Mahalanobis kernel-based support vector data description for detection of large shifts in mean vector

2015

Vu Nguyen
University of Central Florida

Find similar works at: https://stars.library.ucf.edu/etd

University of Central Florida Libraries http://library.ucf.edu

Part of the Statistics and Probability Commons

STARS Citation

https://stars.library.ucf.edu/etd/1160

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of STARS. For more information, please contact lee.dotson@ucf.edu.
MAHALANOBIS KERNEL-BASED SUPPORT VECTOR DATA DESCRIPTION FOR DETECTION OF LARGE SHIFTS IN MEAN VECTOR

by

VU NGUYEN
B.S. Truman State University, 2011

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in the Department of Statistics in the College of Sciences at the University of Central Florida Orlando, Florida

Spring Term
2015

Major Professor: Edgard Messan-Tchao Maboudou
ABSTRACT

Statistical process control (SPC) applies the science of statistics to various process control in order to provide higher-quality products and better services. The K chart is one among the many important tools that SPC offers. Creation of the K chart is based on Support Vector Data Description (SVDD), a popular data classifier method inspired by Support Vector Machine (SVM). As any methods associated with SVM, SVDD benefits from a wide variety of choices of kernel, which determines the effectiveness of the whole model. Among the most popular choices is the Euclidean distance-based Gaussian kernel, which enables SVDD to obtain a flexible data description, thus enhances its overall predictive capability. This thesis explores an even more robust approach by incorporating the Mahalanobis distance-based kernel (hereinafter referred to as Mahalanobis kernel) to SVDD and compare it with SVDD using the traditional Gaussian kernel. Method's sensitivity is benchmarked by Average Run Lengths obtained from multiple Monte Carlo simulations. Data of such simulations are generated from multivariate normal, multivariate Student's (t), and multivariate gamma populations using R, a popular software environment for statistical computing. One case study is also discussed using a real data set received from Halberg Chronobiology Center. Compared to Gaussian kernel, Mahalanobis kernel makes SVDD and thus the K chart significantly more sensitive to large shifts in mean vector, and also in covariance matrix.
To Fernando
ACKNOWLEDGMENTS

As this thesis has taken a full year to finally come to completion, it would have not been possible if not for the generous support of several individuals. First I would like to thank Dr. Edgard Maboudou for giving me the opportunity to collaborate with him. His insightful guidance and unparallel patience with me have helped me overcome many challenges along the way. A special thank goes to the other two members of my advisory committee, Dr. David Nickerson and Dr. James Schott, for providing some much needed teaching. I would like to also thank Ms. Aida Encarnacion and Ms. Elena Sequera of the department of statistics for their opportune and invaluable support. Last but not least, I want to thank my brothers and sisters in the UCF men's rowing team for standing by my side throughout the year, giving me numerous encouragements that never cease to inspire me to move forward. I appreciate what you all have done for me so much and hope that I may be able to return the favor someday.
TABLE OF CONTENTS

LIST OF FIGURES ........................................................................................................................................... ix

LIST OF TABLES ................................................................................................................................................ xi

LIST OF ABBREVIATIONS .................................................................................................................................... xii

CHAPTER ONE: INTRODUCTION ...................................................................................................................... 1

1.1. Statistical Process Control and Control Chart .......................................................................................... 1

1.2. Multivariate Control Chart ....................................................................................................................... 3

1.3. Rational Subgroup ..................................................................................................................................... 5

1.4. Motivation and Contribution .................................................................................................................... 7

CHAPTER TWO: SUPPORT VECTOR METHODS ........................................................................................ 8

2.1. Support Vector Machine .......................................................................................................................... 8

2.2. Support Vector Data Description ............................................................................................................ 13

2.3. K Chart ..................................................................................................................................................... 16

2.4. Proposal to Improve the K Chart ............................................................................................................. 17

CHAPTER THREE: METHODOLOGY ........................................................................................................... 19
3.1. Simulation Data Generation ................................................................. 21
  3.1.1. Multivariate Normal ................................................................. 22
  3.1.2. Multivariate Student's  ......................................................... 22
  3.1.3. Multivariate Gamma ............................................................... 23

3.2. Grid Search ......................................................................................... 23

3.3. Monte Carlo Simulation ................................................................. 29
  3.3.1. Adjust Control Limit ............................................................... 31
  3.3.2. Simulation on Out-of-Control Data ........................................ 33

CHAPTER FOUR: RESULTS ................................................................. 34

  4.1. Multivariate Normal ................................................................. 34

  4.2. Multivariate Student's (t) ........................................................... 39
    4.2.1 Three Degrees of Freedom ................................................... 39
    4.2.2. Five Degrees of Freedom .................................................... 42

  4.3. Multivariate Gamma ................................................................. 46

CHAPTER FIVE: CASE STUDY ............................................................. 51
LIST OF FIGURES

Figure 1: Example of a classical control chart................................................................................ 2

Figure 2: Example of a Hotelling Control Chart ............................................................................ 4

Figure 3: Example of a binary classification problem with two classes depicted by circles and squares; SVM solves this by constructing the separating hyperplane that maximizes the margin between two classes.................................................................................................................. 10

Figure 4: Example of a curved hyperplane constructed by SVM using Gaussian kernel.............. 13

Figure 5: Example of a K chart..................................................................................................... 17

Figure 6: Example of how different values of $C$ and $\sigma$ can affect data description -- a hypersphere projected onto 2-dimensional space here as a yellow perimeter................................................................. 25

Figure 7: Visualization of a 9-by-9 grid search for optimum $C$ and $\sigma$ values ($k = 8$).......... 26

Figure 8: Example the procedure to determine new ranges for $C$ and $\sigma$............................... 28

Figure 9: Average run lengths with 99% confidence intervals on multivariate normal observations generated with shifts in mean vector ........................................................................................................ 37

Figure 10: Average run lengths with 99% confidence intervals on multivariate normal observations generated with shifts in covariance matrix ............................................................. 38
Figure 11: Average run lengths with 99% confidence intervals on multivariate Student's
observations with 3 degrees of freedom, generated with shifts in mean vector ....................... 40

Figure 12: Average run lengths with 99% confidence intervals on multivariate Students
observations with 3 degrees of freedom, generated with shifts in covariance matrix ............... 41

Figure 13: Average run lengths with 99% confidence intervals on multivariate Student's
observations with 5 degrees of freedom, generated with shifts in mean vector ....................... 43

Figure 14: Average run lengths with 99% confidence intervals on multivariate Students
observations with 5 degrees of freedom, generated with shifts in covariance matrix ............... 44

Figure 15: Average run lengths with 99% confidence intervals on multivariate gamma
observations generated with shifts in mean vector ..................................................................... 49

Figure 16: Average run lengths with 99% confidence intervals on multivariate gamma
observations generated with shifts in covariance matrix ............................................................ 50

Figure 17: Monitoring process on the second half of Halberg data by a K chart constructed with
SVDD using Gaussian kernel ....................................................................................................... 54

Figure 18: Monitoring process on the second half of Halberg data by a K chart constructed with
SVDD using Mahalanobis kernel ............................................................................................... 55
LIST OF TABLES

Table 1: Averages and standard errors of run lengths from three different methods to detect out-of-control observations for multivariate normal variates generated with shifts in mean vector $\mu$ and covariance matrix $\Sigma$ ................................................................................................................................. 36

Table 2: Averages and standard errors of run lengths from both methods to detect out-of-control observations for multivariate Student's variates with 3 degrees of freedom generated with shifts in mean vector $\mu$ and covariance matrix $\Sigma$ ................................................................................................................................. 39

Table 3: Averages and standard errors of run lengths from both methods to detect out-of-control observations for multivariate Student's variates with 5 degrees of freedom generated with shifts in mean vector $\mu$ and covariance matrix $\Sigma$ ................................................................................................................................. 42

Table 4: Averages and standard errors of run lengths from both methods to detect out-of-control observations for multivariate gamma variates generated with shifts in mean vector $\mu$ and covariance matrix $\Sigma$ ................................................................................................................................. 46
**LIST OF ABBREVIATIONS**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARL</td>
<td>Average Run Length</td>
</tr>
<tr>
<td>IC</td>
<td>In-Control</td>
</tr>
<tr>
<td>LASSO</td>
<td>Least Absolute Shrinkage and Selection Operator</td>
</tr>
<tr>
<td>LCL</td>
<td>Lower Control Limit</td>
</tr>
<tr>
<td>OC</td>
<td>Out-of-Control</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis</td>
</tr>
<tr>
<td>SPC</td>
<td>Statistical Process Control</td>
</tr>
<tr>
<td>SVDD</td>
<td>Support Vector Data Description</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>UCL</td>
<td>Upper Control Limit</td>
</tr>
</tbody>
</table>
CHAPTER ONE: INTRODUCTION

1.1. Statistical Process Control and Control Chart

Process Control has a long history in the development of human society. It is safe to say that process control began as early as when mankind associated quality with manufacturing process. Except in a monopolistic environment, consumers continually compare one product to another, motivating manufacturers to improve their offerings, by either improving the quality, or lowering the price. Quality improvement lies not only in the final product itself, but also in the process of making it. Statistical process control (SPC) is a methodology that employs the science of statistics in process control, providing the latter with powerful tools. Most major achievements in SPC occurred during the twentieth century; many of them are still prevalent in today’s industrial setting, only further strengthen by recent breakthroughs in statistical learning and data mining techniques. Invented by Walter Shewhart in the 1920s while he was working at Bell Telephone Laboratories, control chart (sometimes called Shewhart chart) is an important SPC tool whose concept has bore many different useful variations. A control chart is a graphical presentation that helps quality control engineers, designers, statisticians, or anyone of concern, visually inspect the process (Figure 1).
A classical control chart typically displays control process for one particular quality. It consists of a center line (sometimes referred to as *average*) running across the center of the chart. There is also an upper limit line (or *upper control limit* -- UCL) above the center line and a lower limit line (or *lower control limit* -- LCL) below it. Both limits are determined by various statistical calculations, which often (but not always) rely on some distributional assumptions. A process is considered in-control (IC) when it behaves as it is designed (i.e. producing products with specifications conforming to their standards). It is expected that when the process being monitored is in-control, observations will be found between the LCL and UCL, where their differences are attributed to random, statistical deviations rather than irregularities. However, when one observation is found above the UCL or below the LCL, it is considered an outlier or anomaly. In such situation, the process is deemed to have gone out-of-control (OC), and requires attention to identify the cause(s) and determine appropriate course of remedial actions.

![A Basic Control Chart](image)

Figure 1: Example of a classical control chart
1.2. Multivariate Control Chart

Ever since its conception almost a century ago, control chart has been studied and further developed from its classical form to increase effectiveness and better suit various purposes in different industrial settings. There are two different categories of control charts: univariate and multivariate control charts. Univariate control charts, which include the classical $\bar{X}$ chart design, only monitor one quality at a time. In contrast, multivariate control charts monitor 2 or more qualities at the same time. The major advantages of univariate control charts are that: they are simpler thus quicker to learn and easier to use, and the amount of computation involved is usually manageable for manual calculations. In practice, however, most data are naturally multivariate, and there usually exists some form of correlations among the variables, for which it is not possible to account if monitored individually (i.e. from using a collection of univariate charts on individual qualities). Because of that, multivariate methods are more powerful and typically more useful in practice. The drawbacks of multivariate control charts are that: computations are complex and laborious, and some knowledge of matrix algebra is required. For those reasons, acceptance and development of multivariate control charts in industrial settings were slow for a few decades. But nowadays with the advent of modern computers, the first drawback has largely become negligible. Consequently, multivariate control charts have gained much attention for the past decade, with the most popular one being Hotelling control chart (Hotelling, 1947), which is based on Hotelling's $T^2$ statistic:

$$T^2 = (x - \bar{x})' S^{-1} (x - \bar{x})$$  \hspace{1cm} (1.1)
where $x$ is a $p$-variable observation with sample mean vector $\bar{x}$ and sample covariance matrix $S$.

The Hotelling’s $T^2$ distance, proposed by Harold Hotelling in 1947, is a measure that accounts for the covariance structure ($S$) of a multivariate normal distribution. It is generally considered the multivariate counterpart of the Student's-t statistic. And Hotelling control chart itself is considered a direct multivariate extension of the univariate $\bar{X}$ chart.

Figure 2: Example of a Hotelling Control Chart

Despite being the most popular one, Hotelling control chart is not the only multivariate control chart in existence. (Hawkins and Maboudou, 2008) use Multivariate Exponentially Weighted Moving Covariance Matrix (MEWMC) chart to monitor changes in covariance matrix. (Wang and Jiang, 2009) and (Zou and Qiu, 2009) construct control charts for monitoring multivariate mean vector using the Least Absolute Shrinkage and Selection Operator (LASSO) type penalty. (Maboudou and Diawara, 2013) also propose a LASSO chart for monitoring
covariance matrix. Whilst praised for their powerful properties, those charts share the same drawback with the Hotelling chart: the assumption that data are multivariate normal. In practice, the distribution of data is usually unknown. Consequently, even though there exist methods to assert multivariate normality, this assumption imposes some limitations on practical usability of the charts (i.e. they may not be useable when data are not multivariate normal.) This holds true for many other multivariate control charts, such as Principal Component Analysis (PCA), and Partial Least Squares (PLS) charts. Besides attempts to eliminate the normality assumption requirement on some of the methods, such as for PCA chart (Phaladiganon et al., 2012), Sun and Tsung have introduced a novel approach, called kernel-distance-based control chart (or K chart), which does not rely on any distributional assumptions (Sun and Tsung, 2003). The K chart is discussed in details in chapter two.

1.3. Rational Subgroup

Shewhart advocated segregating data into rational subgroups so that variation within subgroups is minimized and variation among subgroups is maximized; this makes the chart more sensitive to large shifts (Shewhart, 1931). Data set is divided into subgroups of equal size. Let \((x_1, x_2, x_3, \ldots x_m)^T\) be a data set, where \(x_i\) is a \(p\)-dimension vector that is the \(i^{th}\) observation among \(m\) individual observations. Supposed \(k\) subgroups of size \(n\) are desired, that means every \(n\) observations form a group \(G_i\) where \(i = 1, 2, 3, \ldots k\). The mean vector and sample covariance matrix for each subgroup are then calculated as followed:
\[
\bar{x}_i = \frac{1}{n} \sum_{x_j \in G_i} x_j
\]  
\quad (1.2)

\[
S_i = \frac{1}{n-1} \sum_{x_j \in G_i} (x_j - \bar{x}_i)(x_j - \bar{x}_i)^T.
\]  
\quad (1.3)

Rather than conducting process control on single observations \(x_i\), the subgroup means \(\bar{x}_i\) as calculated in (1.2) are treated as observations instead. The sample mean vector is calculated by taking the average of all \(k\) subgroup means (which in turn should not be different from the mean vector of all individual observations):

\[
\bar{x} = \frac{1}{k} \sum_k \bar{x}_i.
\]  
\quad (1.4)

And the sample covariance matrix is estimated by calculating the average of all \(k\) subgroup covariance matrices:

\[
\bar{S} = \frac{1}{k} \sum_k S_i.
\]  
\quad (1.5)

Dividing individual observations into rational subgroups obviously involves extra work during data preparation, and results in lower data resolution. For example, if raw data which are daily measurements are compressed into weekly averages, and an out-of-control observation is detected, then we will know some anomalies have occurred in that certain week, but cannot immediately tell which day. However this should not be a big concern for three reasons. First, it has been established that using subgroups as observations increases the control process' sensitivity with large shifts (Shewhart, 1930), which is the main target. Second, in practice, majority of the observations should be in-control most of the times, so regular monitoring at high
resolution typically is unnecessary. And last but not least, rational subgroups not only represent a trade-off between detecting power and data resolution, but also may be used as blocking for data sources or any potential treatment effects that are worth investigating. For example, observations from a sensor or data collector can be grouped together to detect any potential anomalies originated from the sensors or collectors themselves.

1.4. Motivation and Contribution

It wouldn't be an overstatement to say that statistical process control has a strong impact on our modern society. With a wide array of tools to offer, it helps ensure the high quality of food, products, and even services we receive every day. Any development in statistical process control can potentially result in a huge leap in quality of products and services, thus bring about higher standards of living. This thesis attempts to improve the K chart, a well-known and powerful multivariate control chart, by incorporating some novel approach to make it even more robust in detecting process-wide changes than most currently existing tools. The rest of the thesis is organized as followed: Chapter two discusses several support vector methods, including support vector machine, support vector data description, and the latter's usage in constructing the K chart; Chapter three talks about the methodology that is carried throughout this research; Chapter four discusses results found by simulations; Chapter five presents a case study using real data; And finally a quick summary and conclusion are given in chapter six.
CHAPTER TWO: SUPPORT VECTOR METHODS

2.1. Support Vector Machine

The support vector methods were first proposed for binary classification problems (Vapnik, 1995 & 1998), then recently extended to many other machine learning techniques. The most renowned support vector method is support vector machine, which is a supervised machine learning model used for classification and regression analysis. In machine learning, a learning model is considered supervised when labels for training observations are available. A binary classification problem is one in which observations need to be identified into either one of two classes (hence binary). Let \( x_i \) with \( i = 1, 2, \ldots, m \), and \( x_i \in \mathbb{R}^p \) with corresponding labels \( y_i \in \{-1, 1\} \). Here \( m \) is the number of training observations, and \( p \) is the number of variables of each observation. Support vector machine (SVM) seeks to construct a decision boundary (a hyper-plane in multidimensional space) which separates the two classes such that the distance in the direction perpendicular to it between the nearest points of each class (also known as the margin between two classes) is maximized. Achieving such is equivalent to solving the following quadratic programming problem:

\[
\text{Min } \Phi(w) = \frac{1}{2} (w \cdot w) \tag{2.1}
\]

under the constraints

\[
y_i [(w \cdot x_i) + b] \geq 1 , \text{ for } i = 1, 2, \ldots, m. \tag{2.2}
\]
There exists only one hyperplane that maximizes the margin, so the solution \((w_0, b_0)\) to the optimization problem above is unique, and can be obtained by:

\[
w_0 = \sum_{i=1}^{m} \alpha_{0i} y_i x_i \tag{2.3}
\]

where \(\alpha_{0i}'s\) are non-negative real coefficients and the solutions of the following quadratic programming problem:

\[
\text{Max } W(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \tag{2.4}
\]

under the constraints

\[
\alpha_i \geq 0, \quad i = 1,2,3,\ldots,m \tag{2.5}
\]

\[
\sum_i \alpha_i y_i = 0 \tag{2.6}
\]

which can be solved by computational softwares. Here, all observations that correspond to non-zero \(\alpha\) values (i.e. all \(x_i\) such that \(\alpha_i > 0\)) are call the support vectors. \(b_0\) can be obtained using any two support vectors from both classes:

\[
b_0 = -\frac{1}{2} \left[ (w_0 \cdot x^*(1)) + (w_0 \cdot x^*(-1)) \right] \tag{2.7}
\]

where \(x^*(t)\) denotes any support vector belonging to the class with label \(y_i = t\). The classification function for a new observation \(z\) is:

\[
f(z) = \sum_{i=1}^{m} \alpha_{0i} y_i(z \cdot x_i) + b_0. \tag{2.8}
\]
If $f(z)$ is positive, $z$ is classified into the class with label $y_i = 1$. Otherwise it belongs to the class with label $y_i = -1$.

Figure 3: Example of a binary classification problem with two classes depicted by circles and squares; SVM solves this by constructing the separating hyperplane that maximizes the margin between two classes.

(Source: Sun and Tsung, 2003)

Figure 3 shows a hyperplane constructed by the classical case of SVM, which uses what is known as a \textit{linear} kernel -- defined as a simple inner product of two vector observations:

$$K(x_i, x_j) = (x_i \cdot x_j).$$  

(2.9)
A linear kernel results in a flat boundary which is represented by a straight line in two-dimensional projection of the hyperspace. In practice, it is usually not possible to completely separate two classes using a flat hyperplane -- this is known as *nonlinear* problems. For example: a few observations of a class may be situated among a cluster of the other's. In other words, some degree of flexibility is required of the hyperplane in order to better separate two classes. SVM achieves that by replacing the inner product in (2.4) with a kernel function, such as:

$$K(x_i, x_j) = (x_i \cdot x_j)^d$$  \hspace{1cm} (2.10)

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i-x_j\|^2}{\sigma^2}\right).$$ \hspace{1cm} (2.11)

(2.10) is called *polynomial* kernel function with the exponent constant $d \in \mathbb{N}^+$; and (2.11) is known as *Gaussian* kernel function (also called *radial basis function*) with the scale variable $\sigma^2$. These are not the only choices for kernels. In fact, any function that meets the Mercer's condition (Schölkopf et al., 1999) can be used as a kernel function (Vapnik, 1995). The Mercer's theorem is stated below.

**Mercer's theorem:** A symmetric function $K(x, y)$ can be expressed as an inner product

$$K(x, y) = \langle \phi(x), \phi(y) \rangle$$ \hspace{1cm} (2.12)

for some $\phi$ if and only if $K(x, y)$ is positive definite, i.e.:

$$\int K(x, y) \ g(x) \ g(y) \ dx \ dy \geq 0 \ \forall \ g.$$ \hspace{1cm} (2.13)
In other words, any function that satisfies (2.13) can be used as a kernel function. Regardless, polynomial and Gaussian kernels are popular choices for SVM. Both are capable of increasing the flexibility of the boundary hyperplane, but with polynomial kernels, SVM's analytical capability is limited by the exponent constant \(d\); while Gaussian kernels provide the model with a potentially infinite degree of complexity that can grow with the data (Chang et al., 2010).

While support vector machine is a powerful model and widely-used, it still struggles with one challenge just like any popular data classifying algorithms such as logistic regression and artificial neural network, which is the sufficiently representative availability of data from all classes. In particular, a sizable number of out-of-control observations are usually required to make accurate predictions. It may sound like a simple problem but in reality is not. In-control data are usually overwhelmingly available thus relatively easy and cheap to get; on the other hands, out-of-control data more often than not are hard and expensive to obtain. For example, it is typically not difficult to collect a great amount of in-control data for an functional machine. Meanwhile, in order to obtain a sufficiently representative number of out-of-control observations from the same machine, it would have to be broken in every possible combination of ways -- which is not realistically viable. The support vector data description method (Tax et al., 1999) offers a solution to that problem.
Figure 4: Example of a curved hyperplane constructed by SVM using Gaussian kernel

(Source: Sung and Tsung, 2013)

2.2. Support Vector Data Description

Inspired by support vector machine, support vector data description (SVDD) classifies in-control from out-of-control observations by constructing a description which takes form of a hypersphere to enclose in-control data, and any observations that fall outside such description's boundary are declared out-of-control. While out-of-control observations can help tighten the description (Tax and Duin, 2004), SVDD algorithm targets only in-control observations hence
does not require out-of-control data for training. Consequently, SVDD is a great candidate for classification problems where obtaining out-of-control data is challenging or not cost-effective.

With the objective of creating a boundary within the hyperspace that contains training data, SVDD seeks to minimize the volume of this hypersphere (or description) while maximizing the number of training objects it can enclose. (Chang et al., 2007) used the Karush-Kuhn-Tucker (KKT) optimality conditions and Slater's condition for strong duality to obtain an optimal solution to the SVDD problem. Let \( a \) be the center of the hypersphere, and \( R \) be its radius (i.e. distance between \( a \) and the boundary, \( R \geq 0 \)); Let \( x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^T \) for \( i = 1, 2, \ldots, m \) be a sequence of \( p \)-variable training observations. The problem becomes:

\[
\text{Min } (R + C \sum \xi_i) \tag{2.14}
\]

with the constraint:

\[
\|x_i - a\|^2 \leq R + \xi_i \tag{2.15}
\]

where \( \xi_i \geq 0 \) is a slack variable that allows \( x_i \) to be out of the hypersphere. \( C \) is a trade-off variable that regulates the hypersphere's volume and misclassification errors. (2.14) can be solved by the following Lagrangian:

\[
L(R, a, \alpha_i, \gamma_i, \xi_i) = R + C \sum \xi_i - \sum \alpha_i \left( R + \xi_i - \|x_i - a\|^2 \right) - \sum \gamma_i \xi_i \tag{2.16}
\]

where \( \alpha_i \geq 0 \) and \( \gamma_i \geq 0 \) are the Lagrange multipliers. Setting partial derivatives of \( L \) with respect to \( R \), \( a \), and \( \xi_i \) to zero provides the following constraints:
\[ \sum \alpha_i = 1 \quad (2.17) \]
\[ a = \sum \alpha_i x_i \quad (2.18) \]
\[ \alpha_i = C - \gamma_i. \quad (2.19) \]

After substituting these constraints to (2.16), the problem becomes:

\[ L = \sum \alpha_i (x_i \cdot x_i) - \sum_{ij} \alpha_i \alpha_j (x_i \cdot x_j). \quad (2.20) \]

As it stems from SVM, SVDD also benefits from the capability of using various kernels to make the description's boundary more flexible thus increase the overall robustness of its classification. Among the most popular choices for kernel is again the Gaussian, whose function is as defined in (2.11). Upon employing a kernel, the inner products in (2.20) are be replaced by the kernel function:

\[ L = \sum \alpha_i K(x_i, x_i) - \sum_{ij} \alpha_i \alpha_j K(x_i, x_j). \quad (2.21) \]

The solution \( \alpha_i \) for \( i = 1, 2, ..., m \) can be obtained via quadratic programming, in particular maximizing (2.21), subject to \( 0 \leq \alpha_i \leq C \) and \( \sum \alpha_i = 1 \). Like SVM, the support vectors for SVDD are all observations that correspond to non-zero \( \alpha \) values (i.e. all \( x_i \) such that \( \alpha_i > 0 \)).

There are several options for quadratic programming solver in R, such as "solve.QP" in package "quadprog", or "ipop" in package "kernlab". All results in this thesis are presented with solutions produced by "ipop", which uses an interior point method which is a very popular choice among many methods to solve quadratic programming problems. Simple simulation-based verifications have confirmed that "solve.QP" would have provided the same results.
2.3. K Chart

Design of the chart of kernel distance (K chart) is similar to that of the Hotelling control chart: observations are enumerated along the horizontal axis; the vertical axis displays kernel distances (rather than Hotelling's $T^2$ statistics as seen in a Hotelling chart); and the most crucial piece of the chart being its control limit, which sets criterion for out-of-control detection. As its name suggested, the K chart is based on the kernel distance in a high-dimensional space between a group of quality characteristics and the kernel center, which is obtained from SVDD's solution. Therefore, the first step in constructing the K chart is performing SVDD.

Upon obtaining the $\alpha_i$ from quadratic programming solver, the center of the hypersphere $a$ can be calculated using (2.18). Then, the kernel-distance from $z$ to the center $a$ is computed by the following equation:

$$d(z, a) = \sqrt{K(z, z) - 2 \sum_i \alpha_i K(z, x_i) + \sum_{ij} \alpha_i \alpha_j K(x_i, x_j)} \quad (2.22)$$

where $z$ denotes the object whose kernel-distance to the center $a$ is being measured, while $x$ represents all observations in the training set. If an object's kernel distance is greater than the control limit, it is declared out-of-control.
2.4. Proposal to Improve the K Chart

The K chart is designed to use SVDD with Gaussian kernel, and while this combination has been proven to be a robust tool for statistical process control (Sun and Tsung, 2003), there are certainly ways to improve it. The Gaussian kernel is based on Euclidean distance between two observation-vectors (for example, \(x\) and \(y\)), which is defined to be:

\[
d(x, y) = \sqrt{(x - y)^T(x - y)}.
\] (2.23)

The Gaussian kernel function (2.11) can be rewritten as followed:
\[ K(x, y) = \exp \left[ -\frac{(x-y)^T (x-y)}{\sigma^2} \right]. \]  \hspace{1cm} (2.24)

where \( \sigma \) is a scale variable that controls shape of the hypersphere. Here the Gaussian kernel function (2.24) includes no covariance matrix in its calculation. Consequently, SVDD with Gaussian kernel has no problem detecting shifts in the mean vector, but is not so sensitive to shifts in covariance. This thesis explores another choice for kernel, which is based on the Mahalanobis distance rather than the traditional Euclidean distance as seen above:

\[ d_M(x, y) = \sqrt{(x - y)^T S^{-1}(x - y)}. \]  \hspace{1cm} (2.25)

It is called Mahalanobis-distance based kernel (hereinafter referred to as \textit{Mahalanobis kernel}), whose function is defined as:

\[ K_M(x, y) = \exp \left[ -\frac{(x-y)^T S^{-1}(x-y)}{\sigma^2} \right]. \]  \hspace{1cm} (2.26)

\( K_M(x, y) \) satisfies the Mercer condition (2.13) mentioned above, so (2.26) is a valid kernel function. The Mahalanobis kernel and distance functions are mostly similar to their Gaussian counterparts, with the only difference being the incorporation of a covariance matrix \( S \) in the calculation. Hence, K chart powered by SVDD with Mahalanobis kernel-distance ought to be more sensitive to changes in the covariance matrix than with Gaussian kernel.
CHAPTER THREE: METHODOLOGY

This thesis uses multiple Monte Carlo simulations to benchmark SVDD using Mahalanobis kernel against SVDD using Gaussian kernel in detecting shifts introduced to mean vector and covariance matrix of the same data. Simulation data are generated from multivariate normal, multivariate Student's (t), and multivariate gamma populations. Both methods are also compared with Hotelling's $T^2$ in multivariate normal case. Bearing in mind that the true objective of this thesis is improvement in terms of detective power to the K chart. The K chart is a "Shewhart type" control chart -- which, as established in the first chapter, benefits from remapping data into rational subgroups. For that reason, all works presented in this thesis are with data segregated into subgroups. Given a data set $(x_1, x_2, x_3, ... x_m)^T$, where $x_i$ is a $p$-dimension vector that is the $i^{th}$ observation among $m$ individual observations. The data set is remapped into $k$ subgroups with the $j^{th}$ group having mean vector $\bar{x}_j$ and covariance matrix $S_j$ as defined by (1.2) and (1.3) respectively. Since these mean vectors $\bar{x}_j$ are registered as SVDD's new training observations, the objective function of SVDD as seen above can be rewritten as:

$$\min \left( R + C \sum \xi_j \right) \quad (3.1)$$

with the constraint:

$$\|\bar{x}_j - a\|^2 \leq R + \xi_j \quad (3.2)$$

where $j = 1, 2, ..., k$. And the Lagrangian (2.16) now becomes:
\[
L(\bar{R}, a, a_j, \gamma_j, \xi_j) = \bar{R} + C \sum \xi_j - \sum a_j \left( \bar{R} + \xi_j - \|\bar{x}_j - a\|^2 \right) - \sum \gamma_j \xi_j. \tag{3.3}
\]

Again, setting partial derivatives of \(L\) with respect to \(\bar{R}, a\), and \(\xi_j\) to zero provides the following constraints:

\[
\sum a_j = 1 \tag{3.4}
\]

\[
a = \sum a_j \bar{x}_j \tag{3.5}
\]

\[
\alpha_i = C - \gamma_j. \tag{3.6}
\]

After substituting these constraints to (3.3), the problem becomes:

\[
L = \sum_j a_j (\bar{x}_j \cdot \bar{x}_j) - \sum_{jl} \alpha_j \alpha_l \bar{x}_j \cdot \bar{x}_l, \text{ where } l = 1, 2, \ldots, k. \tag{3.7}
\]

Upon employing a kernel function, such as (2.24) and (2.26), (3.7) becomes:

\[
L = \sum_j a_j K(\bar{x}_j, \bar{x}_j) - \sum_{jl} \alpha_j \alpha_l K(\bar{x}_j, \bar{x}_l). \tag{3.8}
\]

The support mean vectors in this case are all subgroup means that correspond to non-zero \(\alpha\) values (i.e. all \(\bar{x}_j\) such that \(\alpha_j > 0\)). The Mahalanobis kernel functions (2.26) uses a sample covariance matrix \(S\), which is estimated by the average of all subgroup covariance matrices \(\bar{S}\) given by (1.5). Hence the Mahalanobis kernel tricks in (3.8) can be written as:

\[
K(\bar{x}_j, \bar{x}_l) = \exp \left[ -\frac{(\bar{x}_j - \bar{x}_l)^T S^{-1} (\bar{x}_j - \bar{x}_l)}{\sigma^2} \right]. \tag{3.9}
\]
For Gaussian kernels, $\tilde{S}$ in (3.9) is simply changed to an identity matrix. Choosing a value for $\sigma$ is discussed in the Grid Search section below.

3.1. Simulation Data Generation

Simulations are run in the statistical software environment $R$, on four cases from three multivariate distributions: multivariate normal, multivariate Student's t with 3 and 5 degrees of freedom, and multivariate gamma with an exponential marginal generated through a Wishart's matrix (London and Gennings, 1999). The number of variables is set at $p = 5$. For simulations presented in this thesis, the subgroup size $n$ is set to twice the number of variables. Particularly we have $p = 5$ variables, so subgroup size is set to $n = 10$. Also, $k = 100$ subgroups are desired for each population, so a total number of $m = 1,000$ in-control observations are generated for each case. In-control parameters are mean vector $\mu = (0, 0, ..., 0)^T$ and covariance matrix $\Sigma$ such that:

$$\sigma_{i,j} = \begin{cases} 
2 & i = j \\
0.7|i-j| & i \neq j 
\end{cases}$$

$$\Sigma = \begin{pmatrix}
2.00 & 0.70 & 0.49 & 0.34 & 0.24 \\
0.70 & 2.00 & 0.49 & 0.34 & 0.24 \\
0.49 & 0.70 & 2.00 & 0.70 & 0.49 \\
0.34 & 0.49 & 0.70 & 2.00 & 0.70 \\
0.24 & 0.34 & 0.49 & 0.70 & 2.00 
\end{pmatrix}$$

To generate out-of-control data, shifts ($\delta$) are introduced as ten increments of 0.1 to only the very first element of the mean vector $\mu$ ($\mu_1$), or of the covariance matrix $\Sigma$ ($\sigma_{1,1}$). Denote the out-of-control parameters as $\mu'$ and $\Sigma'$:

21
\[
\begin{align*}
\mu' &= \begin{pmatrix} 0 + \delta \\ 0 \\ 0 \\ 0 \end{pmatrix}, \\
\Sigma' &= \begin{pmatrix} 2.00 + \delta & 0.70 & 0.49 & 0.34 & 0.24 \\ 0.70 & 2.00 & 0.49 & 0.34 & 0.24 \\ 0.49 & 0.70 & 2.00 & 0.70 & 0.49 \\ 0.34 & 0.49 & 0.70 & 2.00 & 0.70 \\ 0.24 & 0.34 & 0.49 & 0.70 & 2.00 \end{pmatrix}
\end{align*}
\]

where \( \delta = 0.1 \times t \) for \( t = 1, 2, 3, \ldots, 10 \). There are some differences among treatments for each population as described below.

3.1.1. Multivariate Normal

Multivariate normal variates are the simplest to generate. In-control data are generated by command "rmvnorm" in R's package "MASS" with mean vector \( \mu \) and covariance matrix \( \Sigma \). Out-of-control data are generated by the same command with \( \mu' \) and \( \Sigma' \) as inputs.

3.1.2. Multivariate Student's

The generating function used for multivariate Student's population is "rmvt" in R's package "mvtnorm". The function takes only degrees of freedom and covariance matrices as inputs, since multivariate t populations assume a location vector of 0. Thus, for in-control observations, 3 and 5 degrees of freedom are entered with \( \Sigma \) as the function's arguments. To generate out-of-control observations: for shifts in covariance matrix, \( \Sigma' \) is used instead of \( \Sigma \); and for shifts in mean vector, \( \mu' \) is added directly to generated variates.
3.1.3. Multivariate Gamma

There is unfortunately no package in R to generate multivariate gamma observations. The algorithm to generate multivariate gamma observations through Wishart's matrices as described by (London and Gennings, 1999) is implemented in R as followed:

1. Generate $n$ multivariate normal observations where $n$ is the size of a rational subgroup.
   Set $n = 10$ for this case. Particularly, generate $X \sim \mathcal{N}_5(0, \Sigma)$, then $X$ is a 10-by-5 matrix.
2. Compute the Wishart's matrix: $W = X'X$
3. Extract the diagonal elements of the Wishart's matrix -- they constitute one observation of a multivariate gamma distribution with an exponential marginal.
4. Repeat step 1–3 until the desired number of observations (i.e. 1,000) is reached

Wishart's matrices are inner products of multivariate normal variates, which must be generated with a zero mean vector; so shifts in mean vector $\mu'$ must be added directly to generated observations like with Student's observations above. However, shifts in covariance matrix can be accounted for by using $\Sigma'$ to generate normal variates in step 1.

3.2. Grid Search

In order to obtain a good data description, two variables $C$ and $\sigma$ must be carefully chosen. Recall that the core algorithm in SVDD seeks to minimize the volume of the hypersphere while at the same time tries to enclose as many data points as possible. At times, a
fraction of the training objects (especially the ones on the outer rim) may be ousted if doing so sufficiently decreases the volume of the description. While on other occasions, the hypersphere may be inflated slightly in order to capture more observations. The variable \( C \) controls such trade-offs. Generally, increasing the value of \( C \) shifts the algorithm's focus from minimizing the volume to maximizing the number of objects captured, as seen in Figure 6 below. The second variable that can directly affect data description is the scale variable \( \sigma \) in the kernel functions, (2.24) and (2.26). While \( C \) controls the volume, \( \sigma \) controls the shape of the hypersphere. In general, a larger value of \( \sigma \) will give the hypersphere a "rounder" appearance, and a smaller value of \( \sigma \) will yield a more flexible shape. Because of that, a higher value of \( \sigma \) is more prone to wrongly capture out-of-control objects. Consequently, \( \sigma \) has a positive relationship with Type II error (or false acceptance) rate, and thus a negative relationship with the Type I error (or false rejection) rate. Deciding on the values for both those \( C \) and \( \sigma \) variables is not a simple task, as the combined effect of such selections will determine the effectiveness of SVDD.
Figure 6: Example of how different values of $C$ and $\sigma$ can affect data description -- a hypersphere projected onto 2-dimensional space here as a yellow perimeter.

(Source: Tax and Duin, 2014)

A grid search algorithm (Tax and Duin, 2001) is used to determine the optimal pair of $C$ and $\sigma$. The only limitation for both $C$ and $\sigma$ is that they both have to be positive, so a range of arbitrary positive values for each variable is initially set (i.e. 0.01 to 100). The range is divided into $r$ equal segments which, with two variables, returns $(r + 1)^2$ different combinations of two values. The process can be visualized (Figure 7) as a $(r + 1)$-by-$(r + 1)$ grid with ranges of $C$ and $\sigma$ as horizontal and vertical axes respectively (hence the name grid search).
SVDD is then performed with each combination of $C$ and $\sigma$ (which can be visualized as an intersection on the grid), and the following heuristic error is calculated:

$$\Lambda(C, \sigma) = \frac{\#SV}{k} + \lambda \#SV_b \left( \frac{\sigma}{s_{\text{max}}} \right)^p$$  \hspace{1cm} (3.10)

where $\#SV$ indicates the number of support vectors (both on, and outside of the boundary); $\#SV_b$ is the number of support vector exactly on the boundary (which gives $0 < \alpha_i < C$). $k$ is the number of $p$-variable training observations, which are the subgroup means in this case; $\lambda$, which regulates the trade-off between the error on training data and on the outlier data, is fixed to 1; $s_{\text{max}}$ is the maximum of kernel distances from each training objects to center of the hypersphere.
The objective of this grid search algorithm is to determine the pair of $C$ and $\sigma$ that yields the best description, such pair should also minimize the heuristic error given by $(3.10)$ (Tax and Duin, 2011). The grid search is repeated several times, each with narrower ranges for $C$ and $\sigma$. At each iteration, heuristic errors are calculated for all $(r + 1)^2$ combinations. Then new ranges for $C$ and $\sigma$ are the immediate values around the intersection that gives the minimum error of the current grid. For example, if the current grid has its minimum error ($\Lambda$) at intersection $(3, 7)$, then the new range for $C$ is defined by the values that correspond to the second and the forth columns, while the new range for $\sigma$ is defined by the values that correspond to the sixth and the eighth rows (Figure 8).

The value of $r$, which directly affects the number of grids, should be chosen with care: a value of $r$ that is too small produces a grid with low resolution, which yields little progress and thus requires many iterations; on the other hands, if $r$ is too large, it will incur a lot of computational expenses along with many wasteful calculations since most results on the grid are discarded after each iteration. For simulations in this thesis, a reasonable value for $r$ is between 10 and 20, and it requires about 4 to 5 iterations to reach a saturated error value.
Figure 8: Example the procedure to determine new ranges for $C$ and $\sigma$.

If the minimum of the current grid is at intersection (3, 7), then the new range for $C$ is given by columns 2 and 4, while the new range for $\sigma$ is given by rows 6 and 8.

The heuristic error is deemed saturated when its value no longer significantly decreases after a new iteration. At such point, grid search is stopped, and the current pair of $C$ and $\sigma$ is considered optimal, then used for actual training with SVDD. For simulations presented in this thesis, most errors don't decrease further (with differences noticeable to 6 decimal places) after 5 iterations. This is done with both Gaussian and Mahalanobis kernels. While both are responsive to grid search (i.e. heuristic errors decrease after each iteration), Gaussian kernel often takes more steps to reach saturation. On the other hands, Mahalanobis kernel may return unsolvable if the lower range for $C$ is too small (i.e. 0.01). This is the infeasibility of the dual problem when
\( C < \frac{1}{k} \) as pointed out by (Cevikalp and Triggs, 2012). Usually this can be remedied by slightly increasing the value for \( C \), as the optimum for \( C \) typically never takes a too small value anyway.

3.3. Monte Carlo Simulation

A Monte Carlo simulation originates from Monte Carlo methods, a family of computational techniques that rely on repeated random sampling to produce numerical results. Invented by Stanislaw Ulam while he was working on nuclear weapon projects at the Los Alamos National Laboratory in the 1940s, Monte Carlo methods were further implemented into computer algorithms on the ENIAC (Electronic Numerical Integrator and Computer) by John von Neumann, one of Ulam's colleagues at Los Alamos. Monte Carlo methods typically involves a great amount of random generations, thus use of computerized algorithms is essential for their efficiencies. Nowadays, Monte Carlo methods are among the most important tools in statistical computing and many other fields that conduct study with computational simulations.

Monte Carlo methods are primarily used for three classes of problems: random variable generation, optimization, and numerical estimation -- the latter of which is employed in this thesis. Suppose the data \( Y_1, Y_2, Y_3, \ldots, Y_N \) are results obtained from executing \( N \) independent runs of a simulation, with \( Y_i \) being output of the \( i^{th} \) run. Presume the objective of the simulation is to estimate some numerical measurement \( L = E(Y) \) and \( |L| < \infty \), then an unbiased estimator for \( L \) is the sample mean of the \( \{Y_i\} \), that is:
\[ \bar{Y} = \frac{1}{N} \sum_{i=1}^{N} Y_i. \]  

(3.11)

Provided that \( \sigma^2 \), the variance of \( Y \), is finite and \( N \geq 30 \), \( \bar{Y} \) approximately has a normal distribution with mean \( L \), and variance \( \frac{\sigma^2}{N} \). In most cases where \( \sigma^2 \) is unknown, it can be estimated with the sample variance of the \( \{Y_i\} \):

\[ S^2 = \frac{1}{N-1} \sum_{i=1}^{N} (Y_i - \bar{Y})^2 \]  

(3.12)

which, by the Law of Large Number, tends to approach \( \sigma^2 \) as \( N \to \infty \). This results in an approximate \((1 - \alpha)\) confidence interval for \( L \) as:

\[ \left( \bar{Y} - Z_{1-\alpha/2} \frac{S}{\sqrt{N}} ; \ \bar{Y} + Z_{1-\alpha/2} \frac{S}{\sqrt{N}} \right) \]  

(3.13)

where \( Z_\theta \) denotes the \( \theta \)-quantile of the standard normal distribution, \( \mathcal{N}(0,1) \). \( \frac{S}{\sqrt{N}} \) is called the estimated standard error.

In this thesis, Monte Carlo simulations are used to obtain the Average Run Length (ARL) -- the target benchmark. A simulation begins by generating a group of observations then measure its kernel distance to the center of the hyperspace obtained from SVDD. Recall that given a data point \( z \), the kernel-distance between \( z \) and the center \( a \) can be calculated by (2.22). If the distance is within the control limit, the group is deemed in-control, the simulation then increments its run count by one, generates another group of observations and repeats. When a group is classified as out-of-control, the simulation stops and records the number of runs it has
managed to that point (run length). The process is repeated 20,000 times; and at the end an average run lengths (ARL) is calculated by (3.11). As the run lengths count how many in-control observations there are until the first out-of-control point is observed, they follow a geometric distribution. The renown Central Limit Theorem states that: given samples of size 30 or more, the sample means follow a normal distribution, regardless what distribution the individual observations have. In this case, each Monte Carlo simulation with \( N = 20,000 \) replications produces 20,000 run lengths, which together can be treated as a sample with size of 20,000, which certainly is greater than 30. Thus, even though the run lengths follow a geometric distribution, the ARLs are normally distributed, and can be compared using confidence intervals constructed with (3.13).

3.3.1. Adjust Control Limit

A temporary control limit is calculated by taking the \( 100(1 - \alpha) \) th-percentile of all kernel distances obtained from in-control (training) observations. Because of that reason, some of the in-control observations are purposely declared out-of-control, particularly the top \( \alpha \) percent that have their kernel distances longer than the temporary control limit. Hence, even if new observations are generated from the same (in-control) population, at some point the simulation will erroneously deem a group as out-of-control, which is a Type I error. Nevertheless, it has been discussed that: since the run lengths will halt at the first out-of-control detection, they follow a geometric distribution. So if a Type I error rate of \( \alpha \) is desired, the ARL or expected value of the run lengths should be \( \frac{1}{\alpha} \). In this thesis, Type I error rate \( \alpha \) is set at 0.005,
so an ARL of 200 would be expected from observations generated from an in-control population.

This can be put in a clearer way as: we want one misclassification (Type I error) for every 200 observations; this ratio gives \( \alpha = 0.005 \).

In reality, using a control limit which is taken at the \( 100(1 - \alpha)^\text{th} \)-percentile of all kernel distances usually doesn't yield an ARL of 200 -- the ARLs tend to fall short of 200. This is mostly due to the limited amount of training objects (100 subgroups) compared to the sheer number of Monte Carlo replications (20,000). Increasing the number of training objects to 20,000 is possible (at least in a simulation setting like this), but not advisable as that is computationally expensive, and unnecessary. In real-world problems, the amount of training observations is typically outclassed by the number of monitoring objects as well. That makes sense as training data are usually limited to some periods of collection, while monitoring process can go on forever, thus potentially provides an infinite amount of data. For the purpose of benchmarking the algorithms, temporary control limits (which are obtained from the percentiles) are adjusted slightly while every other variables are held fixed to bring the ARLs to just about 200. Increasing the control limits makes it harder to classify a group as out-of-control, thus allows the runs lengths to go further before getting halted, consequently increases the ARL. The opposite also holds true: decreasing the control limit should decrease the ARL as well. Basically, once a temporary control limit is obtained from SVDD, it is used in one initial 20,000-replicate Monte Carlo simulation with observations generated from in-control populations (i.e. parameters with 0 shift). If the returned ARL is more than 200 (unlikely), the control limit is slightly decreased, and retried with another simulation. If the returned ARL is less than 200 (most likely),
the control limit is slightly increased, and tried again with another simulation. This process is repeated until an ARL value of approximately 200 is attained. The magnitude of adjustment given to the control limits is largely determined by trial-and-error, but usually not large. From the starting point of 200, the ARLs' behaviors are observed over out-of-control data, which are generated after shifts are introduced into the parameters, as described in the next section.

3.3.2. Simulation on Out-of-Control Data

After all methods have their ARLs set at approximately the same value (i.e. 200) by manipulating the control limits using Monte Carlo simulations on in-control data, we begin introducing shifts to the parameters ($\mu$ and $\Sigma$) to generate out-of-control data, and observe how the ARLs of each method change in response to those new generations. It is found that the ARLs generally shorten as the shifts increase in magnitude (with some exceptions in multivariate t's case, as seen in summary tables below). However, comparing the ARLs at the same shifts reveals which method is more sensitive -- one with significantly shorter ARL at the same shift level must be reacting faster to that change compared to the other(s). These Monte Carlo simulations also run for 20,000 replications; the result ARLs with estimated standard errors are tabulated in the next chapter where findings are presented.
CHAPTER FOUR: RESULTS

4.1. Multivariate Normal

The resulting ARLs from Monte Carlo simulations with multivariate normal variates are summarized in Table 1 below. Only for this case of multivariate normal, Hotelling's $T^2$ is included in comparisons, as the Hotelling chart has been shown to perform worse than K chart with Gaussian kernel for non-normal multivariate data (Sukchotrat et al., 2010).

For changes in the mean vector, in general all three methods manage to pick up the signal well as seen by how the ARLs steadily decrease as the shifts increase. But the Gaussian kernel can only keep up with Mahalanobis kernel on the first, smallest shift of 0.1. From 0.2 onward, SVDD with Mahalanobis kernel bests its Gaussian counterpart by a large margin. $T^2$ loses to SVDD with Gaussian kernel on the first two shifts (0.1 and 0.2) but wins back on the larger shifts (0.3 onward). $T^2$ in fact is almost as good as SVDD with Mahalanobis kernel on most of the mean vector shifts. For changes in the covariance matrix, SVDD with Mahalanobis kernel performs approximately as well as Hotelling's $T^2$, which is expected, as the $T^2$ statistic also incorporates the covariance matrix in its calculation. Regardless, both methods greatly outperform SVDD with Gaussian kernel.

In short, for multivariate normal observations, Mahalanobis kernel performs noticeably better than Gaussian kernel as its ARL decreases at a significantly faster rate for shifts in both mean vector and covariance matrix. Mahalanobis kernel is better than $T^2$ at detecting shifts in
mean vector, despite the latter being a the most popular choice for control chart with multivariate normal populations. Figure 9 and 10 below shows the ARLs with their respective 99% confidence intervals, which are represented by a pair of tiny fences on top of each bar. The huge sample size (20,000 replicates) leads to such small widths for the intervals, even at a high level of confidence (99%). That means the estimations of ARLs here are so precise that if the same simulation is repeated again and again, it will return an ARL value within that (extremely narrow) confidence interval 99% of the times. So that if we find an ARL significantly shorter than (i.e. below the confidence interval of) another, it is likely to be shorter all the times.
Table 1: Averages and standard errors of run lengths from three different methods to detect out-of-control observations for multivariate normal variates generated with shifts in mean vector $\mu$ and covariance matrix $\Sigma$

<table>
<thead>
<tr>
<th>Shift</th>
<th>Gaussian SVDD</th>
<th>Mahalanobis SVDD</th>
<th>Hotelling's $T^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\mu_1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In-control (+0.0)</td>
<td>201.14 ± 1.43</td>
<td>199.66 ± 1.41</td>
<td>200.64 ± 1.40</td>
</tr>
<tr>
<td>+0.1</td>
<td>181.54 ± 1.28</td>
<td>180.65 ± 1.28</td>
<td>189.56 ± 1.33</td>
</tr>
<tr>
<td>+0.2</td>
<td>156.16 ± 1.11</td>
<td>147.86 ± 1.03</td>
<td>159.74 ± 1.13</td>
</tr>
<tr>
<td>+0.3</td>
<td>124.74 ± 0.88</td>
<td>107.36 ± 0.77</td>
<td>119.61 ± 0.83</td>
</tr>
<tr>
<td>+0.4</td>
<td>97.37 ± 0.68</td>
<td>74.68 ± 0.53</td>
<td>83.60 ± 0.58</td>
</tr>
<tr>
<td>+0.5</td>
<td>73.88 ± 0.52</td>
<td>49.48 ± 0.34</td>
<td>55.60 ± 0.38</td>
</tr>
<tr>
<td>+0.6</td>
<td>53.02 ± 0.37</td>
<td>32.65 ± 0.22</td>
<td>36.54 ± 0.26</td>
</tr>
<tr>
<td>+0.7</td>
<td>38.76 ± 0.27</td>
<td>22.06 ± 0.15</td>
<td>24.32 ± 0.17</td>
</tr>
<tr>
<td>+0.8</td>
<td>27.78 ± 0.19</td>
<td>14.70 ± 0.10</td>
<td>16.27 ± 0.11</td>
</tr>
<tr>
<td>+0.9</td>
<td>20.13 ± 0.14</td>
<td>10.25 ± 0.07</td>
<td>11.03 ± 0.07</td>
</tr>
<tr>
<td>+1.0</td>
<td>14.46 ± 0.10</td>
<td>7.18 ± 0.05</td>
<td>7.78 ± 0.05</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{1,1}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+0.1</td>
<td>194.98 ± 1.38</td>
<td>184.30 ± 1.30</td>
<td>185.99 ± 1.30</td>
</tr>
<tr>
<td>+0.2</td>
<td>189.48 ± 1.34</td>
<td>169.06 ± 1.18</td>
<td>169.91 ± 1.19</td>
</tr>
<tr>
<td>+0.3</td>
<td>179.10 ± 1.27</td>
<td>157.16 ± 1.09</td>
<td>155.45 ± 1.09</td>
</tr>
<tr>
<td>+0.4</td>
<td>173.24 ± 1.22</td>
<td>145.33 ± 1.03</td>
<td>142.95 ± 1.00</td>
</tr>
<tr>
<td>+0.5</td>
<td>164.41 ± 1.16</td>
<td>134.40 ± 0.95</td>
<td>131.57 ± 0.92</td>
</tr>
<tr>
<td>+0.6</td>
<td>158.33 ± 1.10</td>
<td>123.17 ± 0.87</td>
<td>122.02 ± 0.86</td>
</tr>
<tr>
<td>+0.7</td>
<td>150.07 ± 1.04</td>
<td>111.92 ± 0.79</td>
<td>112.53 ± 0.80</td>
</tr>
<tr>
<td>+0.8</td>
<td>143.95 ± 1.00</td>
<td>104.27 ± 0.73</td>
<td>102.04 ± 0.72</td>
</tr>
<tr>
<td>+0.9</td>
<td>138.08 ± 0.97</td>
<td>94.78 ± 0.67</td>
<td>94.59 ± 0.66</td>
</tr>
<tr>
<td>+1.0</td>
<td>132.78 ± 0.93</td>
<td>87.98 ± 0.61</td>
<td>86.91 ± 0.61</td>
</tr>
</tbody>
</table>
Figure 9: Average run lengths with 99% confidence intervals on multivariate normal observations generated with shifts in mean vector.
Figure 10: Average run lengths with 99% confidence intervals on multivariate normal observations generated with shifts in covariance matrix.
4.2. Multivariate Student's (t)

4.2.1 Three Degrees of Freedom

The resulting ARLs from Monte Carlo simulations with multivariate Student's variates with 3 degrees of freedom are summarized in Table 2 below.

Table 2: Averages and standard errors of run lengths from both methods to detect out-of-control observations for multivariate Student's variates with 3 degrees of freedom generated with shifts in mean vector \( \mu \) and covariance matrix \( \Sigma \)

<table>
<thead>
<tr>
<th>Shift</th>
<th>Gaussian SVDD</th>
<th>Mahalanobis SVDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-control (+0.0)</td>
<td>200.39 ± 1.41</td>
<td>200.41 ± 1.41</td>
</tr>
<tr>
<td>( \mu_1 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+0.1</td>
<td>201.15 ± 1.42</td>
<td>196.59 ± 1.39</td>
</tr>
<tr>
<td>+0.2</td>
<td>198.28 ± 1.41</td>
<td>196.81 ± 1.38</td>
</tr>
<tr>
<td>+0.3</td>
<td>198.51 ± 1.40</td>
<td>194.59 ± 1.37</td>
</tr>
<tr>
<td>+0.4</td>
<td>195.40 ± 1.39</td>
<td>192.38 ± 1.34</td>
</tr>
<tr>
<td>+0.5</td>
<td>194.79 ± 1.39</td>
<td>184.76 ± 1.30</td>
</tr>
<tr>
<td>+0.6</td>
<td>190.52 ± 1.34</td>
<td>184.71 ± 1.29</td>
</tr>
<tr>
<td>+0.7</td>
<td>189.85 ± 1.33</td>
<td>176.81 ± 1.26</td>
</tr>
<tr>
<td>+0.8</td>
<td>183.24 ± 1.31</td>
<td>172.81 ± 1.23</td>
</tr>
<tr>
<td>+0.9</td>
<td>178.84 ± 1.26</td>
<td>168.84 ± 1.19</td>
</tr>
<tr>
<td>+1.0</td>
<td>177.40 ± 1.26</td>
<td>163.57 ± 1.16</td>
</tr>
<tr>
<td>( \sigma_{1,1} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+0.1</td>
<td>198.38 ± 1.41</td>
<td>196.08 ± 1.38</td>
</tr>
<tr>
<td>+0.2</td>
<td>195.79 ± 1.39</td>
<td>193.60 ± 1.37</td>
</tr>
<tr>
<td>+0.3</td>
<td>189.19 ± 1.34</td>
<td>186.99 ± 1.33</td>
</tr>
<tr>
<td>+0.4</td>
<td>188.88 ± 1.33</td>
<td>182.99 ± 1.30</td>
</tr>
<tr>
<td>+0.5</td>
<td>182.91 ± 1.28</td>
<td>178.35 ± 1.25</td>
</tr>
<tr>
<td>+0.6</td>
<td>181.14 ± 1.28</td>
<td>177.60 ± 1.25</td>
</tr>
<tr>
<td>+0.7</td>
<td>178.78 ± 1.26</td>
<td>172.11 ± 1.22</td>
</tr>
<tr>
<td>+0.8</td>
<td>175.85 ± 1.25</td>
<td>167.89 ± 1.18</td>
</tr>
<tr>
<td>+0.9</td>
<td>173.36 ± 1.22</td>
<td>165.78 ± 1.17</td>
</tr>
<tr>
<td>+1.0</td>
<td>169.06 ± 1.20</td>
<td>162.23 ± 1.14</td>
</tr>
</tbody>
</table>
Figure 11: Average run lengths with 99% confidence intervals on multivariate Student's observations with 3 degrees of freedom, generated with shifts in mean vector.
Figure 12: Average run lengths with 99% confidence intervals on multivariate Students observations with 3 degrees of freedom, generated with shifts in covariance matrix.
4.2.2. Five Degrees of Freedom

The resulting ARLs from Monte Carlo simulations with multivariate Student's variates with 5 degrees of freedom are summarized in Table 3 below.

Table 3: Averages and standard errors of run lengths from both methods to detect out-of-control observations for multivariate Student's variates with 5 degrees of freedom generated with shifts in mean vector $\mu$ and covariance matrix $\Sigma$

<table>
<thead>
<tr>
<th>Shift</th>
<th>Gaussian SVDD</th>
<th>Mahalanobis SVDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-control (+0.0)</td>
<td>199.81 ± 1.41</td>
<td>201.41 ± 1.43</td>
</tr>
<tr>
<td>$\mu_1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+0.1</td>
<td>204.46 ± 1.45</td>
<td>192.45 ± 1.36</td>
</tr>
<tr>
<td>+0.2</td>
<td>200.86 ± 1.42</td>
<td>179.79 ± 1.27</td>
</tr>
<tr>
<td>+0.3</td>
<td>195.90 ± 1.40</td>
<td>163.62 ± 1.15</td>
</tr>
<tr>
<td>+0.4</td>
<td>185.96 ± 1.30</td>
<td>141.86 ± 1.00</td>
</tr>
<tr>
<td>+0.5</td>
<td>173.18 ± 1.21</td>
<td>121.91 ± 0.86</td>
</tr>
<tr>
<td>+0.6</td>
<td>159.87 ± 1.14</td>
<td>100.92 ± 0.71</td>
</tr>
<tr>
<td>+0.7</td>
<td>140.62 ± 1.00</td>
<td>82.37 ± 0.58</td>
</tr>
<tr>
<td>+0.8</td>
<td>124.85 ± 0.88</td>
<td>65.23 ± 0.46</td>
</tr>
<tr>
<td>+0.9</td>
<td>107.27 ± 0.75</td>
<td>51.11 ± 0.36</td>
</tr>
<tr>
<td>+1.0</td>
<td>89.86 ± 0.64</td>
<td>39.48 ± 0.28</td>
</tr>
<tr>
<td>$\sigma_{1,1}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+0.1</td>
<td>193.12 ± 1.36</td>
<td>192.71 ± 1.34</td>
</tr>
<tr>
<td>+0.2</td>
<td>186.58 ± 1.31</td>
<td>182.05 ± 1.29</td>
</tr>
<tr>
<td>+0.3</td>
<td>182.65 ± 1.30</td>
<td>175.10 ± 1.24</td>
</tr>
<tr>
<td>+0.4</td>
<td>180.38 ± 1.26</td>
<td>167.17 ± 1.19</td>
</tr>
<tr>
<td>+0.5</td>
<td>171.54 ± 1.21</td>
<td>158.86 ± 1.12</td>
</tr>
<tr>
<td>+0.6</td>
<td>165.85 ± 1.16</td>
<td>148.17 ± 1.05</td>
</tr>
<tr>
<td>+0.7</td>
<td>160.57 ± 1.14</td>
<td>143.22 ± 1.02</td>
</tr>
<tr>
<td>+0.8</td>
<td>154.82 ± 1.09</td>
<td>134.66 ± 0.95</td>
</tr>
<tr>
<td>+0.9</td>
<td>151.28 ± 1.07</td>
<td>127.28 ± 0.90</td>
</tr>
<tr>
<td>+1.0</td>
<td>147.25 ± 1.03</td>
<td>120.56 ± 0.85</td>
</tr>
</tbody>
</table>
Figure 13: Average run lengths with 99% confidence intervals on multivariate Student's observations with 5 degrees of freedom, generated with shifts in mean vector
Figure 14: Average run lengths with 99% confidence intervals on multivariate Students observations with 5 degrees of freedom, generated with shifts in covariance matrix
For multivariate t with three degrees of freedom, both methods are able to detect the shifts as apparent by their decreasing average run lengths. However, despite the shifts are of the same magnitudes as that in the multivariate normal case, neither of the methods can pick up as quickly. For example at +1.0 shift in mean vector, in multivariate normal case, the ARLs for both Gaussian and Mahalanobis are approximately 14 and 7, respectively. Yet at the same shift in multivariate t with 3 degrees of freedom, the ARLs are about 177 and 163. Slower rates of descend (compared to that in multivariate normal case) are also observed for changes in covariance matrix. Nevertheless, SVDD with Mahalanobis kernel outperforms SVDD with Gaussian kernel as evident by its significantly lower average run lengths for shifts in both the mean vector and the covariance matrix.

For multivariate t with five degrees of freedom, again both methods manage to detect the changes in both the mean vector and the covariance matrix. Compared to three degrees of freedom case, there are some improvements in both methods' sensitivities as their average run lengths decrease at a faster rate, though still not as good as what is seen with multivariate normal. At +1.0 shift in mean vector now the ARLs of 90 and 39 are observed for Gaussian and Mahalanobis kernels, in that order, compared to 177 and 163 with three degrees of freedom. This is expected as when the value of degrees of freedom gets larger, the Student's t distribution approaches normality, thus detection power (sensitivity) is expected to increase with the degree of freedom. Still, results indicate that the Mahalanobis kernel also performs better than the Gaussian kernel in this case.
4.3. Multivariate Gamma

The resulting ARLs from Monte Carlo simulations with multivariate gamma variates are summarized in Table 4 below.

Table 4: Averages and standard errors of run lengths from both methods to detect out-of-control observations for multivariate gamma variates generated with shifts in mean vector $\mu$ and covariance matrix $\Sigma$

<table>
<thead>
<tr>
<th>Shift</th>
<th>Gaussian SVDD</th>
<th>Mahalanobis SVDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-control (+0.0)</td>
<td>221.44 ± 1.55</td>
<td>220.06 ± 1.56</td>
</tr>
<tr>
<td>$\mu_1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+0.1</td>
<td>220.64 ± 1.56</td>
<td>217.48 ± 1.53</td>
</tr>
<tr>
<td>+0.2</td>
<td>222.69 ± 1.58</td>
<td>211.06 ± 1.50</td>
</tr>
<tr>
<td>+0.3</td>
<td>221.78 ± 1.56</td>
<td>207.51 ± 1.46</td>
</tr>
<tr>
<td>+0.4</td>
<td>220.06 ± 1.54</td>
<td>204.37 ± 1.44</td>
</tr>
<tr>
<td>+0.5</td>
<td>216.95 ± 1.53</td>
<td>197.66 ± 1.42</td>
</tr>
<tr>
<td>+0.6</td>
<td>215.56 ± 1.52</td>
<td>192.44 ± 1.35</td>
</tr>
<tr>
<td>+0.7</td>
<td>211.47 ± 1.49</td>
<td>188.91 ± 1.34</td>
</tr>
<tr>
<td>+0.8</td>
<td>209.99 ± 1.49</td>
<td>181.47 ± 1.28</td>
</tr>
<tr>
<td>+0.9</td>
<td>204.85 ± 1.44</td>
<td>175.46 ± 1.23</td>
</tr>
<tr>
<td>+1.0</td>
<td>204.48 ± 1.46</td>
<td>168.32 ± 1.18</td>
</tr>
<tr>
<td>$\sigma_{1,1}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+0.1</td>
<td>169.91 ± 1.20</td>
<td>146.55 ± 1.04</td>
</tr>
<tr>
<td>+0.2</td>
<td>99.66 ± 0.70</td>
<td>83.03 ± 0.59</td>
</tr>
<tr>
<td>+0.3</td>
<td>53.45 ± 0.38</td>
<td>43.74 ± 0.30</td>
</tr>
<tr>
<td>+0.4</td>
<td>28.16 ± 0.20</td>
<td>23.85 ± 0.17</td>
</tr>
<tr>
<td>+0.5</td>
<td>15.80 ± 0.11</td>
<td>13.71 ± 0.09</td>
</tr>
<tr>
<td>+0.6</td>
<td>9.50 ± 0.06</td>
<td>8.60 ± 0.06</td>
</tr>
<tr>
<td>+0.7</td>
<td>6.21 ± 0.04</td>
<td>5.66 ± 0.04</td>
</tr>
<tr>
<td>+0.8</td>
<td>4.38 ± 0.03</td>
<td>4.03 ± 0.02</td>
</tr>
<tr>
<td>+0.9</td>
<td>3.20 ± 0.02</td>
<td>3.02 ± 0.02</td>
</tr>
<tr>
<td>+1.0</td>
<td>2.53 ± 0.01</td>
<td>2.40 ± 0.01</td>
</tr>
</tbody>
</table>
This case of multivariate gamma observations is quite unique compared to the others. First, the in-control average run lengths are not set to approximately 200 as seen for the other three multivariate distributions. Recall that the first step of the Monte Carlo simulations presented in this thesis is bringing the in-control ARLs to 200 for comparison purpose. And that is achieved by manually adjusting the control limits in a trial-and-error manner. From what is experienced with the other multivariate distributions above, control limits typically do not require large adjustments to bring the ARL to 200, and the relationship between the control limit and the average run length is found to be, likely, non-linear. Those findings are preserved for the multivariate gamma case here, except that the aforementioned relationship is now extremely steep. That means even a minuscule adjustment to the control limit results in a massive leap of the ARL for the multivariate gamma observations. For this simulation, a great deal of trials have been attempted to bring the ARLs to 200 to no avail, the ARL still bounces back and forth between approximately 190 and about 220 for changes as little as one unit in nineteenth decimal places ($10^{-19}$). In the end, due to time and resources constraints, the ARLs are settled at approximately 220 for both kernels as seen in Table 4. The conjecture is that if the control limit keeps getting adjusted further, eventually an ARL of 200 will be attained. But first of all, it may take a long time to reach that value; and secondly, the main objective of this thesis is not achieving an ARL of 200, but rather comparison between SVDD using Gaussian kernel and SVDD using Mahalanobis kernel, which requires achieving an ARL agreement between both methods and it doesn't have to be 200. Also, an $\alpha$ value of 0.005 (or less) is desired. Hence, 220 is chosen as it corresponds to $\alpha = \frac{1}{220} \approx 0.0045$. 

47
While both methods manage to detect shifts in the mean vector, it is obvious, from Table 4 above and Figure 15 below, that the Mahalanobis kernel is the one with better performance as its ARLs remain persistently shorter than that of the Gaussian kernel throughout all the shifts. However, the rates of ARL descend as shift magnitude increases for both kernels are not as great as what are observed in multivariate normal case before, indicating that: for changes in mean vector, these methods are not as sensitive for multivariate gamma as for multivariate normal observations.

On the other hands, both methods display great power when it comes to detecting shifts in covariance matrix, as seen in Table 4 above and Figure 16 below. By far this is the greatest response rate (of ARL descend) in all simulations presented. Recall that each multivariate gamma observation is a vector of diagonal elements of a Wishart's matrix, which is an inner product-matrix of a set of multivariate normal variates generated with mean 0 and covariance matrix \( \Sigma \). Any shifts in \( \Sigma \) may be reflected on the normal generations, and magnified with the calculations of Wishart's matrices, and thus becoming easier for SVDD to detect. Nevertheless, Mahalanobis kernel is again found to be better than Gaussian kernel at detecting shifts in covariance matrix for multivariate gamma observations.
Figure 15: Average run lengths with 99% confidence intervals on multivariate gamma observations generated with shifts in mean vector.
Figure 16: Average run lengths with 99% confidence intervals on multivariate gamma observations generated with shifts in covariance matrix.
CHAPTER FIVE: CASE STUDY

5.1. Overview

The data set used is from an ambulatory blood pressure monitoring research project, which originates from the Halberg Chronobiology Center in Roseville, Minnesota. Each observation in the data set (hereinafter referred to as Halberg data) is a weekly average of 4 variables: systolic blood pressure, diastolic blood pressure, heart rate, and arterial pressure (Maboudou and Hawkins, 2013). There are a total of 320 observations in the data set, which is divided into two halves. The first half is used for training purpose with SVDD using both Gaussian and Mahalanobis kernels, and the second half of is used to demonstrate monitoring process. Both the training and monitoring sets are further segmented into rational subgroups. With the fact that individual observations are weekly averages, rational subgroup size is set to 5, thus making new observations (groups) a 5-week (or a little more than month-long) averages. This results in 36 training and 36 monitoring objects. Grid search is used to determine optimal pairs of $C$ and $\sigma$ to train with SVDD using both kernels. This has been much the same as a simulation as described above, but here is as far as that similarity goes. First, there are only 36 training objects available; a simple percentile-based control limit obtained from such small sample will be abysmal. Second, since the distribution of individual observations (before segregation into subgroups) is unknown, it is not possible to rely on Monte Carlo simulations to adjust control limits and find ARLs like above. In short, something must be done to get a better
control limit, and another scheme to benchmark the methods' performances is also required. Those two issues are addressed in the next sections.

5.2. Use Bootstrap to Obtain Control Limit

Published by Bradley Efron in 1979, *bootstrap* is a method that relies on *random sampling with replacement* to perform testing or estimation when the theoretical distribution of a statistic of interest is complex or unknown, or when the sample size is too small to make straightforward statistical inference (Adèr et al., 2008). Random sampling with replacement indicates a sampling scheme in which the randomly selected element is returned to the pool of selection so it may be chosen again, thus an element may appear multiple times in one sample. The basic idea of bootstrap is that: even when the population is unknown, inference or estimation regarding some parameter can be modeled by resampling the sample data, effectively simulating the population and thus allowing parameter inference or estimation.

In this case study on Halberg data, upon calculating the kernel-distances of all 36 training observations from the center of the hyperspace which is obtained from SVDD, bootstrap is employed to estimate the control limit instead of finding a simple $100(1 - \alpha)^{th}$-percentile of the kernel distances. The procedure (Sukchotrat et al., 2010) is as followed:

1. Calculate the kernel distances to the center $a$: $D_i = d(z_i, a)$ using (2.22), with $z_i$ being the $i^{th}$ training object, and $i = 1, 2, ..., 36$. 

52
2. Sample $D_i$ with replacement to obtain $B$ bootstrap samples of size 36. For this case study, $B$ is set at 5,000 which should result in 5,000 bootstrap samples.

3. Obtain $L_j$, the $100(1 - \alpha)^{th}$-percentile of the $j^{th}$ bootstrap sample, for $j = 1, 2, \ldots, 5000$.

4. The control limit is estimated as the average of all the $100(1 - \alpha)^{th}$-percentile values from $B$ bootstrap samples: $CL = \sum_{j=1}^{5000} L_j$

The bootstrap control limit is then used in monitoring process with the K chart.

5.3. Results

Upon obtaining a control limit from bootstrapping as discussed in the previous section, a K chart can then be constructed for monitoring purpose. The control limit is displayed as a horizontal line across the chart, taking a single value on the vertical axis, which represents kernel distance. The horizontal axis plots the objects that are under monitoring process. Each of 36 objects in the monitoring set one-by-one has its kernel-distances to the kernel center $a$ calculated using (2.22). If such distance is less than or equal to the control limit, the current observation is deemed in-control, and the monitoring proceeds to the next one. As soon as an object is declared out-of-control (i.e. its kernel distance to the center is found greater than the control limit), the process halts. Since the training set, then the monitoring set are the same for both SVDD using Gaussian and SVDD using Mahalanobis kernels, whichever of the two methods detects an out-of-control point sooner must be the better one. Figure 17 and Figure 18 below shows the K charts that are powered by SVDD using Gaussian kernel and Mahalanobis kernel, respectively.
Figure 17: Monitoring process on the second half of Halberg data by a K chart constructed with SVDD using Gaussian kernel
Figure 18: Monitoring process on the second half of Halberg data by a K chart constructed with SVDD using Mahalanobis kernel
It appears that both methods have found an out-of-control object relatively early into the monitoring process, yet they disagree on which one. While the Gaussian kernel reports the eighth object (in fact a subgroup mean) as being out-of-control, the Mahalanobis kernel insists that it is actually the seventh. In the simulation study conducted above, all variates generated within the Monte Carlo simulations (besides those in the adjust control limit phase) are out-of-control -- as they are drawn from populations with shifted parameters. So that, any out-of-control flags in the simulations are valid. That is not true in this case, as the Halberg data set is not labeled, it is unknown which object is actually out-of-control; thus when two methods give two different answers, it is hard to immediately tell which one is correct. A solution for this problem is conducting a hypothesis testing on those two objects to find out which one of them is actually out-of-control.

5.4. Multivariate Kruskal-Wallis Test

As established above, since the distribution of Halberg data is unknown (as most data sets in real-world problems), any testing procedures that are based on any distributional assumptions are not valid in this case. So it has to be a nonparametric test -- that is the first key point. The objective of the test is determining whether any of the seventh and the eighth objects are out-of-control, where being out-of-control means the object(s) follows a different distribution than that of in-control population. Recall that the first half of the Halberg data are used as in-control training objects; hence if any of the objects in question can be determined to follow a different distribution than that of the first half set, they must be out-of-control. So the second key point is:
the hypothesis test is going to be a test of distribution, which can tell whether or not two samples come from the same population distribution. While there are many choices of nonparametric distribution tests, such as: \( \chi^2 \) Goodness-of-Fit, Mann-Whitney-Wilcoxon, Kolmogorov-Smirnov, et cetera, the problem is that: not all of them have a multivariate counterpart, which is desired in this case.

A multivariate approach for Kruskal-Wallis test for analysis of variance has been proposed by (Choi and Marden, 1997). The procedure is as followed. Given a sample \( A \) of \( p \)-dimension observations: \( A = (X^{(1)}, X^{(2)}, \ldots, X^{(n)}) \), the general centered and scaled rank function of an observation \( X^{(i)} \) within \( A \) is defined by:

\[
R(X^{(i)}) = \frac{1}{n} \sum_{j=1}^{n} \frac{X^{(j)} - X^{(i)}}{\|X^{(j)} - X^{(i)}\|}.
\]  

(5.1)

A within-group rank function is required when observations need to be separated into groups. Let \( M \) be a subset of indices \( \{1, 2, \ldots, n\} \) and let \( m = \#M \). For \( i \in M \), the within-group rank function of \( X^{(i)} \) among observations indexed by \( M \) is:

\[
R_M(X^{(i)}) = \frac{1}{m} \sum_{j \in M} \frac{X^{(j)} - X^{(i)}}{\|X^{(j)} - X^{(i)}\|}.
\]  

(5.2)

Suppose \( K \) groups are being test for distribution, where for each \( k = 1, 2, \ldots, K \), \( N_k \) consists of the indices for the observations in group \( k \) and \( n_k = \#N_k \). Assume all observations are independent and for each \( k \), the observations \( X^{(i)} \) for \( i \in N_k \) have distribution \( F_k \). The test hypotheses are:
\[ H_0: F_1 = F_2 = \cdots = F_K \quad (\text{versus}) \quad H_a: F_i \neq F_j, \text{ for some } i \neq j. \tag{5.3} \]

Let \( \bar{R}^{(k)} \) be the average group rank for group \( k \), with the ranks here calculated relative to all observations (i.e. the \textit{entire} sample, not just the observations in group \( k \)):

\[
\bar{R}^{(k)} = \frac{1}{n_k} \sum_{i \in N_k} R_{N_k} \left( X^{(i)} \right). \quad (5.4)
\]

The pooled estimated of the covariance of the rank vector is:

\[
\tilde{\Sigma}_n = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i \in N_k} R_{N_k} \left( X^{(i)} \right) R_{N_k} \left( X^{(i)} \right)'. \quad (5.5)
\]

The test statistic is defined by:

\[
KW = \sum_{k=1}^{K} n_k \left( \bar{R}^{(k)} \right)' \tilde{\Sigma}_n^{-1} \bar{R}^{(k)}. \quad (5.6)
\]

Under the null hypothesis, we have:

\[
KW \rightarrow \chi^2_{p(K-1)}. \quad (5.7)
\]

The testing procedure is programmed into R for execution. Two tests are conducted separately on subgroup number 8 (as flagged by the Gaussian kernel) and subgroup number 7 (as indicated by the Mahalanobis kernel) to see whether they follow the distribution of the first half set, which is training data and assumed to be in-control. It should be noted that while subgroup means are used for training SVDD and making predictions in the K chart, the procedure of this multivariate Kruskal-Wallis test calls for individual observations, so the data set is returned to its original shape for testing purpose.
Let \( F_{IC} \) be the distribution of the in-control data (first half of the data set that is used for training purpose), and \( F_8 \) be the distribution of the observations in the eighth subgroup, while observations in the seventh subgroup have \( F_7 \) distribution. The test hypotheses are:

Test 1: \[ H_0: F_8 = F_{IC} \quad \text{(versus)} \quad H_a: F_8 \neq F_{IC} \]

Test 2: \[ H_0: F_7 = F_{IC} \quad \text{(versus)} \quad H_a: F_7 \neq F_{IC} \]

The test statistic for Test 1 is found to be \( KW = 13.777 \), which in turn gives a p-value of \( 0.008 \). Recall that \( \alpha \) level is set at 0.005. So the test statistic fails to reject the null hypothesis and cannot conclude that the observations in the eighth subgroup do not have the same distribution as the in-control observations. In other words, there is not enough evidence to declare the eighth subgroup out-of-control. The test statistic for Test 2 is calculated to be \( KW = 25.073 \), which gives a p-value of \( 4.864 \times 10^{-5} \) or 0.00004864. This results in rejecting the null hypothesis of Test 2, and conclude that the observations in the seventh subgroup do not have the same distribution as the in-control observations. In other words, there is sufficient evidence to declare the seventh subgroup out-of-control.

The two instances of multivariate Kruskal-Wallis test provide conclusive evidence that the seventh subgroup is out-of-control so the earlier decision of the K chart that is constructed with SVDD using Mahalanobis kernel is correct. There are several potential explanations on the why the K chart with Gaussian SVDD picks the eighth subgroup rather than the seventh. First of all, the p-value of the test statistic obtained from observations in the eighth group is 0.008, which is a close call. Setting \( \alpha \) level at 0.05 or 0.01 may have declared the group out-of-control.
Secondly, how SVDD structures its description plays a vital role in deciding the effectiveness of the model (such as K chart). The fact that the chart skips group number seven and picks up number eight does not mean it can't detect any anomaly signal from seven, it just takes a longer time (i.e. one extra period) to respond. Anyhow, this case study has showed that: even when knowing nothing about the data's distribution, SVDD with Mahalanobis kernel is still indeed more sensitive than SVDD with Gaussian kernel in detecting out-of-control objects -- further strengthening the finding that is obtained from the simulations above.
CHAPTER SIX: CONCLUSION

Powered by support vector data description (SVDD), the K chart is an important tool for statistical process control. SVDD benefits from a wide variety of kernel choices to make accurate classifications. Native to creation of the K chart, the Gaussian kernel is the most popular choice for SVDD as it offers the method a limitless degree of flexibility to describe data.

This thesis proposes to incorporate an even-more-robust Mahalanobis kernel into SVDD to improve the K chart's performance. Benchmarked by Average Run Length (ARL), results obtained from Monte Carlo simulations on three different multivariate distributions show that SVDD using Mahalanobis kernel is more sensitive than SVDD using Gaussian kernel for detecting shifts in both mean vector and covariance matrix. SVDD using Mahalanobis kernel even surpasses Hotelling's $T^2$ statistic in multivariate normal case, which has always been considered the latter's forte. A case study using real data also finds that the Mahalanobis kernel improves the K chart's ability to make timely and more accurate out-of-control detection over the Gaussian kernel.
LIST OF REFERENCES


