rotation matrix for a yaw, pitch and roll degree change of $\langle 90°, 0°, -30° \rangle$, and $\mathbf{\Lambda}_2 = \text{diag}(0.25, 0.1, 0.75)$ is a diagonal scaling matrix.

These rotation matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ turn the states in three-dimensional space so that the states are at right angles to each other in at least one dimension, and the scaling matrices $\mathbf{\Lambda}_1$ and $\mathbf{\Lambda}_2$ inflate or deflate the process variances along each principal component. The `mspProcessData` function calls the internal function `dataStateSwitch` which splits the observations by state and applies the state-specific rotation and scaling through the internal `rotateScale3D` function.

5.3.3 Synthetic Fault Induction

Faults can be introduced to single- or multi-state data via the `mspProcessData` function. The default fault start index is 8500, or roughly 84% through the cycle of 10,080 observations. These faults are added through the internal `faultSwitch` function.

(1) Fault 1A is a positive shift to all three features before state rotation: $\mathbf{X}^*(t_s) = \mathbf{X}(t_s) + 2, \ s \geq 8500$.

(2) Fault 1B is a positive shift to the x feature before state rotation: $x^*(t_s) = x(t_s) + 2, \ s \geq 8500$.

(3) Fault 1C is a positive shift to the x and z features in State 3 only and *after* state rotation: $x^*(t_s) = x(t_s) + 2, \ z^*(t_s) = z(t_s) + 2, \ s \geq 8500$.

(4) Fault 2A is a positive drift across all the process monitoring features before state rotation: $\mathbf{X}^*(t_s) = \mathbf{X}(t_s) + (s - 8500) \times 10^{-3}$, $s > 8500$.

(5) Fault 2B is a positive drift across the y and z process monitoring features before state rotation: $y^*(t_s) = y(t_s) + (s - 8500) \times 10^{-3}$, $z^*(t_s) = z(t_s) + (s - 8500) \times 10^{-3}$, $s > 8500$.

(6) Fault 2C is a negative drift in the y process monitoring feature in State 2 only and *after* state rotation: $y^*(t_s) = y(t_s) - 1.5 \times \frac{s-8500}{10080-8500}$, for $s > 8500$.

(7) Fault 3A is an amplification of the underlying latent variable t for all features. The maximum latent drift of this fault will be 5 + 1: $\mathbf{X}^*(t_s) = \mathbf{X}(t_s^*)$, $s > 8500$, where $t_s^* = \left[\frac{5(s-8500)}{\omega-8500} + 1\right] t_s$.

(8) Fault 3B is a mutation of the underlying latent variable t for the z feature: $z^*(t_s) = z(\log t_s^*)$, $s \geq 8500$. This fault will dampen the underlying latent effect for z if $t_s > 1$ and amplify this effect if $t_s < 1$.

(9) Fault 3C is a polynomial mutation of the error for the y feature in State 2 only and *after* state rotation: $y^*(t_s) = y(t_s) + 2 * \mathbf{e}_3(s) - 0.25$, for $s > 8500$.

5.3.4 Composing the Data Generation

The `mspProcessData` function can generate weeks of non-linear, non-stationary, auto-correlated, multi-state, multivariate process data useful to test new process monitoring techniques. Users can generate IC observations to measure false alarm rates, or induce one of nine pre-built faults to test detection time and consistency with repeated Monte Carlo sampling. We expect this function will generate interesting data useful to compare new and improved process monitoring techniques with existing methods. A detailed description of this function's inputs and outputs are shown in Section 5.6.

5.4 Training with `mspTrain`

The `mspTrain` function will generate projection matrices and test statistic thresholds from training data matrices. It requires data matrices to be in the `xts` matrix format.

5.4.1 The `xts` *Data Matrix*

An `xts` data matrix is first and foremost a matrix, *not* a data frame. For users very familiar with data frame manipulation (with `dplyr` for instance), the slight but profound differences between manipulating matrices and data frames quickly become apparent. Because of the class requirements for matrices, *all* features must be integer or double vectors. The `mspTrain` function cannot train on non-numeric information.

The `xts` object class stands for *extendible time series* and comes from the package `xts`, which is built on the package `zoo`. The date and time information for the multivariate stochastic process (necessarily as `POSIX` objects) are stored as the row indices of `xts` matrices. We recommend the package `lubridate` for manipulating `POSIX` objects.

5.4.2 The State Vector

The state membership (or *class*) vector of the data matrix is the column of integer values corresponding to the generation state of the observation. When implementing single-state AD-PCA, this class vector will simply be a numeric column of the same value – for instance, 1. However, for MSAD-PCA, the class vector indicates from which state each observation has been drawn.

The `mspTrain` function will split the observations by the class vector, apply single-state AD-PCA to each class, then return the class-specific projection matrices and thresholds. Because of this split-apply-combine strategy, users must ensure that one or more classes are not too "rare" – that is, the class sample sizes should be sufficiently large to allow for stable covariance matrix inversion. For p features (including

lags), stable covariance inversion requires a class sample size near $p^2/2$ at minimum. Because of this, attention must be given to model parsimony – we recommend against blocking observations on a factor unless that factor has sufficent observations and significantly affects the observations' mean vector or covariance matrix.

5.4.3 Adaptive and Dynamic Modeling

Because of the non-linear, non-stationary, and autocorrelated nature of some process monitoring applications, the `mspTrain` function allows users to include lags of all feature variables. Including lags of the features in the data matrix (dynamic) can significantly reduce the negative effects of modeling autocorrelated observations. Further, the `mspTrain` function includes the option to update the training window over time. Re-estimating the projection matrix and test statistic thresholds at pre-specified time intervals (adaptive) can reduce the negative effects of non-linearity and non-stationarity when modeling the observations. The idea is to divide a non-linear and non-stationary process between some equally-spaced boundaries (every day, for instance), so that the process becomes locally linear and stationary within these boundaries. As time progresses, the oldest observations are "forgotten" and the newest observations are "learned". This causes the projection and IC thresholds to "adapt" over time.

5.4.4 Model Training

After the observations have been split by class, the `mspTrain` function will call the internal function `processMonitor`, which subsequently calls the internal function `faultFilter`. This function will calculate a linear projection matrix of the data by taking the PCA of the training data matrix. The observations will then be projected linearly into a reduced-feature subspace which preserves a chosen proportion of the energy of the training data, where the energy of a matrix is the relative sum of eigen-

values of that matrix. The default proportion is 90%. This projection is calculated by the internal `pca` function.

Furthermore, non-parametric threshold values are calculated for the two process monitoring statistics – *Squared Prediction Error* (SPE) and *Hotelling's* T^2 (T^2). These monitoring statistics are described in the motivating paper (chapter four). These threshold values are calculated by the internal `threshold` function, and passed up through the function pipe to be returned by `mspTrain`. The α-level of the non-parametric threshold is controlled by the user, but defaults to 0.001.

Finally, the `mspTrain` function will remove any observation(s) that would cause an alarm from the training data set. The alarm-free observations will be returned in one `xts` matrix, while the alarmed observations will be returned by another. The alarm criteria are discussed in Section 5.5. When training the process monitor, pay attention to any observations flagged as alarms. The proportion of observations flagged as faults may be higher than the α-level specified, so some tuning may be necessary. As with the `mspProcessData` function, a detailed description of the inputs and outputs from the `mspTrain` function are shown in Section 5.6.

5.5 Monitor and Issue Alarms with `mspMonitor` and `mspWarning`

After training the model with `mspTrain`, the projection matrices and non-parametric monitoring statistic thresholds can be used to flag incoming observations which are potentially out of control. The `mspMonitor` function was instead designed to test a single incoming observation at a time via a script or batch file, but *can* check every observation in a test matrix, which is useful for analyzing past data. To this end, the `mspMonitor` function projects a single observation with the class projection matrices returned by `mspTrain` and checks the observation's *SPE* and T^2 statistics against the thresholds also returned by `mspTrain`. The `mspMonitor` function will then append the monitoring statistic values and indicators to the end of the observation

row. Indicator values will be returned as a "1" if the values of these statistics exceed their IC thresholds. This new appended observation will be passed to the `mspWarning` function.

The `mspWarning` function takes in an observation returned by the `mspMonitor` function and an integer value (denoted r for this discussion) dictating the number of sequential flagged observations observed before an alarm is raised. If an observation returned by the `mspMonitor` function has positive statistic indicator values for either the SPE or T^2 monitoring statistics, then the `mspWarning` function will query the most recent r observations for other flags. If all r observations are positive for anomalies, then the `mspWarning` function will issue an alarm. The number of flags necessary to trigger an alarm defaults to 5. However, this default value depends heavily on the scale of the data: for continuous observations aggregated and recorded every five seconds, the number of sequential flags necessary to trigger an alarm could be much higher, perhaps even 20 or more. In contrast, for observations aggregated to the 10-minute-scale, only three sequential flags may be necessary.

In future updates of this package, this function will also have an option to issue an alarm if a critical mass of non-sequential flags is reached in a set period of observations. This modification may be necessary if the sampling frequency is less than the 1-minute-level. Additionally, this function will also be equipped to take in a cell phone number and service provider and issue an alarm via SMS through email. Detailed descriptions of the inputs and outputs from the `mspMonitor` and `mspWarning` functions are shown in the next section.

5.6 Example Simulation Workflow

This section provides a fully commented code walk-through for the main `msp.*` functions in the `mvMonitoring` package. This package depends on few other packages, and most of these dependencies are on commonly-used packages. The exception is `BMS`.

```
library(devtools)
library(xts)
# install.packages("BMS") # If necessary
build("C:/Users/gabriel_odom/Documents/GitHub/mvMonitoring/mvMonitoring")
## [1] "C:/Users/gabriel_odom/Documents/GitHub/mvMonitoring/
                                                mvMonitoring_0.1.1.tar.gz"
library(mvMonitoring)
```

5.6.1 Generating Synthetic Data

First, begin by generating multi-state data from a fault scenario. This code will yield observations under Fault 2A, as described in Section 5.3.3. We choose the default options for the period length (7 days * 24 hours * 60 minutes = 10,080 observations), the starting index of the fault (8500 out of 10080), and the time stamp for beginning the data is 16 May of 2016 at 10:00AM (my wedding anniversary). As we can see from the structure of the output, `mspProcessData` returns an `xts` matrix with 10080 rows and four columns (the state indicator and the three features).

```
fault1A_xts <- mspProcessData(faults = "A1",
                              period = 7 * 24 * 60,
                              faultStartIndex = 8500,
                              startTime = "2015-05-16 10:00:00 CST")
str(fault1A_xts)
## An 'xts' object on 2015-05-16 10:00:00/2015-05-23 09:59:00 containing:
##   Data: num [1:10080, 1:4] 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:4] "state" "x" "y" "z"
```

```
##   Indexed by objects of class: [POSIXct,POSIXt] TZ:
##   xts Attributes:
##  NULL
```

5.6.2 Train the Fault Detection Threshold

Now that these observations are generated and stored in memory, the `mspTrain` function can train the MSAD-PCA model. The last 1620 observations (27 hours' worth) will be saved for validation. The `mspTrain` function takes in the training data matrix and corresponding class label column. If this function errors, make sure the label column is not included in the data matrix – the constant values will cause a rank deficiency training covariance matrix. The `mspTrain` function will

(1) Train on the first three days' worth of observations, as set by `trainObs`.

(2) Scan the fourth day for anomalies, as set by `updateFreq`.

(3) Remove any alarmed observations.

(4) "Forget" the first day's observations.

(5) "Learn" the non-alarmed observations from the fourth day.

(6) Retrain and repeat until the end of the training matrix.

Furthermore, the `Dynamic = TRUE` option means that the `mspTrain` function will include the lags specified by the `lagsIncluded` argument. Finally, the number of sequential anomalous observations necessary to raise an alarm is set at 5 by the `faultsToTriggerAlarm` argument. These last three arguments are set to their defaults.

```
train1A_xts <- fault1A_xts[1:8461,]
# This function will run in 13 seconds on the author's machine.
train1A_ls <- mspTrain(data = train1A_xts[,-1],
```

```
                        labelVector = train1A_xts[,1],
                        trainObs = 3 * 24 * 60,
                        updateFreq = 1 * 24 * 60,
                        Dynamic = TRUE,
                        lagsIncluded = 0:1,
                        faultsToTriggerAlarm = 5)
str(train1A_ls)

## List of 4
##  $ FaultChecks    :An 'xts' object on 2015-05-19 08:59:00/2015-05-22
##                                                    07:00:00 containing:
##   Data: num [1:4142, 1:5] 0.364 0.425 0.821 1.004 0.864 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:5] "SPE" "SPE_Flag" "T2" "T2_Flag" ...
##   Indexed by objects of class: [POSIXct,POSIXt] TZ:
##   xts Attributes:
##  NULL
##  $ Non_Alarmed_Obs:An 'xts' object on 2015-05-19 08:59:00/2015-05-22
##                                                    07:00:00 containing:
##   Data: num [1:4142, 1:7] 2 3 1 1 1 1 1 1 1 1 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:7] "state" "x" "y" "z" ...
##   Indexed by objects of class: [POSIXct,POSIXt] TZ:
##   xts Attributes:
##  NULL
##  $ Alarms         :An 'xts' object of zero-width
```

```
##  $ TrainingSpecs  :List of 3
##   ..$ 1:List of 6
##   .. ..$ SPE_threshold   : Named num 4.7
##   .. .. ..- attr(*, "names")= chr "99.9%"
##   .. ..$ T2_threshold    : Named num 59.7
##   .. .. ..- attr(*, "names")= chr "99.9%"
##   .. ..$ projectionMatrix: num [1:6, 1:2] 0.424 -0.384 0.437 0.433
##                                                                -0.308 ...
##   .. ..$ LambdaInv       : num [1:2, 1:2] 0.206 0 0 1.513
##   .. ..$ muTrain         : Named num [1:6] 1.38 -2.09 2.96 1.37
##                                                             -2.05 ...
##   .. .. ..- attr(*, "names")= chr [1:6] "x" "y" "z" "x.1" ...
##   .. ..$ RootPrecisTrain : num [1:6, 1:6] 2.54 0 0 0 0 ...
##   .. ..- attr(*, "class")= chr [1:2] "threshold" "pca"
##   ..$ 2:List of 6
##   .. ..$ SPE_threshold   : Named num 1.35
##   .. .. ..- attr(*, "names")= chr "99.9%"
##   .. ..$ T2_threshold    : Named num 76.8
##   .. .. ..- attr(*, "names")= chr "99.9%"
##   .. ..$ projectionMatrix: num [1:6, 1:3] -0.4995 -0.0334 0.4931
##                                                    -0.5255 -0.1688 ...
##   .. ..$ LambdaInv       : num [1:3, 1:3] 0.285 0 0 0 0.72 ...
##   .. ..$ muTrain         : Named num [1:6] 2.977 -0.566 -4.498 2.951
##                                                          -0.591 ...
##   .. .. ..- attr(*, "names")= chr [1:6] "x" "y" "z" "x.1" ...
##   .. ..$ RootPrecisTrain : num [1:6, 1:6] 1.07 0 0 0 0 ...
##   .. ..- attr(*, "class")= chr [1:2] "threshold" "pca"
```

```
##   ..$ 3:List of 6
##   .. ..$ SPE_threshold   : Named num 5.85
##   .. .. ..- attr(*, "names")= chr "99.9%"
##   .. ..$ T2_threshold    : Named num 80.7
##   .. .. ..- attr(*, "names")= chr "99.9%"
##   .. ..$ projectionMatrix: num [1:6, 1:2] 0.396 0.454 0.454 -0.153
##                                                                0.461 ...
##   .. ..$ LambdaInv       : num [1:2, 1:2] 0.276 0 0 0.499
##   .. ..$ muTrain         : Named num [1:6] 0.528 0.272 1.44 0.568
##                                                                0.258 ...
##   .. .. ..- attr(*, "names")= chr [1:6] "x" "y" "z" "x.1" ...
##   .. ..$ RootPrecisTrain : num [1:6, 1:6] 16.4 0 0 0 0 ...
##   .. ..- attr(*, "class")= chr [1:2] "threshold" "pca"
```

The `mspTrain` function returns a list of four objects:

(1) `FaultChecks`: An `xts` matrix of monitoring statistics and associated indicators for all observations after the burn-in of `trainObs`. It will have 8461 - `trainObs` number of rows and five columns:

   (a) `SPE`: the SPE statistic for each observation.

   (b) `SPE_Flag`: an indicator showing if the SPE statistic for that observation is beyond the calculated threshold; 0 is normal, 1 is flagged.

   (c) `T2`: the T^2 statistic for each observation.

   (d) `T2_Flag`: an indicator showing if the T^2 statistic for that observation is beyond the calculated threshold; 0 is normal, 1 is flagged.

   (e) `Alarm`: an indicator showing if the observation is in a sequence of flagged observations; 0 is normal, 1 is alarmed.

(2) `Non_Alarmed_Obs`: An `xts` matrix containing all observations with an alarm code of 0 from `FaultChecks`. Of note, this matrix contains the data, while the `FaultChecks` matrix only contains the monitoring statistics and indicators.

(3) `Alarms`: An `xts` matrix of all the observations removed from the training data matrix.

(4) `TrainingSpecs`: a list with length equal to the number of classes – in this case 3. For each class, this list contains a list of six objects:

   (a) `SPE_Threshold`: a named numeric scalar of the $1-\alpha$ percentile of the non-parametric estimate of the SPE statistic density.

   (b) `T2_Threshold`: a named numeric scalar of the $1-\alpha$ percentile of the non-parametric estimate of the T^2 statistic density.

   (c) `projectionMatrix`: The $p \times q$ matrix of eigenvectors necessary to project a p-dimensional observation to q dimensions. This is necessary to reduce the dimension of any test observation, and is used in calculating the SPE statistic for test observations.

   (d) `LambdaInv`: The inverse of the diagonal $q \times q$ matrix of eigenvalues. This matrix is used to calculate the T^2 statistic for test observations.

   (e) `muTrain`: The mean vector of the training observations. This is used to center the test observations on the training mean.

   (f) `RootPrecisTrain`: The $p \times p$ diagonal matrix of the inverse square roots of the feature variances. This is used to scale the test observations into the training scale.

5.6.3 Test New Observations for Anomalies

The training data summary was given by `mspTrain`, so this information can now be used to monitor incoming observations for system faults.

5.6.3.1 *Adding lagged features.* First, concatenate the last given observation from the training set as "row 0" of the test data set. This will enable `mspMonitor` to include lag-1 features. Similarly, one would include the last k observations of the training set should the process dictate the need for any lag-k features. Because the Fault Start Index was set to 8500, this testing window will show the change point between observations generated under normal conditions and those generated under a fault state.

```
test1A_xts <- fault1A_xts[8460:8520, -1]
lagTest1A_xts <- lag.xts(test1A_xts, 0:1)
lagTest1A_xts <- cbind(fault1A_xts[8461:8520,1],
                       lagTest1A_xts[-1,])
head(lagTest1A_xts)
##                     state         x         y        z       x.1
## 2015-05-22 07:00:00     1 0.7465717 -1.582803 1.141042 0.3817970
## 2015-05-22 07:01:00     1 0.8064225 -1.564885 1.226738 0.7465717
## 2015-05-22 07:02:00     1 0.7621322 -1.603962 1.150504 0.8064225
## 2015-05-22 07:03:00     1 0.7995411 -1.785750 1.149927 0.7621322
## 2015-05-22 07:04:00     1 0.8638972 -1.822695 1.116856 0.7995411
## 2015-05-22 07:05:00     1 0.7735952 -1.505855 1.139623 0.8638972
##                           y.1       z.1
## 2015-05-22 07:00:00  0.1052954 0.3582939
## 2015-05-22 07:01:00 -1.5828027 1.1410421
## 2015-05-22 07:02:00 -1.5648852 1.2267384
## 2015-05-22 07:03:00 -1.6039621 1.1505043
## 2015-05-22 07:04:00 -1.7857497 1.1499272
## 2015-05-22 07:05:00 -1.8226952 1.1168557
```

5.6.3.2 *Monitoring the test data.* With the lagged test observations in the working environment, the `mspMonitor` function can be applied. This function (similarly to `mspTrain`) takes in the label information as a separate argument from the input data. Further, the `mspMonitor` function takes in the `TrainingSpecs` object returned in the results list from `mspTrain`. Notice that the first six rows of the matrix returned by `mspMonitor` are the exact same as the first six rows of the lagged test matrix, except that the rows of the matrix from the `mspMonitor` function have the monitoring statistics and corresponding indicator columns appended.

```
monitor1A_xts <- mspMonitor(observations = lagTest1A_xts[,-1],
                            labelVector = lagTest1A_xts[,1],
                            trainingSummary = train1A_ls$TrainingSpecs)
head(monitor1A_xts)
##                             x         y        z       x.1        y.1
## 2015-05-22 07:00:00 0.7465717 -1.582803 1.141042 0.3817970  0.1052954
## 2015-05-22 07:01:00 0.8064225 -1.564885 1.226738 0.7465717 -1.5828027
## 2015-05-22 07:02:00 0.7621322 -1.603962 1.150504 0.8064225 -1.5648852
## 2015-05-22 07:03:00 0.7995411 -1.785750 1.149927 0.7621322 -1.6039621
## 2015-05-22 07:04:00 0.8638972 -1.822695 1.116856 0.7995411 -1.7857497
## 2015-05-22 07:05:00 0.7735952 -1.505855 1.139623 0.8638972 -1.8226952
##                            z.1       SPE SPE_Flag        T2 T2_Flag Alarm
## 2015-05-22 07:00:00 0.3582939 0.3441627        0 29.445080       0    NA
## 2015-05-22 07:01:00 1.1410421 0.3811468        0  3.329163       0    NA
## 2015-05-22 07:02:00 1.2267384 0.2937754        0  3.222249       0    NA
## 2015-05-22 07:03:00 1.1505043 0.1424375        0  2.810259       0    NA
## 2015-05-22 07:04:00 1.1499272 0.2947956        0  2.630555       0    NA
## 2015-05-22 07:05:00 1.1168557 0.8742699        0  3.708552       0    NA
```

5.6.4 Warn Operators During Alarms

Note that all values in the `Alarm` column of the returned matrix above are recorded with `NA` values. This is because the `mspMonitor` function does not check the monitor output matrix for the sequential flags necessary to trigger an alarm. This is the responsibility of the `mspWarning` function. Because the `mspWarning` function is designed to test one incoming observation at a time through a script or batch file, the following example is designed to mimic the behavior of the `mspWarning` function as each new observation comes online.

```
alarm1A_xts <- monitor1A_xts
for(i in 1:nrow(alarm1A_xts)){
  if(i < (5 + 1)){
    alarm1A_xts[1:i,] <- mspWarning(alarm1A_xts[1:i,])
  }else{
    alarm1A_xts[(i - 5):i,] <- mspWarning(alarm1A_xts[(i - 5):i,])
  }
}
```

The fault was introduced at index 8500, which corresponds to about 40 minutes into the test hour.

```
plot(alarm1A_xts[, ncol(alarm1A_xts)],
     main = "Alarm Codes for Test Data")
```

The alarm codes are

(1) "0": No alarm.

(2) "1": Hotelling's T^2 alarm.

(3) "2": Squared Prediction Error alarm.

(4) "3": Both alarms.

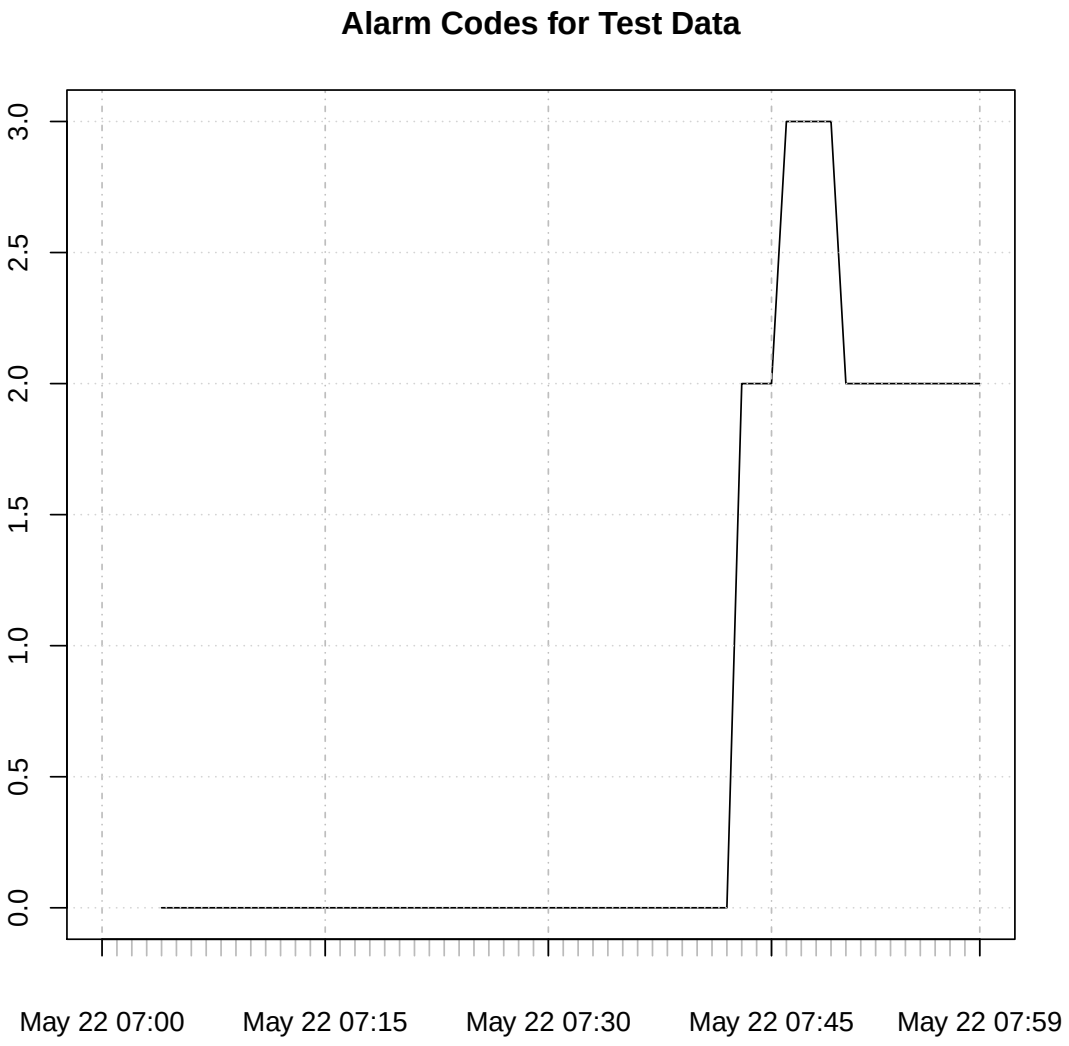


Figure 5.1: Time series plot of alarms. The fault was introduced at 07:40 on 22 May.

As we can see, the monitoring function detects a process anomaly after 40 minutes into the test hour, and the warning function issues the corresponding alarms.

5.7 Conclusion

This vignette supplies motivation for the `mvMonitoring` package and discusses implementing a multivariate process monitoring scheme with this package using synthetic data. We believe that this software will provide system engineers with the tools necessary to quickly and accurately detect abnormalities in multivariate, autocorrelated, non-stationary, non-linear, and multi-state industrial and engineering systems. Further, a synthetic example is given showing how the functions within this package would be implemented and tuned in practice.

CHAPTER SIX

Discussion

As the breadth of science increases, so will the dimensionality of recorded data. Around 80 years ago, R.A. Fisher introduced the world to his four-dimensional "Iris" data set. Today, we look to unlock the secrets of the human genome containing tens of thousands of protein-coding genes, or query information from the million-dimensional Internet of Things for patterns and hidden interactions. We believe that as these "big data" problems grow more complex, new techniques in linear dimension reduction will become more and more valuable. In this dissertation, we briefly scratched the surface of one such linear dimension reduction method.

In chapter two, we discussed improving the sample quadratic classifier by replacing a sample covariance estimator with a regularized covariance estimator for cases where the number of observations is only slightly larger than the original feature dimension. We showed that estimating sample covariance matrices with a regularized estimator can improve classification performance. In chapter three, we compared the computational costs associated with linear dimension reduction for ill-posed cases using principal component analysis and the singular value decomposition. We show that these two methods yield nearly identical classification results, but the singular value decomposition can be thousands of times less computationally expensive.

In chapter four, we modified adaptive-dynamic principal component analysis to account for known process state information. We show that correctly accounting for process states decreases fault detection times and increases fault detection consistency, while not suffering a decrease in power if process data do not have multiple states. Further, in chapter five we describe the `mvMonitoring` package designed to

make multi-state (and single-state) Adaptive-Dynamic PCA for multivariate statistical process monitoring available for `R` users. We also show in chapter five some code examples necessary to reproduce the simulations in chapter four. We believe this package will provide necessary and timely assistance to wastewater treatment quality control engineers.

We summarize our contributions to two main camps within the statistical literature. The first suite of contributions are to the field of supervised classification under the constraints of insufficient observations. We apply a covariance matrix estimator, designed for matrix inversion with decreased estimator variability, to the task of classifying observations from a data-poor environment. This result will better equip scientists to predict the group membership of future observations, even under a small sample-size constraint. Furthermore, we provide a computational comparison of two popular linear dimension reduction routines, principal component analysis and the singular value decomposition, and settle conclusively that the singular value decomposition is superior to principal component analysis for supervised classification in high-dimensional, low sample-size contexts. The second suite of contributions we make is to the field of multivariate statistical process control. We provide a new quasi-linear dimension reduction approach specifically designed to assist engineers in monitoring a potentially non-Gaussian, non-linear, non-stationary, and autocorrelated multivariate process drawn from disparate system states. Additionally, we give a working example of software we designed and successfully implemented in practice that can be useful to perform such process monitoring.

Our aims for these discussed contributions are as follows. For heteroscedastic linear discriminant analysis, we expect data scientists to further apply different covariance estimators to supervised classification exercises under the assumption of unequal covariance matrices. While we believe that the Haff Shrinkage Estimator we employed is an appropriate covariance estimator, there may be other covariance or

precision estimators with more attractive properties. Further, we hope that these answers themselves raise additional research questions about the proper application of covariance and precision matrix estimators in other branches of statistical modeling, such as multi-dimensional hypothesis testing. Concerning our benchmark comparison of principal component analysis and the singular value decomposition when applied to ill-posed supervised classification exercises, we hope that computer scientists continue to improve the algorithms associated with these and other matrix decompositions. We also hope that data scientists working in high-dimensional genetics will leave principal component analysis behind in favor of the superior singular value decomposition for their dimension reduction needs. Furthermore, we believe that our work will motivate statisticians to further study the theoretical properties of the singular value decomposition when applied to ill-posed observations. We believe such research will bolster the breadth of theoretical results associated with the singular value decomposition and upgrade it from a clever result to the *de facto* method for linear dimension reduction in ill-posed cases. Finally, we hope that environmental chemists and engineers will be able to use our software package `mvMonitoring`, and by extension the multi-state monitoring adaptation to adaptive-dynamic principal component analysis, to great success in monitoring the behavior of their multivariate processes.

As with any piece of academic literature, we had too many good questions to ask and too little time to answer them. We have further research interests in comparing other covariance or precision estimators to round out our work in chapter two, including what we have termed the "sparse correlation covariance estimator". Additionally, concerning our work on multivariate statistical process control, our immediate next step is toward fault attribution; that is, given that our method detects a fault, can we ascertain what circumstance may have caused it? The "great leap forward" after accurate fault detection and attribution is fault intervention, wherein the system is trained to apply small remedies to correct negative fault effects after their identifica-

tion. Ultimately, the final goal of this work is complete automation of a decentralized wastewater treatment process.

While altruism should receive lofty praise, and passion for science for science's own sake can be a virtue—in the style of *ars gratia artis*, the true purpose of this dissertation is the gainful employment of its author. To this end, the author summarizes his efforts for any future employers or industry subject-matter experts. This dissertation shows that the author is capable of identifying novel research areas and reading the appropriate literature necessary to familiarize himself with their connected branches of science. Further, this manuscript demonstrates the author's perseverance and diligence in pursuit of a goal, which are characteristics readily applicable to securing external funding, completing a full-spectrum industrial project, and/or advancing the cause of science as a whole. Specifically, and as shown by his list of intra- and interdisciplinary co-authors, this author is well-equipped to succeed in a university, medical center, or industrial research environment. Moreover, this treatise exhibits plainly the author's excellent written communication ability, a skill often underrepresented by practitioners of the hard sciences. Finally, this completed work is evidentiary of the years of trial, error, success, and experience with which the author is now invested.

APPENDICES

APPENDIX A

Selected Proofs

Lemma 1. *Let* $\boldsymbol{W} \in \mathbb{R}_{p\times s}$*, where* $s := (m-1)(p+1)$*, be*

$$\boldsymbol{W} := [\boldsymbol{g}_2 - \boldsymbol{g}_1 | \dots | \boldsymbol{g}_m - \boldsymbol{g}_1 | \boldsymbol{H}_2 - \boldsymbol{H}_1 | \dots | \boldsymbol{H}_m - \boldsymbol{H}_1], \tag{A.1}$$

where $\boldsymbol{g}_i \in \mathbb{R}_{p\times 1}$*,* $\boldsymbol{H}_i \in \mathbb{R}_p^>$ *and is symmetric, and* $\boldsymbol{g}_1 \neq \boldsymbol{g}_k$ *and* $\boldsymbol{H}_1 \neq \boldsymbol{H}_k$ *for at least one value of* k*, where* $2 \leq k \leq m$ *and* $i = 1, \dots, m$*. Also, let rank*$(\boldsymbol{W}) = 1 \leq q < p$*, and let* $\boldsymbol{F} \in \mathbb{R}_{p\times q}$ *and* $\boldsymbol{G} \in \mathbb{R}_{q\times s}$ *be matrix components of a full-rank decomposition of* $\boldsymbol{W}$ *so that* $\boldsymbol{W} = \boldsymbol{F}\boldsymbol{G}$ *with rank*$(\boldsymbol{F}) = q$*. Then,*

(a) $\boldsymbol{F}\boldsymbol{F}^+ (\boldsymbol{g}_i - \boldsymbol{g}_1) = \boldsymbol{g}_i - \boldsymbol{g}_1$*;* $\boldsymbol{F}\boldsymbol{F}^+ (\boldsymbol{H}_i - \boldsymbol{H}_1) = \boldsymbol{H}_i - \boldsymbol{H}_1$

(b) $\left(\boldsymbol{I}_p - \boldsymbol{F}\boldsymbol{F}^+\right)(\boldsymbol{g}_i - \boldsymbol{g}_1) = \boldsymbol{0}$*;* $\left(\boldsymbol{I}_p - \boldsymbol{F}\boldsymbol{F}^+\right)(\boldsymbol{H}_i - \boldsymbol{H}_1) = \boldsymbol{0}$

(c) $\boldsymbol{F}\boldsymbol{F}^+ (\boldsymbol{H}_i - \boldsymbol{H}_1) = (\boldsymbol{H}_i - \boldsymbol{H}_1)\boldsymbol{F}\boldsymbol{F}^+$

(d) $\boldsymbol{F}\boldsymbol{F}^+ \boldsymbol{H}_i = \boldsymbol{H}_i \boldsymbol{F}\boldsymbol{F}^+$*, and*

(e) $\left(\boldsymbol{I} - \boldsymbol{F}\boldsymbol{F}^+\right)\boldsymbol{H}_i = \boldsymbol{H}_1\left(\boldsymbol{I} - \boldsymbol{F}\boldsymbol{F}^+\right)$.

Proof. For parts (a) and (b), note that $\mathbf{g}_i - \mathbf{g}_1$, $\mathbf{H}_i - \mathbf{H}_1 \in \mathcal{C}(\mathbf{F})$, and $\mathbf{g}_i - \mathbf{g}_1$, $\mathbf{H}_i - \mathbf{H}_1 \in \mathcal{N}(\mathbf{I}_p - \mathbf{F}\mathbf{F}^+)$. For part (c), recalling that $\mathbf{H}_i \in \mathbb{S}_p$,

$$\mathbf{F}\mathbf{F}^+ [\mathbf{H}_i - \mathbf{H}_1] = [\mathbf{H}_i - \mathbf{H}_1]^T = [\mathbf{H}_i - \mathbf{H}_1]\mathbf{F}\mathbf{F}^+.$$

For part (d), recall that for $\mathbf{x} \in \mathbb{R}_{p\times 1}$, $\mathbf{x}^T\mathbf{F}\mathbf{F}^+$ projects $\mathbf{x}$ onto the row space of $\mathbf{F}\mathbf{F}^+$. Because $\mathbf{F}\mathbf{F}^+ \in \mathbb{S}_p$, the column space and row space are equal. Thus, $\mathbf{F}\mathbf{F}^+\mathbf{H}_i = \mathbf{H}_i\mathbf{F}\mathbf{F}^+$. Finally, for (e), we have by parts (b) and (d) that

$$\begin{aligned}
\left(\mathbf{I}_p - \mathbf{F}\mathbf{F}^+\right)(\mathbf{H}_i - \mathbf{H}_1) = \mathbf{0} \quad &\iff \quad \left(\mathbf{I}_p - \mathbf{F}\mathbf{F}^+\right)\mathbf{H}_i = \mathbf{H}_1 - \mathbf{F}\mathbf{F}^+\mathbf{H}_1 \\
&\iff \quad \left(\mathbf{I}_p - \mathbf{F}\mathbf{F}^+\right)\mathbf{H}_i = \mathbf{H}_1 - \mathbf{H}_1\mathbf{F}\mathbf{F}^+ \\
&\iff \quad \left(\mathbf{I}_p - \mathbf{F}\mathbf{F}^+\right)\mathbf{H}_i = \mathbf{H}_1\left(\mathbf{I}_p - \mathbf{F}\mathbf{F}^+\right).
\end{aligned}$$

□

Lemma 2. *Consider the matrices* $\boldsymbol{F}$ *and* $\boldsymbol{H}_i$, $i = 1, \ldots, m$, *defined in Lemma 1. Then,* $\left[\boldsymbol{F}^{+}\boldsymbol{H}_i\boldsymbol{F}^{+T}\right]^{-1} = \boldsymbol{F}^{T}\boldsymbol{H}_i^{-1}\boldsymbol{F}$.

Proof. Let $\mathbf{P} := \mathbf{F}\mathbf{F}^{+}\mathbf{H}_i\mathbf{F}\mathbf{F}^{+}$ and $\mathbf{Q} := \mathbf{F}\mathbf{F}^{+}\mathbf{H}_i^{-1}\mathbf{F}\mathbf{F}^{+}$. Using Lemma 1(d),

$$\begin{aligned}(\mathbf{PQ})^T &= \left[\left(\mathbf{F}\mathbf{F}^{+}\mathbf{H}_i\mathbf{F}\mathbf{F}^{+}\right)\left(\mathbf{F}\mathbf{F}^{+}\mathbf{H}_i^{-1}\mathbf{F}\mathbf{F}^{+}\right)\right]^T \\ &= \mathbf{F}\mathbf{F}^{+}\mathbf{H}_i\mathbf{F}\mathbf{F}^{+}\mathbf{H}_i^{-1}\mathbf{F}\mathbf{F}^{+} \\ &= \mathbf{PQ}.\end{aligned}$$

Therefore, $\mathbf{PQ}$ and $(\mathbf{F}^{+})^T\mathbf{F}^{+}\mathbf{PQ}$ are symmetric. Similarly, $\mathbf{QP}(\mathbf{F}^{+})^T\mathbf{F}^{+}$ and $\mathbf{QP}$ are symmetric. Further, using Lemma 1(d), $\mathbf{PQP} = \mathbf{P}$ and $\mathbf{QPQ} = \mathbf{Q}$; thus $\mathbf{Q} = \mathbf{P}^{+}$. Therefore, by Theorem 1 (i) and (ii) of Hartwig (1986), we establish our result. □

Lemma 3. *Consider* $\boldsymbol{F}$, $\boldsymbol{g}_i$ *and* $\boldsymbol{H}_i$, $i = 1, \ldots, m$, *defined in Lemma 1. Also, let* $\boldsymbol{C} = \boldsymbol{R}\left[\boldsymbol{I} - \boldsymbol{F}\boldsymbol{F}^{+}\right] \in \mathbb{R}_{(p-q)\times p}$, *where* $\boldsymbol{R} \in \mathbb{R}_{(p-q)\times p}$ *such that rank*$(\boldsymbol{C}) = p - q$. *We have that*

(a) $\boldsymbol{C}\boldsymbol{g}_i = \boldsymbol{C}\boldsymbol{g}_1$,

(b) $\boldsymbol{C}\boldsymbol{H}_i\boldsymbol{C}^{T} - \boldsymbol{C}\boldsymbol{H}_i\boldsymbol{F}^{+T}\left(\boldsymbol{F}^{+}\boldsymbol{H}_i\boldsymbol{F}^{+T}\right)^{-1}\boldsymbol{F}^{+}\boldsymbol{H}_i\boldsymbol{C}^{T} = \boldsymbol{C}\boldsymbol{H}_1\boldsymbol{C}^{T}$.

Proof. The proof of (a) follows trivially from Lemma 1(b). Also, by Lemma 1(d), note that

$$\begin{aligned}\mathbf{F}^{+}\mathbf{H}_i\mathbf{C}^T &= \mathbf{F}^{+}\mathbf{H}_i\left(\mathbf{I}_p - \mathbf{F}\mathbf{F}^{+}\right)^T\mathbf{R}^T \\ &= \left(\mathbf{F}^{+}\mathbf{H}_i - \mathbf{F}^{+}\mathbf{H}_i\mathbf{F}\mathbf{F}^{+}\right)\mathbf{R}^T \\ &= \mathbf{0}.\end{aligned}$$

Then by Lemma 1 (b), $\mathbf{C}\mathbf{H}_i\mathbf{C}^T = \mathbf{C}\mathbf{H}_1\mathbf{C}^T$. □

APPENDIX B

Simulation Parameter Configuration

2.0.1 Mean Vectors

Table B.1: Mean vectors for simulations 1–3.

$\boldsymbol{\mu}_{11}$	(0.21,	-0.37,	2.29,	1.66,	-0.81,	0.65,	0.18,	0.84,	1.88,	0.56)^T
$\boldsymbol{\mu}_{12}$	(4.64,	4.06,	6.72,	6.09,	3.62,	5.08,	4.61,	5.27,	6.31,	4.99)^T
$\boldsymbol{\mu}_{13}$	(8.21,	7.63,	10.29,	9.66,	7.19,	8.65,	8.18,	8.84,	9.88,	8.56)^T
$\boldsymbol{\mu}_{21}$	(-1.43,	-0.66,	-0.94,	0.31,	-0.19,	0.89,	0.25,	-0.34,	1.25,	-1.60)^T
$\boldsymbol{\mu}_{22}$	(-0.43,	0.34,	0.06,	1.31,	0.81,	1.89,	1.25,	0.66,	2.25,	-0.60)^T
$\boldsymbol{\mu}_{23}$	(0.57,	1.34,	1.06,	2.31,	1.81,	2.89,	2.25,	1.66,	3.25,	0.40)^T
$\boldsymbol{\mu}_{31}$	(-0.15,	-0.96,	1.67,	0.70,	-0.25,	-2.54,	-1.67,	2.00,	1.11,	-0.86)^T
$\boldsymbol{\mu}_{32}$	(0.85,	-0.96,	2.67,	0.70,	0.75,	-2.54,	-0.67,	2.00,	2.11,	-0.86)^T
$\boldsymbol{\mu}_{33}$	(-1.15,	-1.96,	0.67,	-0.30,	-1.25,	-3.54,	-2.67,	1.00,	0.11,	-1.86)^T

2.0.2 Configuration 1 Covariance Matrices

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} 15.01 & 0.81 & 1.25 & 1.13 & -2.10 & -2.30 & 2.43 & -3.30 & -0.87 & -1.11 \\ 0.81 & 26.10 & 1.51 & -0.74 & 0.89 & 4.35 & 1.25 & 1.85 & -0.50 & -0.39 \\ 1.25 & 1.51 & 24.55 & -5.57 & 3.96 & 1.62 & -0.27 & 0.49 & -5.97 & 0.87 \\ 1.13 & -0.74 & -5.57 & 29.36 & -3.58 & -0.89 & 2.21 & -3.71 & -0.52 & -2.19 \\ -2.10 & 0.89 & 3.96 & -3.58 & 20.17 & 6.05 & -5.20 & 2.22 & -0.80 & -2.69 \\ -2.30 & 4.35 & 1.62 & -0.89 & 6.05 & 40.18 & -5.18 & 3.83 & 1.94 & 0.51 \\ 2.43 & 1.25 & -0.27 & 2.21 & -5.20 & -5.18 & 17.93 & -0.17 & -3.09 & 0.49 \\ -3.30 & 1.85 & 0.49 & -3.71 & 2.22 & 3.83 & -0.17 & 26.05 & -0.54 & 3.34 \\ -0.87 & -0.50 & -5.97 & -0.52 & -0.80 & 1.94 & -3.09 & -0.54 & 16.3 & -1.04 \\ -1.11 & -0.39 & 0.87 & -2.19 & -2.69 & 0.51 & 0.49 & 3.34 & -1.04 & 26.84 \end{bmatrix},$$

$$\boldsymbol{\Sigma}_2 = \begin{bmatrix} 45.01 & 30.81 & 31.25 & 31.13 & 27.90 & 27.70 & 32.43 & 26.70 & 29.13 & 28.89 \\ 30.81 & 56.10 & 31.51 & 29.26 & 30.89 & 34.35 & 31.25 & 31.85 & 29.50 & 29.61 \\ 31.25 & 31.51 & 54.55 & 24.43 & 33.96 & 31.62 & 29.73 & 30.49 & 24.03 & 30.87 \\ 31.13 & 29.26 & 24.43 & 59.36 & 26.42 & 29.11 & 32.21 & 26.29 & 29.48 & 27.81 \\ 27.90 & 30.89 & 33.96 & 26.42 & 50.17 & 36.05 & 24.80 & 32.22 & 29.20 & 27.31 \\ 27.70 & 34.35 & 31.62 & 29.11 & 36.05 & 70.18 & 24.82 & 33.83 & 31.94 & 30.51 \\ 32.43 & 31.25 & 29.73 & 32.21 & 24.80 & 24.82 & 47.93 & 29.83 & 26.91 & 30.49 \\ 26.70 & 31.85 & 30.49 & 26.29 & 32.22 & 33.83 & 29.83 & 56.05 & 29.46 & 33.34 \\ 29.13 & 29.50 & 24.03 & 29.48 & 29.20 & 31.94 & 26.91 & 29.46 & 46.30 & 28.96 \\ 28.89 & 29.61 & 30.87 & 27.81 & 27.31 & 30.51 & 30.49 & 33.34 & 28.96 & 56.84 \end{bmatrix}$$

$$\boldsymbol{\Sigma}_3 = \begin{bmatrix} 75.01 & 60.81 & 61.25 & 61.13 & 57.9 & 57.7 & 62.43 & 56.70 & 59.13 & 58.89 \\ 60.81 & 86.10 & 61.51 & 59.26 & 60.89 & 64.35 & 61.25 & 61.85 & 59.50 & 59.61 \\ 61.25 & 61.51 & 84.55 & 54.43 & 63.96 & 61.62 & 59.73 & 60.49 & 54.03 & 60.87 \\ 61.13 & 59.26 & 54.43 & 89.36 & 56.42 & 59.11 & 62.21 & 56.29 & 59.48 & 57.81 \\ 57.90 & 60.89 & 63.96 & 56.42 & 80.17 & 66.05 & 54.80 & 62.22 & 59.20 & 57.31 \\ 57.70 & 64.35 & 61.62 & 59.11 & 66.05 & 100.18 & 54.82 & 63.83 & 61.94 & 60.51 \\ 62.43 & 61.25 & 59.73 & 62.21 & 54.80 & 54.82 & 77.93 & 59.83 & 56.91 & 60.49 \\ 56.70 & 61.85 & 60.49 & 56.29 & 62.22 & 63.83 & 59.83 & 86.05 & 59.46 & 63.34 \\ 59.13 & 59.50 & 54.03 & 59.48 & 59.20 & 61.94 & 56.91 & 59.46 & 76.30 & 58.96 \\ 58.89 & 59.61 & 60.87 & 57.81 & 57.31 & 60.51 & 60.49 & 63.34 & 58.96 & 86.84 \end{bmatrix}.$$

2.0.3 Configuration 2 Covariance Matrices

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix},$$

$$\boldsymbol{\Sigma}_2 = \begin{bmatrix} 2.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 2.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 2.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 2.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 2.0 \end{bmatrix},$$

$$\boldsymbol{\Sigma}_3 = \begin{bmatrix} 2.0 & 1.0 & 0.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 2.0 & 0.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 0.0 & 0.0 & 10.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 1.0 & 1.0 & 0.0 & 2.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 0.0 & 1.0 & 2.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 0.0 & 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 0.0 & 1.0 & 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 0.0 & 1.0 & 1.0 & 1.0 & 1.0 & 2.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 0.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 2.0 & 1.0 \\ 1.0 & 1.0 & 0.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 2.0 \end{bmatrix}$$

2.0.4 Configuration 3 Covariance Matrices

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} 2.0 & 0.8 & 1.0 & 0.8 & 0.6 & 0.8 & 0.8 & 1.0 & 0.8 & 0.6 \\ 0.8 & 2.0 & 1.0 & 0.4 & 0.8 & 0.6 & 0.6 & 1.0 & 0.8 & 0.6 \\ 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 0.6 & 0.8 & 0.8 & 0.8 & 0.8 \\ 0.8 & 0.4 & 1.0 & 2.0 & 0.6 & 0.8 & 0.4 & 0.6 & 0.8 & 0.6 \\ 0.6 & 0.8 & 1.0 & 0.6 & 2.0 & 0.6 & 0.8 & 1.0 & 0.6 & 0.6 \\ 0.8 & 0.6 & 0.6 & 0.8 & 0.6 & 2.4 & 0.6 & 0.8 & 0.8 & 0.8 \\ 0.8 & 0.6 & 0.8 & 0.4 & 0.8 & 0.6 & 2.8 & 0.4 & 0.4 & 0.4 \\ 1.0 & 1.0 & 0.8 & 0.6 & 1.0 & 0.8 & 0.4 & 2.4 & -0.1 & -0.1 \\ 0.8 & 0.8 & 0.8 & 0.8 & 0.6 & 0.8 & 0.4 & -0.1 & 2.8 & -0.2 \\ 0.6 & 0.6 & 0.8 & 0.6 & 0.6 & 0.8 & 0.4 & -0.1 & -0.2 & 2.2 \end{bmatrix},$$

$$\boldsymbol{\Sigma}_2 = \begin{bmatrix} 20.0 & 0.8 & 1.0 & 0.8 & 0.6 & 0.8 & 0.8 & 1.0 & 0.8 & 0.6 \\ 0.8 & 40.0 & 1.0 & 0.4 & 0.8 & 0.6 & 0.6 & 1.0 & 0.8 & 0.6 \\ 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 0.6 & 0.8 & 0.8 & 0.8 & 0.8 \\ 0.8 & 0.4 & 1.0 & 2.0 & 0.6 & 0.8 & 0.4 & 0.6 & 0.8 & 0.6 \\ 0.6 & 0.8 & 1.0 & 0.6 & 2.0 & 0.6 & 0.8 & 1.0 & 0.6 & 0.6 \\ 0.8 & 0.6 & 0.6 & 0.8 & 0.6 & 2.4 & 0.6 & 0.8 & 0.8 & 0.8 \\ 0.8 & 0.6 & 0.8 & 0.4 & 0.8 & 0.6 & 2.8 & 0.4 & 0.4 & 0.4 \\ 1.0 & 1.0 & 0.8 & 0.6 & 1.0 & 0.8 & 0.4 & 2.4 & -0.1 & -0.1 \\ 0.8 & 0.8 & 0.8 & 0.8 & 0.6 & 0.8 & 0.4 & -0.1 & 2.8 & -0.2 \\ 0.6 & 0.6 & 0.8 & 0.6 & 0.6 & 0.8 & 0.4 & -0.1 & -0.2 & 2.2 \end{bmatrix}$$

$$\boldsymbol{\Sigma}_3 = \begin{bmatrix} 35.0 & 0.8 & 1.0 & 0.8 & 0.6 & 0.8 & 0.8 & 1.0 & 0.8 & 0.6 \\ 0.8 & 40.0 & 1.0 & 0.4 & 0.8 & 0.6 & 0.6 & 1.0 & 0.8 & 0.6 \\ 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 0.6 & 0.8 & 0.8 & 0.8 & 0.8 \\ 0.8 & 0.4 & 1.0 & 2.0 & 0.6 & 0.8 & 0.4 & 0.6 & 0.8 & 0.6 \\ 0.6 & 0.8 & 1.0 & 0.6 & 2.0 & 0.6 & 0.8 & 1.0 & 0.6 & 0.6 \\ 0.8 & 0.6 & 0.6 & 0.8 & 0.6 & 2.4 & 0.6 & 0.8 & 0.8 & 0.8 \\ 0.8 & 0.6 & 0.8 & 0.4 & 0.8 & 0.6 & 2.8 & 0.4 & 0.4 & 0.4 \\ 1.0 & 1.0 & 0.8 & 0.6 & 1.0 & 0.8 & 0.4 & 2.4 & -0.1 & -0.1 \\ 0.8 & 0.8 & 0.8 & 0.8 & 0.6 & 0.8 & 0.4 & -0.1 & 2.8 & -0.2 \\ 0.6 & 0.6 & 0.8 & 0.6 & 0.6 & 0.8 & 0.4 & -0.1 & -0.2 & 2.2 \end{bmatrix}$$

REFERENCES

Aflalo, Y., & Kimmel, R. (2013). Spectral multidimensional scaling. *PNAS*, *110*(45), 18052–18057.

Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D., & Levine, A. J. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci. U.S.A.*, *96*(12), 6745–6750.

Anderson, T. (1951). Classification by multivariate analysis. *Psychometrika*, *16*(1), 31 – 50.

Apley, D. W., & Lee, H. C. (2012). Design of EWMA control charts for autocorrelated processes with model uncertainty. *Technometrics*, *45*(3), 187–198.

Apley, D. W., & Shi, J. (1999). The GLRT for statistical process control of autocorrelated processes. *IIE Trans.*, *31*(12), 1123–1134.

Apley, D. W., & Tsung, F. (2002). The autoregressive T2 chart for monitoring univariate autocorrelated processes. *J. Qual. Technol.*, *34*(1), 80–96.

Asadzadeh, S., Aghaie, A., & Yang, S. F. (2008). Monitoring and diagnosing multistage processes: A review of cause selecting control charts. *J. Ind. Sys. Eng.*, *2*(3), 214–235.

Bache, K., & Lichman, M. (2013). *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences.

Baggiani, F., & Marsili-Libelli, S. (2009). Real-time fault detection and isolation in biological wastewater treatment plants. *Water Sci. Technol.*, *60*(11), 2949–2961.

Barker, M., & Rayens, W. (2003). Partial least squares for discrimination. *J. Chemometr.*, *17*(3), 166–173.

Belkin, M., & Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, *15*(6), 1373–1396.

Bellman, R. (1961). *Adaptive Control Processes: A Guided Tour*. Princeton University Press.

Bengio, Y., Delalleau, O., Le Roux, N., Paiement, J. F., Vincent, P., & Ouimet, M. (2004). Learning eigenfunctions links spectral embedding and kernel PCA. *Neural Comput.*, *16*(10), 2197–2219.

Boulesteix, A.-L., & Strimmer, K. (2007). Partial least squares: a versatile tool for the analysis of high-dimensional genomic data. *Brief. Bioinform.*, *8*(1), 32–44.

Chang, W.-C. (1983). On using principal components before separating a mixture of two multivariate normal distributions. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *32*(3), 267–275.

Chen, N., Zi, X., & Zou, C. (2016). A distribution-free multivariate control chart. *Technometrics*, *58*(4), 448–459.

Choi, S. W., Lee, C. K., Lee, J. M., Park, J. H., & Lee, I. B. (2005). Fault detection and identification of nonlinear processes based on kernel PCA. *Chemometr. Intell. Lab.*, *75*(1), 55–67.

Choi, S. W., & Lee, I. B. (2004). Nonlinear dynamic process monitoring based on dynamic kernel PCA. *Chem. Eng. Sci.*, *59*(24), 5897–5908.

Chouaib, C., Mohamed-Faouzi, H., & Messaoud, D. (2013). Adaptive kernel principal component analysis for nonlinear dynamic process monitoring. In *Control Conference (ASCC), 2013 9th Asian*, (pp. 1–6).

Cook, R. D., & Forzani, L. (2008). Covariance reducing models: An alternative to spectral modelling of covariance matrices. *Biometrika*, *95*(4), 799–812.

Cook, R. D., & Weisberg, S. (1991). Sliced inverse regression for dimension reduction: comment. *J. Am. Stat. Assoc.*, *86*(414), 328–332.

Cook, R. D., & Yin, X. (2001). Theory & methods: Special Invited Paper: Dimension reduction and visualization in discriminant analysis. *Aust. N.Z. J. Stat.*, *43*(2), 147–199.

Dong, D., & McAvoy, T. J. (1996). Batch tracking via nonlinear principal component analysis. *AIChE J.*, *42*(8), 2199–2208.

Eckart, C., & Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, *1*(3), 211–218.

EPA (2000). Wastewater technology fact sheet: In-plant pump stations. Tech. Rep. EPA 832-F-00-069, U.S. Environmental Protection Agency.

Fan, J., Ke, Z. T., Liu, H., & Xia, L. (2015). QUADRO: A supervised dimension reduction method via Rayleigh quotient optimization. *Ann. Stat.*, *43*(4), 1498–1534.

Fischer, K., & Thiele, C. (1979). On a distributionfree method in discriminant analysis. *Statistics*, *10*(2), 281–289.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Ann. Eugenic.*, *7*(2), 179–188.

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition*. Boston: Academic Press, 2nd edition ed.

Garcia-Munoz, S., Kourti, T., & MacGregor, J. F. (2004). Model predictive monitoring for batch processes. *Ind. Eng. Chem. Res.*, *43*(18), 5929–5941.

Ge, Z., & Song, Z. (2012). *Multivariate Statistical Process Control: Process Monitoring Methods and Applications*. London: Springer-Verlag.

Geary, R. C. (1935). The ratio of the mean deviation to the standard deviation as a test of normality. *Biometrika*, *27*(3-4), 310–332.

Gertler, J., Li, W., Huang, Y., & McAvoy, T. J. (1999). Isolation enhanced principal component analysis. *AIChE J.*, *45*(2), 323.

Gikas, P., & Tchobanoglous, G. (2009). The role of satellite and decentralized strategies in water resources management. *J. Environ. Manage.*, *90*(1), 144–152.

Golub, G. H., & Van Loan, C. F. (1996). *Matrix Computations*. JHU Press, 2 ed.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D., & Lander, E. S. (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, *286*(5439), 531–537.

Gravier, E., Pierron, G., Vincent-Salomon, A., Gruel, N., Raynal, V., Savignoni, A., De Rycke, Y., Pierga, J.-Y., Lucchesi, C., Reyal, F., Fourquet, A., Roman-Roman, S., Radvanyi, F., Sastre-Garau, X., Asselain, B., & Delattre, O. (2010). A prognostic DNA signature for T1t2 node-negative breast cancer patients. *Genes Chromosom. Cancer*, *49*(12), 1125–1134.

Haff, L. R. (1979). Estimation of the inverse covariance matrix: random mixtures of the inverse wishart matrix and the identity. *Ann. Stat.*, *7*(6), 1264–1276.

Hartwig, R. E. (1986). The reverse order law revisited. *Linear. Algebra. Appl.*, *76*, 241–246.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2001). *The elements of statistical learning: data mining, inference, and prediction: with 200 full-color illustrations*. Springer series in statistics. New York: Springer.

Hennig, C. (2004). Asymmetric linear dimension reduction for classification. *J. Comput. Graph. Stat.*, *13*(4), 930–945.

Huang, S., Kong, Z., & Huang, W. (2014). High-dimensional process monitoring and change point detection using embedding distributions in reproducing kernel Hilbert space. *IIE Transactions*, *46*(10), 999–1016.

Jearkpaporn, D., Borror, C. M., Runger, G. C., & Montgomery, D. C. (2007). Process monitoring for mean shifts for multiple stage processes. *Int. J. Prod. Res.*, *45*(23), 5547–5570.

Johnson, R. A., & Wichern, D. W. (2002). *Applied Multivariate Statistical Analysis*. Upper Saddle River, NJ: Prentice Hall, 5th ed.

Kano, M., Nagao, K., Hasebe, S., Hashimoto, I., Ohno, H., Strauss, R., & Bakshi, B. (2000). Comparison of statistical process monitoring methods: Application to the Eastman challenge problem. *Comput. Chem. Eng.*, *24*(2), 175–181.

Kazor, K., Holloway, R. W., Cath, T. Y., & Hering, A. S. (2016). Comparison of linear and nonlinear dimension reduction techniques for automated process monitoring of a decentralized wastewater treatment facility. *Stoch. Environ. Res. Risk Assess.*, *30*(5), 1527–1544.

Khan, J., Wei, J. S., Ringner, M., Saal, L. H., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C. R., Peterson, C., & Meltzer, P. S. (2001). Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nat Med*, *7*(6), 673–679.

Kresta, J. V., MacGregor, J. F., & Marlin, T. E. (1991). Multivariate statistical monitoring of process operating performance. *Can. J. Chem. Eng.*, *69*(1), 35–47.

Krzanowski, W. J. (1992). Ranking principal components to reflect group structure. *J. Chemometrics*, *6*(2), 97–102.

Kumar, N., Andreou, A. G., & Andreou, N. K. A. G. (1996). A generalization of linear discriminant analysis in maximum likelihood framework.

Lee, H. C., & Apley, D. W. (2011). Improved design of robust exponentially weighted moving average control charts for autocorrelated processes. *Qual. Reliab. Engng. Int.*, *27*(3), 337–352.

Lee, J. M., Yoo, C. K., & Lee, I. B. (2004). Statistical process monitoring with independent component analysis. *J. Process Contr.*, *14*(5), 467–485.

Leverenz, H. L., & Asano, T. (2011). 4.03 - Wastewater Reclamation and Reuse System A2. In P. Wilderer (Ed.) *Treatise on Water Science*, (pp. 63–71). Oxford: Elsevier.

Li, K.-C. (1991). Sliced inverse regression for dimension reduction. *J. Am. Stat. Assoc.*, *86*(414), 316–327.

Li, Y., & Tsung, F. (2012). False discovery rate-adjusted charting schemes for multistage process monitoring and fault identification. *Technometrics*, *51*(2), 186–205.

Liu, H., & Motoda, H. (1998). *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Springer Science & Business Media.

Loog, M., & Duin, R. P. W. (2004). Linear dimensionality reduction via a heteroscedastic extension of LDA: the Chernoff criterion. *IEEE Trans. Pattern. Anal. Mach. Intell.*, *26*(6), 732–739.

Mahanta, M. S., Aghaei, A. S., Plataniotis, K. N., & Pasupathy, S. (2012). Heteroscedastic linear feature extraction based on sufficiency conditions. *Pattern Recogn.*, *45*(2), 821–830.

Miao, A., Song, Z., Ge, Z., Zhou, L., & Wen, Q. (2013). Nonlinear fault detection based on locally linear embedding. *J. Control Theory Appl.*, *11*(4), 615–622.

Nguyen, D. V., & Rocke, D. M. (2002). Multi-class cancer classification via partial least squares with gene expression profiles. *Bioinformatics*, *18*(9), 1216–1226.

Ounpraseuth, S. T., Young, P. D., Van Zyl, J. S., Nelson, T. W., & Young, D. M. (2015). Linear dimension reduction for multiple heteroscedastic multivariate normal populations. *OJS*, *05*(04), 311.

Pavlidis, P. (2003). Using ANOVA for gene selection from microarray studies of the nervous system. *Methods*, *31*(4), 282–289.

Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, *2*(11), 559–572.

Peck, R., Jennings, L. W., & Young, D. M. (1988). A comparison of several biased estimators for improving the expected error rate of the sample quadratic discriminant function. *J. Stat. Comput. Sim.*, *29*(2), 143–156.

Penrose, R. (1955). A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, *51*(3), 406–413.

Peters, B. C., Redner, R., & Decell, H. P. (1978). Characterizations of Linear Sufficient Statistics. *Sankhya Ser. A*, *40*(3), 303–309.

Qiu, P. (2013). *Introduction to Statistical Process Control*. Boca Raton, FL: CRC Press.

Qiu, P., & Li, Z. (2012). On nonparametric statistical process control of univariate processes. *Technometrics*, *43*(4), 390–405.

Rato, T. J., & Reis, M. S. (2013). Defining the structure of DPCA models and its impact on process monitoring and prediction activities. *Chemometr. Intell. Lab.*, *125*, 74–86.

Raychaudhuri, S., Stuart, J. M., & Altman, R. B. (2000). Principal components analysis to summarize microarray experiments: Application to sporulation time series. *Pac Symp Biocomput*, (pp. 455–466).

Sanchez-Fernandez, A., Fuente, M. J., & Sainz-Palmero, G. I. (2015). Fault detection in wastewater treatment plants using distributed PCA methods. In *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, (pp. 1–7).

Satagopan, J. M., & Panageas, K. S. (2003). A statistical perspective on gene expression data analysis. *Statist. Med.*, *22*(3), 481–499.

Scholkopf, B., Smola, A., & Muller, K. R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, *10*(5), 1299–1319.

Schott, J. R. (1994). Determining the dimensionality in sliced inverse regression. *J. Am. Stat. Assoc.*, *89*(425), 141–148.

Sheather, S. (2009). *A Modern Approach to Regression with R*. New York, NY: Springer.

Sigillito, V. G., Wing, S. P., Hutton, L. V., & Baker, K. B. (1989). Classification of radar returns from the ionosphere using neural networks. *J. Hopkins Apl. Tech. D.*, *vol. 10*, 262–266. In.

Song, J. J., Ren, Y., & Yan, F. (2009). Classification for high-throughput data with an optimal subset of principal components. *Computational Biology and Chemistry*, *33*(5), 408–413.

Sorlie, T., Perou, C. M., Tibshirani, R., Aas, T., Geisler, S., Johnsen, H., Hastie, T., Eisen, M. B., Rijn, M. v. d., Jeffrey, S. S., Thorsen, T., Quist, H., Matese, J. C., Brown, P. O., Botstein, D., Lonning, P. E., & Borresen-Dale, A.-L. (2001). Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications. *PNAS*, *98*(19), 10869–10874.

Stavropoulos, P., Chantzis, D., Doukas, C., Papacharalampopoulos, A., & Chryssolouris, G. (2013). Monitoring and control of manufacturing processes: A review. *Procedia CIRP*, *8*, 421–425.

Sun, L., Hui, A.-M., Su, Q., Vortmeyer, A., Kotliarov, Y., Pastorino, S., Passaniti, A., Menon, J., Walling, J., Bailey, R., Rosenblum, M., Mikkelsen, T., & Fine, H. A. (2006). Neuronal and glioma-derived stem cell factor induces angiogenesis within the brain. *Cancer Cell*, *9*(4), 287–300.

Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*(5500), 2319–2323.

Torgerson, W. S. (1958). *Theory and Methods of Scaling*, vol. xiii. Oxford, England: Wiley.

van't Veer, L. J., Dai, H., van de Vijver, M. J., He, Y. D., Hart, A. A. M., Mao, M., Peterse, H. L., van der Kooy, K., Marton, M. J., Witteveen, A. T., Schreiber, G. J., Kerkhoven, R. M., Roberts, C., Linsley, P. S., Bernards, R., & Friend, S. H. (2002). Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, *415*(6871), 530–536.

Velilla, S. (2008). A method for dimension reduction in quadratic classification problems. *J. Comput. Graph. Stat.*, *17*(3), 572–589.

Velilla, S., & Hernandez, A. (2005). On the consistency properties of linear and quadratic discriminant analyses. *J. Multivariate Anal.*, *96*(2), 219–236.

Venables, W. N., & Ripley, B. D. (1999). *Modern Applied Statistics with S-Plus (Statistics and Computing)*. New York: Springer, 3rd ed.

Vuono, D., Henkel, J., Benecke, J., Cath, T. Y., Reid, T., Johnson, L., & Drewes, J. E. (2013). Flexible hybrid membrane treatment systems for tailored nutrient management: A new paradigm in urban wastewater treatment. *J. Membrane Sci.*, *446*, 34–41.

Wang, A., & Gehan, E. A. (2005). Gene selection for microarray data analysis using principal component analysis. *Statist. Med.*, *24*(13), 2069–2087.

Wang, H., Yin, J., Pei, J., Yu, P. S., & Yu, J. X. (2006). Suppressing Model Overfitting in Mining Concept-drifting Data Streams. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, (pp. 736–741). New York, NY, USA: ACM.

Wang, J., & He, Q. P. (2010). Multivariate statistical process monitoring based on statistics pattern analysis. *Ind. Eng. Chem. Res.*, *49*(17), 7858–7869.

Weinberger, K. Q., & Saul, L. K. (2006). Unsupervised learning of image manifolds by semidefinite programming. *Int. J. Comput. Vision*, *70*(1), 77–90.

Westfall, P. H., & Young, S. S. (1993). *Resampling-Based Multiple Testing: Examples and Methods for p-Value Adjustment*. New York: John Wiley & Sons.

Wise, B. M., Veltkamp, D. J., Davis, B., Ricker, N. L., & Kowalski, B. R. (1988). Principal components analysis for monitoring the West Valley Liquid-Fed Ceramic Melter. In *Management of Radioactive Wastes, and Non-Radioactive Wastes from Nuclear Facilities*, vol. 20. Tucson, AZ.

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometr. Intell. Lab.*, *2*(1), 37–52.

Young, D. M., Marco, V. R., & Odell, P. L. (1987a). Quadratic discrimination: Some results on optimal low-dimensional representation. *J. Stat. Plan. Infer.*, *17*, 307–319.

Young, D. M., Turner, D. W., & Marco, V. R. (1987b). Some results on error rates for quadratic discrimination with known population parameters. *Biom. J.*, *29*(6), 721–730.

Zhang, W., Liu, X. Y., Qi, R. L., & Jiang, Y. (2013). Improved locally linear embedding based method for nonlinear system fault detection. *Chi. Acad. Sci. Int. J. Adv. Comp. Tech.*, *5*(1).

Zheng, Y. P., & Zhang, L. P. (2013). Fault Diagnosis of Wet Flue Gas Desulphurization System Based on KPCA. In E. Qi, J. Shen, & R. Dou (Eds.) *The 19th International Conference on Industrial Engineering and Engineering Management*, (pp. 279–288). Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-37270-4_27.

Zou, C., Jiang, W., & Tsung, F. (2011). A LASSO-based diagnostic framework for multivariate statistical process control. *Technometrics*, *53*(3), 297–309.