EIGENVALUES AND EIGENVECTORS IN DATA DIMENSION REDUCTION FOR REGRESSION

THESIS

Presented to the Graduate Council of
Texas State University-San Marcos
in Partial Fulfillment
of the Requirements

for the Degree

Master of SCIENCE

by

Randolf H. Reiss, B.S.

San Marcos, Texas
August 2013

EIGENVALUES AND EIGENVECTORS IN DATA DIMENSION REDUCTION FOR REGRESSION

Committee Members Approved:

Qiang Zhao, Chair

Alexander White

Jian Shen

Approved:

J. Michael Willoughby
Dean of the Graduate College

COPYRIGHT

by

Randolf H. Reiss

2013

FAIR USE AND AUTHOR'S PERMISSION STATEMENT

Fair Use

This work is protected by the copyright laws of the United States (Public Law 94-553, section 107). Consistent with fair use as defined in the Copyright Laws, brief quotations from this material are allowed with proper acknowledgement. Use of this material for financial gain without the author's express written permission is not allowed.

Duplication Permission

As the copyright holder of this work, I, Randolf H. Reiss, authorize duplication of this work, in whole or in part, for educational or scholarly purposes only.

ACKNOWLEDGEMENTS

I would like to extend thanks to everyone who has helped in the research and development of this thesis. I would like to thank the committee members for their consideration and care taken in reading and making corrections. A special thanks goes to my wife Amantha and my daughter Arden for their support, encouragement, and understanding throughout my graduate education. I would also like to thank Dr. Qiang Zhao for embarking on this research project. Without his the encouragement, direction, and knowledge, this thesis would not have been possible.

This manuscript was submitted on May 6th, 2013.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

ABSTRACT

EIGENVALUES AND EIGENVECTORS IN DATA DIMENSION REDUCTION FOR REGRESSION

by

Randolf H. Reiss, B.S.

Texas State University-San Marcos
May 2013

SUPERVISING PROFESSOR: QIANG ZHAO

A basic theory of eigenvalues and eigenvectors as a means to reduce the dimension of data, is presented. Iterative methods for finding eigenvalues and eigenvectors are explored with proofs of the existence and uniqueness of solutions. Of particular focus is the Power Method as it is the basis of most eigenvector algorithms. Interpretations of the Power Method are presented in the context of linear algebra and data dimension reduction. It is shown that the algorithms for principal component analysis and partial least squares are extensions of the Power Method. The estimation of parameters for a computer-based pharmaceutical bioreactor simulator is presented as an application. Diagnostics methods of ordinary multiple least squares regression are applied to partial least squares, including detection of hidden extrapolation.

CHAPTER 1

INTRODUCTION

1.1 The Mathematics of Data Analytics

The era of big data is here (Singh and Singh 2012, 1-4 ; Office of the White House, 2012). Technological advancements in the past ten years has enabled a fruition of data collection and storage systems of mammoth proportions (XueFeng et al. 2011, 3220–3232). Following the successful implementation of many data warehouses, there has been a realization that data, no matter how much, is worthless unless it can be transformed into relevant and concise information (Wegman and Solka 2005, 1-54). The quip "data rich and information poor" summarizes the situation of many organizations in the private and public sector (Han and Kamber 2006, 1-9).

Organizations implementing data analytics include businesses looking for a commercial benefit. The industrial sector, including pharmaceutical manufacturers, are no exception. Modern computer production and process control systems are personal computer based and leverage the low cost of consumer computer equipment. The nature of industrial production often produces time series data of many variables that result in large dimensional data sets. An important selling point of many these computer control systems is the ability to store massive amounts of production data with the assumption that something could be done with it. Although not as well publicized as Google® or

Facebook®, the industrial sector is at the forefront of big data and eager to see a return on their investment in data centers. The need to perform data analytics on this type of data is growing.

On the front lines of the deluge of data are the computer scientists and programmers that implement the intricate systems of data collection and storage. Their response is a growing collection of highly creative procedures and ad hoc algorithms classified under the terms of *knowledge discovery in databases* (KDD) and *data mining* (Maimon and Rokach 2005, 1-13). Many books on the subject of data mining describe a multidisciplinary framework of machine learning, statistics, pattern recognition, information retrieval, neural networks, knowledge-based systems, artificial intelligence, high-performance computing, and data visualization (Han and Kamber 2006, xxi; Olson and Delen 2008, 3-4). Others even attempt to differentiate data mining from mathematics and statistics by asserting that statistical methods are just not capable of dealing with the demand of big data (Sumathi and Sivanandam 2006, 2-5).

However, all data mining algorithms, at some level, require the use of numbers, equations and formulas. In order to produce reliable results, many of the nascent ad hoc algorithms of data mining will need to mature to the rigor of mathematic theory. The approach taken in this study is to build on mathematic theory to avoid reinventing the wheel and ensure algorithms are more than an aberration of a particular data set. The theory of statistics is based on mathematics and is primarily concerned with the inference of information from quantitative data. This definition aligns with the goal of most data mining algorithms and, as such, the fundamentals of mathematical and statistical theory should be applied (Wasserman 2004, vii-viii). That is, the solution to big data is mathematical theory and statistical inference.

What appears to be the defining feature of data mining is that it is concerned with the analysis of very large sets of data. Although the size of the data is unprecedented, so is the storage capacity, memory size, and computational capability of modern processors. Thus, the size of the data in relation to the computational power is not a new problem, but it is a problem. However, it is of interest to notice that the collection and analysis of large scale data is no longer a centralized activity of governments or huge multi-nation conglomerates. The relatively low cost of data collection equipment and network connectivity has allowed medium and small organizations, and even individuals, to engage in the practice data collection and analysis. Thus, what is really new about big data and data mining is who is doing it.

When the size of the data to be analyzed is larger than the memory or processing capability of a computer, some reduction in size is required to perform meaningful analytics. Data can be reduced in two ways; reducing the number of records in the database or reducing the size of each record. Reducing the number of records is an application of statistical sampling theory and information on methods can be found in (Lavallee 2007; Good 2006). The primary focus of this study is the reduction in the size of each record. This is also known as a reduction in the dimension of the data.

A regression method uses the relationships between known data to infer an unknown response. Regression methods are often employed to establish relationships in data, classify data, or make predictions. These are often the same goals as data mining. This study explores the mathematics of certain methods of regression that involve data dimension reduction. Specifically, the study focuses on eigenvectors and eigenvalues of data matrices and their use in dimension reduction for regression. Chapter 2 contains a description of the eigenvalue problem and an intuitive introduction to dimension reduction. The mathematical theory of eigenvectors and eigenvalues is presented in Chapter 3. Chapter 4 includes details on the relevant methods for computing eigenvectors and eigenvalues. The regression methods of principal component regression and partial least squares are covered in Chapter 5. Chapter 6 describes an application of partial least squares for estimation of parameters for a pharmaceutical bioreactor simulator. The study concludes with some final remarks and a discussion on future research directions in chapter 7.

1.2 Notation

Unless otherwise specified, the following conventions are established. A vector is defined as $n \times 1$ column of scalars where n is a natural number. A matrix is any collection of p vectors where p is a natural number. All vectors and matrices are elements of a finite dimensional vector space that is defined over the field of real numbers. Matrix and vector addition and multiplication are as usually defined for a real-valued vector space (Halmos 1974, 3-5).

Notation:

$\mathbb{R}$ is the set of real numbers

$\mathbb{C}$ is the set of complex numbers

a is a scalar, real or complex.

a is a vector.

$\|a\|_2 = \sqrt{a_1^2 + a_2^2 + \ldots + a_n^2}$ is the 2-norm of an *n* dimensional vector ***a*** and a_i are the components of ***a***.

A is a matrix.

$\bar{c}$ is the complex conjugate of the vector *c* and is defined as $\overline{(a+bi)} = (a-bi)$ where *i* is the imaginary unit and *a,b* are real numbers.

A' is the transpose of the matrix *A*.

A^* is the conjugate transpose of the matrix *A*.

A^{-1} is the inverse matrix of the matrix *A*.

A^n is the *n*th power of a square matrix and is defined as the product of *n* many *A* matrices.

CHAPTER 2

THE EIGENVALUE PROBLEM AND DIMENSION REDUCTION

2.1 The Eigenvalue Problem

The eigenvalue problem can be considered as the problem of finding the roots of the characteristic polynomial of a square matrix. The *characteristic polynomial* of the square matrix A is defined as $det(A- \lambda I)$ where *det* is the determinant of a matrix, I is the identity matrix of appropriate dimension, and λ is a scalar. This interpretation of the eigenvalue problem is important for proving existence of eigenvalues, but less insightful for the study of the dimension reduction of data. A more intuitive view of the eigenvalue problem is as an equality that establishes a relationship between a matrix, a scalar, and a vector. The eigenvalue problem is defined to be the scalar λ that satisfies the equation

$$A\boldsymbol{x}=\lambda\boldsymbol{x} \tag{1}$$

where A is an $n \times n$ square matrix and $\boldsymbol{x}$ is an $n \times 1$ vector and $\boldsymbol{x} \neq \mathbf{0}$ (Wilkinson, 1965, 2; Horn 1985, 1-77). A solution is an *eigenvalue* λ and an *eigenvector* $\boldsymbol{x}$, also referred to as an *eigenpair,* that satisfies (1). The relationship established by (1) is that multiplication of A on $\boldsymbol{x}$ has the same result as multiplication of the scalar λ on $\boldsymbol{x}$. This relationship is an important aspect of the eigenvalue problem. To investigate this relationship further, some basis definitions and concepts of linear algebra must be introduced.

Definition 1. A set of vectors $\boldsymbol{x}_i$ is said to be *linearly dependent* if there exists a set of scalars α_i, not all equal to zero, such that $\sum_{i=1}^{p} \alpha_i \boldsymbol{x}_i = 0$.

If only two vectors are considered, then they are linearly dependent if one vector is a scalar multiple of the other.

Definition 2. A set of vectors are said to be *linearly independent* if they are not linearly dependent.

Definition 3. When $\boldsymbol{x} = \sum_{i=1}^{p} \alpha_i \boldsymbol{x}_i$, then $\boldsymbol{x}$ is a *linear combination* of the set of vectors $\boldsymbol{x}_i$.

Definition 4. A *vector space* is a set of vectors that is defined over a field of scalars that satisfy the following axioms:

1. For all vectors $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ in the vector space, the operation of addition is defined such that;

   a) Addition of vectors is commutative, $\boldsymbol{x}+\boldsymbol{y} = \boldsymbol{y}+\boldsymbol{x}$.

   b) Addition of vectors is associative, $(\boldsymbol{x}+\boldsymbol{y})+\boldsymbol{z} = \boldsymbol{x}+(\boldsymbol{y}+\boldsymbol{z})$.

   c) There exists a unique vector $\mathbf{0}$ in the vector space such that $\boldsymbol{x}+\mathbf{0} = \mathbf{0}+\boldsymbol{x} = \boldsymbol{x}$.

   d) For each $\boldsymbol{x}$ in the vector space, there exists a unique vector $-\boldsymbol{x}$ such that $\boldsymbol{x}+(-\boldsymbol{x}) = (-\boldsymbol{x})+\boldsymbol{x} = \mathbf{0}$

2. For all vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ in the vector space and all scalars a and b in the field of scalars defined by the vector space, the operation of scalar multiplication is defined such that;
   a) Scalar multiplication is associative, $a(b\boldsymbol{x}) = (ab)\boldsymbol{x}$.
   b) There exists a unique scalar 1 in the field of scalers such that $1\boldsymbol{x} = \boldsymbol{x}$.
   c) Scalar multiplication is distributive with respect to vector addition, $a(\boldsymbol{x}+\boldsymbol{y}) = a\boldsymbol{x}+a\boldsymbol{y}$.
   d) Multiplication by vectors is distributive with respect to scalar addition, $(a+b)\boldsymbol{x} = a\boldsymbol{x}+b\boldsymbol{x}$.

Notice that a vector space is defined over a field of scalars. A definition of a field of scalars and a detailed description of the its relation to a vector space can be found in (Halmos 1974, 3-7).

Definition 5. A *basis* of a vector space is defined as a set of linearly independent vectors such that every vector in the space can be written as a linear combination of the basis vectors.

Definition 6. The *span* of a set of vectors is the space defined by all linear combinations of the set of vectors.

Definition 7. The *rank* of a matrix is the minimum number of linearly independent vectors required to span the same space as the columns of the matrix.

Definition 8. The *nullity* of a matrix is the difference between the matrix rank and the dimension of the vector space.

Consider a column vector of n scalars representing a point in an n-dimensional vector space and an $n \times n$ square matrix A. The matrix A performs a linear transformation on a n-dimensional vector $\boldsymbol{x}$ by matrix multiplication, noted as $A\boldsymbol{x}$. The linear transformation can be decomposed into n linear functionals, each represented by a row of A. The linear functional in the ith row of A is multiplied by the vector $\boldsymbol{x}$ and the result is the ith entry in the transformed column vector. Thus, the effect of a linear transformation is made up of the effects of the individual linear functionals (rows) of the matrix A. The transformation is linear in that the linear functionals can only perform scaling on each component of the column vector. As a result, a linear transformation preserves the operations of vector addition and scalar multiplication (Halmos 1974, 20).

The classic description of the effect of a linear transformation on a vector is that of a rotation, a reflection, and a shear (Lay 2003, 82-87). However, a shear can be decomposed into a reflection and a dilation, and a reflection can be expressed as a rotation about the origin and a dilation. Thus, any linear transformation also can be expressed as a rotation and a dilation. This interpretation of the effect of a linear transformation on a vector is helpful in seeing the intuitive effects of linear transformations.

Consider the linear transformation A in (1) on the eigenvector $\boldsymbol{x}$. The effect of A on $\boldsymbol{x}$ is the same effect the multiplication of $\boldsymbol{x}$ by the scalar λ. Thus, A only has the effect of a dilation on $\boldsymbol{x}$, but not a rotation and the specific amount of the dilatation is λ. The implication of the relationship between the eigenpair and the transformation matrix concerns the invariance of rotation of the transformation in the direction of the eigenvector. As a result, instead of needing to transform the vector $\boldsymbol{x}$ by the entire matrix A, only the single scalar λ is required. The direction described by $\boldsymbol{x}$ is an important property of the linear transformation A.

A basis of a vector space can be constructed, starting with the eigenvector $\boldsymbol{x}$ and adding linearly independent vectors until the set of vectors span the vector space (Halmos 1974, 11). As a result, only the scalar λ is required to perform the linear transformation A on the component of a vector in the direction of $\boldsymbol{x}$. For example, let $\boldsymbol{y}$ be a vector in the vector space. Then $\boldsymbol{y}$ may be expressed as a linear combination of the basis vectors. Let a be the coefficient for the basis vector $\boldsymbol{x}$ in the linear combination. The vector $A\boldsymbol{y}$ can also be expressed as a linear combination of the basis vectors and would have the coefficient $a\lambda$ for the basis vector $\boldsymbol{x}$. This shows the importance of an eigenvector as a basis vector. Specifically, the effect of the linear transformation A on the basis vector $\boldsymbol{x}$ has been reduced from the multiplication of vectors to the multiplication of scalars. Notice that the coefficients of $A\boldsymbol{y}$ for the basis vectors that are not also eigenvectors do not have this simple relationship.

If there is an n-dimensional vector space, then a minimum of n vectors are required to span the vector space. If a matrix has n columns and the rank of the matrix is less than n, then the columns of the matrix are linearly dependent and at least one of the columns may be written as a linear combination of one or more of the remaining columns. In this case, at least one column may be discarded and the rank of the matrix remains the same. A *minimal spanning set* is the smallest number of vectors required to span a vector space and a *maximal linearly independent* set is the largest set of vectors in a vector space that are still linearly independent. Both result in a *basis* of the vector space (Halmos 1974, 10).

Consider the linear transformation A with rank n. It would be desirable to find n linearly independent vectors that have the same property of our eigenpair $\boldsymbol{x}$ and λ. It will be shown that, not only can such a set of vectors be found to satisfy this requirement, but the set will also be orthogonal.

Definition 9. Two vectors are *orthogonal* if their dot product is equal to 0.

Orthogonality of two vectors may be noted as $\boldsymbol{x'y} = \boldsymbol{y'x} = 0$. As set of vectors are called orthogonal if all the vectors are pair-wise orthogonal.

Definition 10. A *projection* is an idempotent linear transformation where idempotent is defined as the property of a linear transformation A for which $A^2 = A$.

Consider three vectors $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$. The projection of $\boldsymbol{x}$ onto $\boldsymbol{y}$ in the direction of $\boldsymbol{z}$ is the vector in the direction of $\boldsymbol{y}$ with the length determined by moving the end point of $\boldsymbol{x}$ in the direction of $\boldsymbol{z}$ until is intersects the vector $\boldsymbol{y}$. Notice that the vector $\boldsymbol{z}$ is not required to be orthogonal to $\boldsymbol{x}$ or $\boldsymbol{y}$. However, the special case when $\boldsymbol{z}$ is orthogonal to $\boldsymbol{y}$ is called an *orthogonal projection* of $\boldsymbol{x}$ onto $\boldsymbol{y}$. Orthogonal projections are common enough that it is assumes that "a projection of $\boldsymbol{x}$ onto $\boldsymbol{y}$" is an orthogonal projection. The case when $\boldsymbol{z}$ is not orthogonal to $\boldsymbol{y}$ is called an *oblique projection* or *non-orthogonal projection* and will always be explicitly defined as such. Projections are an important part of dimension reduction.

The dot product of two vectors can be thought of as the scalar component of a orthogonal projection (Thomas 2005, 867). Notice that, if $\boldsymbol{x}$ and $\boldsymbol{y}$ are orthogonal, a projection of $\boldsymbol{x}$ onto $\boldsymbol{y}$ (or $\boldsymbol{y}$ onto $\boldsymbol{x}$) results in a vector of length 0. Thus, if $\boldsymbol{x}$ and $\boldsymbol{y}$ are orthogonal basis vectors, then the projection of some vector $\boldsymbol{a}$ along $\boldsymbol{x}$ onto $\boldsymbol{y}$ is an orthogonal projection. Likewise, the projection of $\boldsymbol{a}$ along $\boldsymbol{y}$ onto $\boldsymbol{x}$ is an orthogonal projection. Also notice that the result of each projection is a term of the linear combination of the basis vectors that uniquely expresses $\boldsymbol{a}$. More specially, the coefficient of the term in the direction of $\boldsymbol{x}$ is the dot product of $\boldsymbol{x}$ and $\boldsymbol{a}$ and the coefficient of the term in the direction of $\boldsymbol{y}$ is the dot product of $\boldsymbol{y}$ and $\boldsymbol{a}$.

For example, consider the three dimensional euclidean space and the vectors $\boldsymbol{x} = [1,0,0]'$, $\boldsymbol{y} = [0,1,0]'$, and $\boldsymbol{z} = [0,0,1]'$. This set of vectors satisfies a minimal spanning set and a maximal linearly independent set in the vector space and is, therefore, a basis for the vector space. It is also easily shown that these particular basis vectors are orthogonal and have unit length. Consider some vector $\boldsymbol{a}$ in the vector space. Then the linear combination expressing $\boldsymbol{a}$ can be constructed by the orthogonal projection of $\boldsymbol{a}$ onto each basis vector. Let $\boldsymbol{a}\cdot\boldsymbol{x}=1$, $\boldsymbol{a}\cdot\boldsymbol{y}=2$, and $\boldsymbol{a}\cdot\boldsymbol{z}=3$, then $1\boldsymbol{x}$ is the orthogonal projection of $\boldsymbol{a}$ onto $\boldsymbol{x}$, $2\boldsymbol{y}$ is the orthogonal projection of $\boldsymbol{a}$ onto $\boldsymbol{y}$, and $3\boldsymbol{z}$ is the orthogonal projection of $\boldsymbol{a}$ onto $\boldsymbol{z}$. Thus, $\boldsymbol{a}$ may be uniquely expressed as the linear combination $\boldsymbol{a} = 1\boldsymbol{x}+2\boldsymbol{y}+3\boldsymbol{z}$. Assuming the basis as described, we may express $\boldsymbol{a}$ by the coordinates, noted as $\boldsymbol{a} = [1,2,3]'$. Notice that, if the basis vector were not orthogonal, the projections would not be orthogonal and the above computation would be more complicated. This simple example illustrates that a change of basis to an orthogonal basis is as simple as an orthogonal projection onto each basis vector.

Now consider that a matrix may have more than one eigenpair. In fact, it will be shown that a matrix will have the same number of eigenpairs as the rank of the matrix. In the context of a linear transformation, when each basis vector is an eigenvector of the transformation and the eigenvectors are orthogonal, then the linear transformation can be reduced to orthogonal projections. As each orthogonal projection is defined by the dot product of the vector in the direction of each basis vector and, as each basis vector is also an eigenvector of the transformation, the transformation can be expressed as the matrix of the corresponding eigenvalues. That is, we may express a vector as a linear combination of the eigenvectors of the transformation because the eigenvectors are also a basis of the vector space. Thus, the effect of the transformation on each component of the linear combination of the vector can be simplified to scalar multiplication by the corresponding eigenvalue. As a result, the linear transformation may be expressed as a diagonal matrix consisting of the eigenvalues. This is the theory behind the preceding example.

The key is that the change of basis is a linear transformation of the transformation matrix and the specific basis is the set of the eigenvectors of the linear transformation. Thus, a linear transformation with n distinct eigenvalues may be represented as a diagonal matrix and is described as *diagonable* (Halmos 1974, 108) or *diagonalizable* (Friedberg, Insel, and Spence 1989, 216). Notice that this would not be possible if all n eigenvectors of the linear transformation did not exist. With these basic concepts of linear algebra introduced, some intuitive concepts of eigenvalues and eigenvectors will be presented.

2.2 Dimension Reduction of the Data Matrix

Consider a simplified example to illustrate some basic concepts of data dimension reduction. A pharmaceutical drug company uses a reactor to manufacture a new drug. The drug is produced by adding ingredients into the reactor and a process of mixing, pressurization, heating, and cooling results in the drug. The reactor has two sensors that the company feels are important and they are recorded for every batch of the drug that is produced. The sensor readings that are recorded are the maximum temperature and maximum pressure in the reactor during the processing of a batch. The data is stored in a $n \times 2$ matrix where n is the number of batches that have been recorded. Thus, each column of the matrix represents the data from a sensor and each row represents a batch of the drug. Each sensor (variable) is n-dimensional and each batch (observation) is 2-dimensional.

Definition 11. A *data matrix* is a matrix of data values arranged such that each column represents a variable and each row represent an observation (Khattree and Naik 2000, 1-7; Gentle 2007, 8).

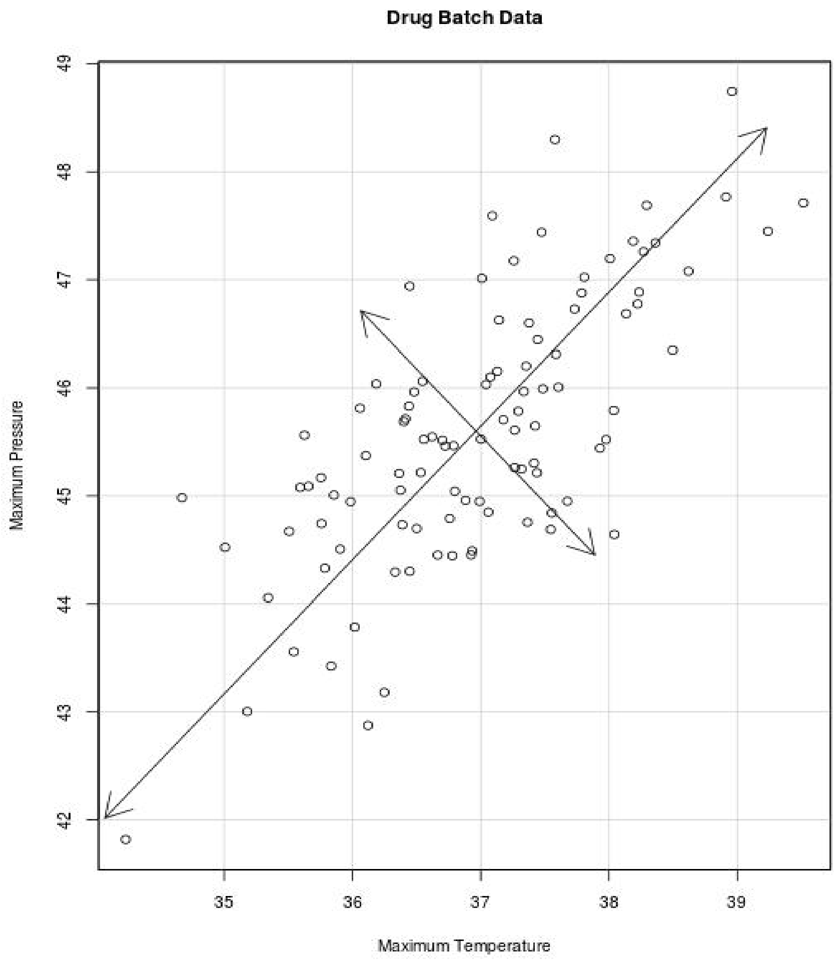

Figure 2.1: Direction of maximum variance of drug batch data.

Consider the graph of the drug batch data in Figure 2.1 with the two arrows to indicate the direction of the greatest variation of the data. Notice that the arrows are perpendicular to one-another but are not parallel to each axis. The direction and length of the arrows in Figure 2.1 are of interest. In the context of reducing the number of variables, while retaining as much information as possible, it makes sense to consider the data in the direction of the longer arrow. The process of extracting this data is a projection onto the vector in the direction of the longer arrow. After this data is extracted, what is left behind is the data in the direction of the second (smaller) arrow. As a result, the arrows are perpendicular and, in fact, they have to be as all the data in the direction of the previous arrow has been removed from the matrix. It is intuitive to think of the arrows as a set of basis vectors that are inherent to the shape of the data. Expressing the data as linear combinations of vectors in the direction of the arrows facilitate the projections required to reduce the dimension of the data. These are key concepts that will be used in the reduction of the dimension of a data matrix.

The arrows in Figure 2.1 are, in fact, eigenvectors that relate to the data matrix and the projection of the data onto these eigenvectors can be performed by a change of basis or linear transformation of the data. Because the eigenvectors are orthogonal, the matrix of the transformation is the matrix of eigenvectors. With properly preprocessed data, the eigenvalues indicate the variance of the data in the direction of each associated eigenvector. Thus, eigenpairs hold important information about the size and shape of the data in a data matrix.

Consider the eigenvalues as the variance of the data in the direction of each eigenvector. If a large amount of variation occurs in one direction, then a small amount of variation will occur in another direction. Thus, eigenvalues have a proportional relationship to one another. Consider the example of the drug manufacturer data in Figure 2.1. The readings of temperature and pressure are from the same reactor and this often means that there is a direct relation. Figure 2.1 indicates that this appears to be the case for this particular data. Notice that the length of the longer arrow (first eigenvalue) is significantly larger than the length of the second arrow (second eigenvalue). This indicates that there exists a stronger correlation between the columns of the data matrix in the direction of the first eigenvector than the second. By correlation, it is meant that there is a linear relationship or non-zero covariance between columns of the data matrix.

Perfectly correlated data is identical to the concept of linearly dependence. That is, if a set of columns are linearly dependent, then, at least one column may be written as a linear combination of the other columns. With many data sources, including industrial sensor reading, there is rarely a perfect linear correlation between columns of a data matrix. When there is a near-dependence relation among the columns, the data is said to have a condition called *multicollinearity in regression* (Montgomery, Peck, and Vining 2006, 111). Another way to think of multicollinearity is that one column can be well predicted by the other columns. Multicollinearity may occur when multiple columns have a common source of information. For example, when two sensors monitor the same physical phenomena, then there may be a correlated response to a perturbation. A more complex example comes from the use of time series data. Considering the drug production example, if data were collected throughout the production run of the batch, then we would have multiple readings of the same sensors at different points in time. The data can be unfolded where each sensor reading at each point in time is a column of the data matrix. As a result, the set of columns of the sensor reading of the same sensor at different points in time will usually have a multicollinear relationship. Thus, a data matrix can be multicollinear by design.

Also notice there may not be multicollinearity of the columns, but, as the number of the columns in the data matrix increases, there can only be an increase in the dependency relation among the entire set of columns. To see this consider that most sources of data are not uncorrelated. Thus, each variable has some level of correlation with the other columns. As columns are added, this level of correlation can only increase. Instead of a few columns having a near-dependent relation, there are many columns may have a low pair-wise dependence relation that, all together, represent enough overlap to result in a near-dependency of some columns. In summary, it can be said that there may be an increased dependency relationships between the columns of a data matrix as columns are added.

However, the level of dependence between the columns is also an opportunity to reduce the dimension of the data. If two columns have a near-dependent relationship, then one column may be removed from the data matrix with little loss of information. More often is the case where many columns are correlated, but removing any one columns may also cause an undesired loss of information. That is, each column contains some redundant information and some unique information. The better solution is to condense the columns into a smaller number of columns or smaller dimension that removes the overlap while retaining as much of the unique information as possible. The optimal condensing may be achieved by a projection of the data in the directions of the greatest variation of the data. This is a primary concept is dimension reduction with eigenpairs.

This is the big picture of dimension reduction with eigenpairs. Uncorrelated information in a data matrix may be represented by projections onto inherent structures of the data described by eigenpairs. However, the presentation of these concepts and intuitive interpretations has lacked the rigor of mathematics. A mathematical foundation for eigenpairs must be established to validate these assertions.

CHAPTER 3

MATHEMATICAL THEORY OF EIGENVALUES AND EIGENVECTORS

The validity of the relationship between eigenpairs and data matrices can only be established with the rigor of mathematical proofs. The existence of real eigenpairs of symmetric matrices with real coefficients is presented, followed by the required association of an eigenpair to a non-square data matrix. The theory begins with the existences of eigenvalues.

3.1 Eigenvalues and Eigenvectors of Matrices

Consider an eigenvalue as a root of the characteristic polynomial defined as $det(A- \lambda I)$ where *det* is the determinant of a square matrix and I is the identity matrix of the appropriate dimension. Induction on *The Fundamental Theorem of Algebra* is appropriate to show the existence of roots of a characteristic polynomial (eigenvalues) of a square matrix.

Theorem 1: The Fundamental Theorem of Algebra. Let $p(z)$ be a non-constant polynomial with a complex variable and complex coefficients with degree $n>0$ where n is an natural number. Then $p(z)$ has a root.

Proof:

Suppose $p(z)\neq 0$ for all $z\in\mathbb{C}$. Then $|p(z)|=|a_n z^n+a_{n-1} z^{n-1}+....+a_0|$.

By the triangle inequality,

$|a_n||z^n|-|a_{n-1}||z^{n-1}|-....-|a_0| \le |p(z)| \le |a_n||z^n|+|a_{n-1}||z^{n-1}|+....+|a_0|$, which implies $|z|^n\left[|a_n|-\frac{|a_{n-1}|}{|z|}-\frac{|a_{n-2}|}{|z^2|}-....-\frac{|a_0|}{|z^n|}\right] \le |p(z)| \le |z|^n\left[|a_n|+\frac{|a_{n-1}|}{|z|}+\frac{|a_{n-2}|}{|z^2|}+....+\frac{|a_0|}{|z^n|}\right]$. As

$|z|\to\infty$, $|z|^n\left[|a_n|-\frac{|a_{n-1}|}{|z|}-\frac{|a_{n-2}|}{|z^2|}-....-\frac{|a_0|}{|z^n|}\right]\to\infty$ and $|z|^n\left[|a_n|+\frac{|a_{n-1}|}{|z|}+\frac{|a_{n-2}|}{|z^2|}+....+\frac{|a_0|}{|z^n|}\right]\to\infty$.

By the squeeze principle, $|p(z)|\to\infty$. Then there exists real numbers $M > 0$ and $r > 0$ such that $|p(z)|>M$ when $|z|>r$, which implies that $\frac{1}{|p(z)|}\le\frac{1}{M}$ for $|z|>r$. Thus, $\frac{1}{|p(z)|}$ is bounded on the compact set $|z|\le r$. Then, by the Liouville Theorem, $\frac{1}{|p(z)|}$ is a constant function, which implies $|p(z)|$ is a constant function. This is a contradiction. Therefore, *p*(*z*) must have a root. □

Detailed information on Theorem 1 can be found in (Fine and Rosenberger 1997, 70-71; Remmert, 1991, 266-267; Truss 1997, 200-203; Hirsch 1974, 329-330; Kuttler 2004, 387-388). The general approach of the proof of Theorem 1 is credited to Joseph Liouville who is also credited with the Liouville Theorem which is used in the proof but not proved here. Proofs of the Liouville Theorem can be found in (Fine and Rosenberger 1997, 70; Remmert, 1991, 244-247).

Theorem 2. Let *p*(*z*) be a complex polynomial with degree *n>0*, then *p*(*z*) has *n* roots.

Proof:

Let *p*(*z*) be a complex polynomial. We note that if *p*(*z*) has degree 1, then Theorem 1 is the same as Theorem 2. Let *p*(*z*) be have degree *n*>1. By Theorem 1, *p*(*z*) has at least one complex root a. Then, $p(z)$ can be written as $p(z) = (z-a)h(z)$ where $h(z)$ is a complex polynomial of degree n-1 and, by theorem 1, has at least one complex root. Therefore, by induction on the degree of the polynomial, $p(z)$ has a n roots.□

It is assumed that zero may be a root of $p(z)$. In fact, there may be multiple roots of $p(z)$ with the value zero. Theorem 3 constructs the *minimal polynomial* of $p(z)$ with non-zero roots (Hoffman 1971, 191).

Theorem 3. Let $p(z)$ be a complex polynomial with degree $n>0$. If zero is a root of $p(z)$, then there exists a polynomial $m(z)$ with degree $k<n$ such that $m(z)$ has the same non-zero roots as $p(z)$.

Proof:

Let $p(z)$ be a complex polynomial with degree $n>0$ where p(0)=0. By theorem 2, $p(z)$ has n roots. Then, we may write $p(z) = (z-b_n)(z-b_{n-1})...(z-b_{i+1})(z-b_i)(z-b_{i-1})...(z-b_1)$ where $b_i=0$ for some i such that $0 \leq i \leq n$. Define $m(z) = (z-b_n)(z-b_{n-1})...(z-b_{i+1})(z-b_{i-1})...(z-b_1)$ and see that $m(z)$ has the same non zero roots as $p(z)$ and the degree of $m(z)$ is $n-1$. The process can be repeated for each $b_i = 0$. Therefore, there exists a polynomial with degree $k<n$ with the same non-zero roots as $p(z)$.□

Corollary 3.1 pulls the ideas together from the first three theorems to prove that non-zero square matrices have non-zero eigenvalues.

Corollary 3.1. Every $n \times n$ matrix A, where A is not all zeros, has k non-zero eigenvalues such that $k \leq n$ where k and n are natural numbers.

Proof:

Define the characteristic polynomial of A as as $det(A- \lambda I)$. Then the characteristic polynomial of A has order n. By theorem 3, there exists a minimal polynomial with order $k \leq n$. Therefore, A has k non-zero eigenvalues.□

Proofs can be found with the similar results to corollary 3.1 in (Piziak 2007, 329; Beezer 2004, 457-458).

Note that an eigenvalue may have an algebraic multiplicity greater than one. That is, two or more eigenvalues may exist with the same value. Thus, the characteristic polynomials may contain a factor of the form $(z-b)^g$ where $g>1$. Then we simply point out that $(z-b)^g = (z-b)(z-b)$ for $g = 2$ and there exists two roots equal to b. Thus, the above theorems count each algebraic multiplicity as a separate eigenvalue associated with a single eigenvector. It is noted that the existence of multiple roots of the same value does cause difficulties for algorithms that compute eigenpairs. In the special case where there is more than one eigenvalue that are equal to zero, then the set of corresponding eigenvectors span the null space of the matrix. A complete handling of the subject of eigenvalue multiplicity can found in (Halmos 1974, 104-105; Shores 2007, 258; Bernstein 2009, 245-247).

It is possible for a matrix with real values to have complex eigenvalues just as it is possible for a polynomial with real coefficients have complex roots. However, the roots of a complex valued polynomials can only have complex roots. Thus, the set of complex numbers is referred to as algebraically closed and, as a result, all matrices with complex values have complex eigenvalues. To obtain the results desired, it is required that the scope of the matrices considered be narrowed to only those matrices that are equal to their own conjugate transpose, noted as $A=A^*$. The conjugate transpose of a matrix is the matrix constructed by taking the transpose of the original matrix and then taking the complex conjugate of each entry. A matrix that is equal to its own complex conjugate is called *self-adjoint* or a *Hermitian* matrix. Of importance is that a Hermitian matrix has only real-valued eigenvalues and eigenvectors.

Theorem 4. The eigenvalues of a Hermitian matrix are real.

Proof:

Let A be a Hermitian matrix. By Corollary 2.1, A has n eigenvalues. Suppose λ is a complex eigenvalue and $\boldsymbol{v}$ is the associated complex valued eigenvector. Then, by the definition of an eigenvalue, $A\boldsymbol{v} = \lambda\boldsymbol{v}$. Define the scalar $c = \boldsymbol{v}^*A\boldsymbol{v}$ then $\bar{c} = c^* = (\boldsymbol{v}^*A\boldsymbol{v})^*$ $= \boldsymbol{v}^*A\boldsymbol{v} = c$ implies that c is a real number. Then, $c = \boldsymbol{v}^*A\boldsymbol{v} = \boldsymbol{v}^*\lambda\boldsymbol{v} = \lambda\boldsymbol{v}^*\boldsymbol{v}$ implies $\lambda = c/(\boldsymbol{v}^*\boldsymbol{v})$. As $\boldsymbol{v}^*\boldsymbol{v}$ represents sum of squared complex values, it must be a real value. Thus, λ is real. Therefore, the eigenvalues of A are real. □

Additional information and proofs of Theorem 4 can be found in (Shores 2007, 283; Bapat, 1993, 22; Wilkinson 1965, 25; Bronson 1989, 121).

Theorem 5. The eigenvectors of a Hermitian matrix have only real-values.

Proof:

Let A be a Hermitian matrix. By Theorem 4, there exists a real eigenvalue λ of A. Then there exists a vector $\boldsymbol{v}$ such that $A\boldsymbol{v}= \lambda\boldsymbol{v}$. Suppose that $\boldsymbol{v}$ has only an imaginary component, then define $\boldsymbol{y}=i\boldsymbol{v}$ is a real-valued vector. $A\boldsymbol{y} = A(i\boldsymbol{v}) = i(A\boldsymbol{v}) = i(\lambda\boldsymbol{v}) = \lambda(i\boldsymbol{v}) = \lambda\boldsymbol{y}$ implies that $\boldsymbol{y}$ is a real-valued eigenvector corresponding to the eigenvalue λ. Suppose that $\boldsymbol{v}$ has a real and imaginary component. The define $\boldsymbol{y} = \boldsymbol{v} + \bar{\boldsymbol{v}}$, then **y** is real-valued. Then consider $A\boldsymbol{y} = A(\boldsymbol{v}+\bar{\boldsymbol{v}}) = A\boldsymbol{v}+A\bar{\boldsymbol{v}} = A\boldsymbol{v} + \overline{A\boldsymbol{v}} = \lambda\boldsymbol{v} + \overline{\lambda\boldsymbol{v}} = \lambda\boldsymbol{v} + \lambda\bar{\boldsymbol{v}} = \lambda(\boldsymbol{v} + \bar{\boldsymbol{v}}) = \lambda\boldsymbol{y}$ implies that $\boldsymbol{y}$ is a real-valued eigenvector of of the eigenvalue λ. Therefore, the eigenvectors of a Hermitian matrix are real. □

The reader is referred to (Bronson 1989, 121; Corrochano 2005, 132-133) for more information on theorem 5.

Another desired property of eigenvectors is orthogonality. Theorem 6 provides this result for Hermitian matrices.

Theorem 6. The eigenvectors corresponding to distinct eigenvalues of a Hermitian matrix are orthogonal.

Proof:

Let A be a Hermitian matrix. Let $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$ be the eigenvectors corresponding to distinct eigenvalues $\{ \lambda_1, \lambda_2, \ldots, \lambda_n\}$. By Theorems 4 and 5, the eigenpairs are real-valued. Then $\lambda_k\boldsymbol{v}_j^*\boldsymbol{v}_k = \boldsymbol{v}_j^*\lambda_k\boldsymbol{v}_k = \boldsymbol{v}_j^*A\boldsymbol{v}_k = (A\boldsymbol{v}_j)^*\boldsymbol{v}_k = (\lambda_j\boldsymbol{v}_j)^*\boldsymbol{v}_k = \lambda_j\boldsymbol{v}_j^*\boldsymbol{v}_k$ implies $\lambda_k\boldsymbol{v}_j^*\boldsymbol{v}_k = \lambda_j\boldsymbol{v}_j^*\boldsymbol{v}_k$. Then, $\lambda_k\boldsymbol{v}_j^*\boldsymbol{v}_k - \lambda_j\boldsymbol{v}_j^*\boldsymbol{v}_k = 0$ implies $(\lambda_k - \lambda_j)\, \boldsymbol{v}_j^*\boldsymbol{v}_k = 0$. Then, $\lambda_k \neq \lambda_j$ implies $(\lambda_k - \lambda_j) \neq 0$. Thus,

$\boldsymbol{v}_j^*\boldsymbol{v}_k = 0$. Therefore, the eigenvectors of A are pairwise orthogonal.□

A thorough discussion of the results of Theorem 6 can be found in (Shores 2007, 284; Wilkinson 1965, 26).

Before preceding to the next theorem, a definition of an *orthogonal matrix* is presented. The property of an orthogonal matrix V that will be used in the proof of Theorem 7 is that $V'=V^{-1}$. That is, the transpose of an orthogonal matrix is equal to its inverse.

Definition 12. Let $\{\boldsymbol{v}_n\}$ be a finite set of vectors. If $\boldsymbol{v}_i'\boldsymbol{v}_j = 0$ for all $i \neq j$ and $\boldsymbol{v}_i'\boldsymbol{v}_i = 1$, then $\{\boldsymbol{v}_n\}$ is an *orthonormal* set of vectors (Halmos 1974, 122) and the matrix consisting of columns of $\{\boldsymbol{v}_n\}$ is called an *orthogonal matrix* or an *orthonormal matrix* (Strang 1988, 167; Parlett 1998, 92; Shilov 1977, 239; Stewart 1998, 56-57).

Theorem 7 proves an essential connection between a data matrix and eigenpairs. For any non-square real matrix A, the eigenvalues of $A'A$ and AA' are equal. The theorem further establishes a relationship between the matrix A and the eigenpairs of $A'A$ and AA'. It is shown that A can be written as a product of matrices that are made up of the eigenvectors and eigenvalues of $A'A$ and AA'. This is important as most data matrices are not square and, as such, do not have eigenpairs. However, it will be shown that the eigenpairs of $A'A$ and AA' are related to the structure of the data in the matrix A.

Theorem 7. Let A be a matrix with real values. Then the eigenvalues of $A'A$ and AA' are equal and $A = U\Sigma V'$ where the orthogonal matrices U and V consist of columns of the eigenvectors of $A'A$ and AA' respectively and the matrix Σ is all zeros except the diagonal elements that consists of the square roots of eigenvalues of $A'A$ and AA'.

Proof:

Let A be a real-valued n x p matrix where $p<n$. Then $A'A$ and AA' are symmetric real-valued (Hermitian) matrices. Then there exists a set of vectors $\{\boldsymbol{v}_1,\ldots\boldsymbol{v}_r\}$ of non-zero eigenvectors of $A'A$ corresponding to eigenvalues $\{\lambda_1,\ldots,\lambda_r\}$ where $0 \leq r \leq p$. By the definition of an eigenpair, $A'A\boldsymbol{v}_i = \lambda_i\boldsymbol{v}_i$. Left multiplication by A yields $AA'(A\boldsymbol{v}_i) = \lambda_i(A\boldsymbol{v}_i)$ implies that $A\boldsymbol{v}_i$ and λ_i are eigenpairs of AA'. Thus, $A'A$ and AA' have the same eigenvalues. Define $\mathbf{u}_\mathbf{i} = A\boldsymbol{v}_i / \lambda_i^{1/2}$ for each $\boldsymbol{u}_i$ in $\{\boldsymbol{u}_1,\ldots, \boldsymbol{u}_s\}$ where $0 \leq s \leq n$. Then $\{\boldsymbol{u}_1,\ldots, \boldsymbol{u}_s\}$ are the unit length eigenvectors of AA'. Consider $\boldsymbol{u}_i'\mathrm{A}\, \boldsymbol{v}_j = (A\boldsymbol{v}_i / \lambda_i^{1/2})' A\boldsymbol{v}_j = \boldsymbol{v}_i' A' (1/ \lambda_i^{1/2})\, A\boldsymbol{v}_j$ $= \boldsymbol{v}_i' (A'A\, \boldsymbol{v}_j / \lambda_i^{1/2}) = \boldsymbol{v}_i' (\lambda_j\, \boldsymbol{v}_j / \lambda_i^{1/2}) = \boldsymbol{v}_i'\, \boldsymbol{v}_j (\lambda_j / \lambda_i^{1/2})$. Because $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are orthogonal, $\boldsymbol{v}_i'\, \boldsymbol{v}_j (\lambda_j / \lambda_i^{1/2}) = 0$ when $\mathrm{i} \neq j$. Because $\boldsymbol{v}_i$ is a unit vector, $\boldsymbol{v}_i'\, \boldsymbol{v}_j (\lambda_j / \lambda_i^{1/2}) = (\lambda_j / \lambda_i^{1/2}) = \lambda_i^{1/2}$ when $i = j$. Thus, $U' A V = \Sigma$ where U is the matrix with columns $\boldsymbol{u}_i$, V is the matrix with columns $\boldsymbol{v}_i$, and Σ is a matrix with all zeros except the diagonal elements which are equal to $\lambda_i^{1/2}$. Because each $\boldsymbol{u}_i$ is pairwise orthogonal and scaled to unit length, U is an orthogonal matrix. Similarly, V is an orthogonal matrix. Therefore, $A = U\Sigma V'$. □

Theorem 7 is often presented in the context of the *singular value decomposition* of a non-square matrix and can be found in (Bronson 1989, 125; Bernstein 2009, 301-304; Trefethen 1997, 29).

The relationship between the matrix A and the symmetric matrices $A'A$ and AA' can be more easily seen from the equations

$$A'A=(U\Sigma V')'(U\Sigma V')=V\Sigma U'U\Sigma V'=V\Sigma\Sigma V'=V\Sigma^2V',$$

$$AA'=(U\Sigma V')(U\Sigma V')'=U\Sigma V'V\Sigma U'=U\Sigma\Sigma U'=U\Sigma^2U'.$$

Because, *V* is orthonormal, $V'=V^{-1}$ and *V* transforms the columns of *A'A* into the diagonal matrix of eigenvalues and *V'* transforms the diagonal matrix of eigenvalues back into *A'A*. As *U* is also orthonormal, it has the same effect on *AA'*.

Now consider that each component of *A'A* is the dot product of a column of *A* with another column of *A*. Recall that the a dot product is the scalar component or magnitude of an orthogonal projection of one column onto another. However, we note that, if the columns are not of unit length, the value of the dot product will have some scaling factor from that of the actual length of the projection. But, such a scaling maintains a homogeneous relationship and, as such, the magnitude of the projection is zero when the columns are orthogonal. Consider a transformation that insures the dot product of the columns of *A* are all zero except when one column is multiplied by itself. This same transformation would diagonalize *A'A*. This transformation is exactly *V*. More specifically, *V* is a change of basis matrix that transforms the columns of *A* into a set of orthogonal vectors. Furthermore, as each column of *V* is of unit length, *V* only rotates the columns of *A*, but does not perform a dilation. As a result, *V* does not alter the data in the columns of *A,* but just rotates the columns to align with an orthogonal basis.

Notice that to diagonalize *A'A* is to maximize the elements on the diagonal. Thus, *V* maximizes the elements on the diagonal of *A'A* and these maximized values are the eigenvalues. Recall that the diagonal elements of $A'A$ consist of the the dot products of a column of A multiplied by itself which represents the magnitude or length of the column. Thus, the eigenvalues represent maximized lengths of a columns of A when rotated by V. Another way to say this is that V rotates A such that the columns lengths are maximized. The particular direction that maximizes the length of each column of A are exactly the eigenvectors of $A'A$, which are the columns of V. Thus, it can be said that the eigenvectors of $A'A$ are the directions that maximize the length or variance of the data in A. This shows the relationship between the eigenvectors of the matrix $A'A$ and the inherent structure of the data in A.

A similar interpretation can be made for AA' as the dot products of the rows of A. Thus, U' is a change of basis matrix that transforms the rows of A into an orthogonal set of vectors. However, in the context of a data matrix, there is less relevance to the orthogonalization of the observations. Thus, the decomposition of $A'A$ is the desired result.

The relationship between the eigenpairs of $A'A$, AA', and the non-square matrix A, is more specifically called the non-symmetrical eigenvalue problem and a more rigorous handling of the topic can be found in (Golub and Van Loan 1996, 308-318; Wilkinson 1965, chap. 6 and 7).

As an example, recall the drug data from Chapter 2. The charts on the left of Figure 3.1 shows the original data points expressed as a linear combination of the natural basis of columns of the data matrix. The chart on the right shows the data points after the change of basis performed by multiplication by the linear transformation V. The rotated data points in the chart on the right are expressed as linear combination of the columns of V. Notice that the structure of the data is maintained and the direction of greatest variance of the data now aligns horizontal axis of the chart.

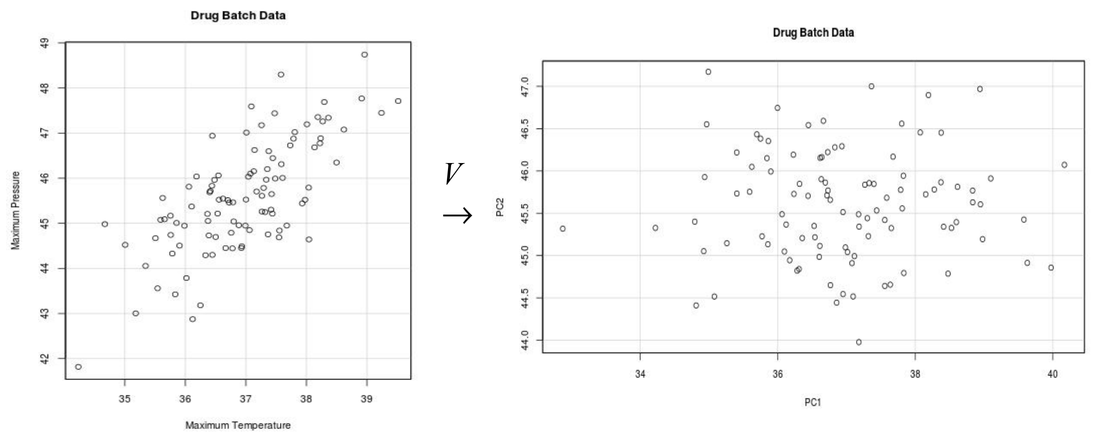

Figure 3.1: Original data (left) and rotated data (right).

Notice that, to make the equation $A = U \Sigma V'$ work for a $n \times p$ matrix A, the matrix Σ is required to be $n \times p$. Assume $p<n$, then the rank of A is, at most, p. The matrix $A'A$ is $p \times p$, which will require V to be $p \times p$. Similarly, AA' and U will be $n \times n$. Theorem 7 states that Σ will be all zeros except the diagonal elements that consists of the square roots of the eigenvalues of $A'A$ and AA'. This implies that there are more rows of Σ than there are eigenvalues and that the elements on the remaining rows are all zeros. These zeros effectively truncate the eigenvectors of AA' to be p dimensional. This reflects that the rank of $A'A$ is equal to the rank of AA' which is equal to the rank of A.

3.2 A Note Concerning the Covariance Matrix

The covariance matrix, also known as the variance-covariance matrix, is a matrix used to describe the linear relationships between structures in a matrix. For the purpose of data dimension reduction, the columns of the data matrix are the structures of interest. Thus, the covariance matrix is made up of variance and covariances of the columns of a data matrix.

More specifically, for a covariance matrix *A* of a $n \times p$ data matrix *D,* the diagonal value a_{ii} is the variance of the *i*th column of *D* and the off-diagonal value a_{ij}, where $i \neq j$, is the covariance between the *i*th column and the *j*th column of *D*. As the data matrix is interpreted as a sample from a population, the definition of a sample covariance defines the entries of *A* as

$$a_{ij} = Cov(\boldsymbol{d}_i, \boldsymbol{d}_j) = \frac{1}{n-1} \sum_{k=1}^{n} (d_{ki} - \bar{d}_i)(d_{kj} - \bar{d}_j), \tag{2}$$

where $\boldsymbol{d}_i$ is the *i*th column of *D*, $\bar{d}_i$ is the mean of $\boldsymbol{d}_i$, and d_{ij} is the entry at the *i*th rows and *j*th column of *D*. In the case where the data is preprocessed to be mean centered, then (2) becomes

$$a_{ij} = Cov(\boldsymbol{d}_i, \boldsymbol{d}_j) = \frac{1}{n-1} \sum_{k=1}^{n} (d_{ik})(d_{kj}) = \frac{1}{n-1} (d_i)' d_j .$$

As a result, the covariance matrix of the columns of the data matrix *D* can be expressed as

$$A = \frac{D' D}{n-1} .$$

This explicitly shows the matrix *D'D* has a homogeneous linear relationship to the covariance matrix of D when the data is mean centered. This convenient relationship maintains that, when $D'D$ is a diagonal matrix, the columns of D are uncorrelated and the diagonal values of $D'D$ are a scalar multiple of the column variances.

For the remainder of this study it is assumed that the data matrix has been preprocessed to be mean centered (Jolliffe 2002 , 31). It will be shown in Chapter 4 that the eigenvectors of $D'D$ have the desired result of being the uncorrelated directions that also maximize the variance of the data in the data matrix.

The theory of eigenpairs is vast and only a small portion has been presented. The proofs offered in this chapter have provided a basic foundation of mathematics and validity of eigenpairs as structures of data matrices. The relationship was established between the eigenpairs of a real symmetric matrix $D'D$ and the non-square data matrix D. The next chapter explores relevant methods for computing eigenpairs of data matrices.

CHAPTER 4

EIGENPAIR ALGORITHMS

The eigenpairs with the largest eigenvalues are of greatest interest when considering the reduction of the dimension of the data. The eigenpair with the largest eigenvalue is called the *largest eigenpair* and represents the direction of the data that contains the most information. Thus, it makes sense that finding the largest eigenpairs of a data matrix fits the criterion for data dimension reductions.

Some eigenpair algorithms compute eigenpairs sequentially and other algorithms compute all the eigenpairs at once. The algorithms that compute all the eigenpairs at once are considered to be more "advanced" algorithms as they are more efficient per eigenpair computed. However, if only a subset of the eigenpairs are desired, then these more "advanced" methods often turn out to be slower overall. For example, there is a data matrix with 4000 columns and only five eigenpairs are required, then it is quicker to compute just those first five eigenpairs with a less efficient sequential algorithm than to use a more "efficient" algorithm that must compute all 4000 eigenpairs. This relatively recent realization has come about due to the explosion in the size of data sets and has caused a renewed interest in more "basic" methods for finding eigenpairs, like the Power Method. However, it is notable that there is not a renewed interest in the Power Method by name, but rather a renewed interest in algorithms, such as the *Non-Linear Iterative Partial Least*

Squares (NIPALS) algorithm, which is easily shown to be equivalent to the Power Method.

4.1 The Power Method

The Power Method is the first algorithm widely used for computing eigenpairs of a matrix and remained a well used computational method, with modifications, until the 1970s (Jolliffe 1986, 8). Harold Hotelling's seminal paper, *Analysis of a Complex of Statistical Variables into Principal Components,* introduced Principal Component Analysis and the Power Method (Hotelling 1933, 417-416). The Power Method has been shown to be the basis of all eigenpair algorithms. As such, most texts use the Power Method to introduce algorithms for computing eigenpairs, but often discount it as old and inefficient.

The Power Method only computes the largest eigenpair of a square matrix. After which, a method called "deflation" can be used to remove the data in the direction of the first eigenvector. As such, the deflated matrix is the residual matrix after regressing the data on first eigenvector. The Power Method can be applied to the deflated matrix to find the second largest eigenpair. This procedure may be repeated until the desired number of eigenpairs have been computed.

The only requirement of the Power Method is a square matrix. Thus, for a $n \times p$ data matrix D, the $p \times p$ symmetric matrix $A=D'D$ must be explicitly computed prior to applying the Power Method. Proof of the convergence of the power method is presented next.

Theorem 9. Convergence of the Power Method

Let A be a $p \times p$ symmetric real valued matrix with k eigenvectors $\boldsymbol{v_i}$ and corresponding to eigenvalues λ_j where $\lambda_1 > \lambda_2 > \lambda_i$ for all $i=3, \ldots, k$ where $k<p$. Then $A^n\boldsymbol{x_0} = \boldsymbol{x_n} \rightarrow \boldsymbol{v_1}$ as $n \rightarrow \infty$ where $\boldsymbol{x_0}$ is any appropriately size vector with a non-zero component in the direction of $\boldsymbol{v_1}$.

Proof:

Any vector in k-dimensional space can be written as a liner combination of the basis formed by the eigenvectors of A. Let $\boldsymbol{x_0} = c_1\, \boldsymbol{v_1} + c_2\, \boldsymbol{v_2} + \ldots + c_k\, \boldsymbol{v_k}$ be an arbitrary starting vector such that $c_1 \neq 0$. Then we see that $A^n\boldsymbol{x_0} = c_1\, \lambda_1^n\, \boldsymbol{v_1} + c_2\, \lambda_2^n\, \boldsymbol{v_2} + \ldots + c_k\, \lambda_k^n\, \boldsymbol{v_k}$. Then, by dividing by λ_1^n, the equation becomes $A^n\boldsymbol{x_0}/\lambda_1^n = c_1\, \boldsymbol{v_1} + c_2\, (\lambda_2/\lambda_1)^n\, \boldsymbol{v_2} + \ldots$ $+ c_k\, (\lambda_k/\lambda_1)^n\, \boldsymbol{v_k}$. Therefore, as $n \rightarrow \infty$, $(\lambda_i/\lambda_1)^n \rightarrow 0$ for all $i = 2, \ldots, k$ and $A^n\boldsymbol{x_0}/\lambda_1^n \rightarrow c_1\, \boldsymbol{v_1}$. □

Additional information and variations of proofs can be found in (Wilkinson 1965, 570-572; Householder 1964, 187-190; Gentle 2007, 245-247; Mathews and Fink 1999, 568-573).

Selection of a starting vector is the first step of the the Power Method. It is a requirement that the starting vector $\boldsymbol{x_0}$ have a component in the direction of the largest eigenvector. However, at this point, there is no indication of the direction of the largest eigenvector. A common practice is to select a column of the A matrix to be used as $\boldsymbol{x_0}$. It has been suggested that the column of A with the greatest euclidean length should be selected as the starting vector. The idea is that a vector with the larger length may be closer to the direction of greatest variance. However, the computation required to find the column with the largest euclidean length can be more intensive than finding the first eigenvector. Thus, it is recommended that $\boldsymbol{x_0}$ be assigned the value of the first column of A and it will be as good an any choice.

Theoretically, the Power Method is described as $A^n\boldsymbol{x_0} \rightarrow \boldsymbol{v_1}$ while $n \rightarrow \infty$ where $\boldsymbol{v_1}$ is the eigenvector with the largest corresponding eigenvalue. This theoretical formulation requires the computation of the powers of the matrix A. Using the starting vector $\boldsymbol{x_0}$, the computations are characterized as follows,

$$\begin{aligned} A\,\boldsymbol{x_0} &= \boldsymbol{x_1} \\ A^2\,\boldsymbol{x_0} &= \boldsymbol{x_2} \\ A^3\,\boldsymbol{x_0} &= \boldsymbol{x_3} \\ &\vdots \\ A^n\,\boldsymbol{x_0} &= \boldsymbol{x_n} \end{aligned}$$

where $\boldsymbol{x_n} \rightarrow \boldsymbol{v_1}$ as $n \rightarrow \infty$. Convergence is achieved when $\|\boldsymbol{x}_n - \boldsymbol{x}_{n-1}\|_2 < \epsilon$ for some prescribed $\epsilon > 0$.

The matrix A^n acts as a linear transformation on the vector $\boldsymbol{x_0}$. The linear transformation A^n is modified at each iteration such that the columns of A^n become aligned with the direction of the first eigenvector. Thus, A^n approaches a rank one matrix as $n \rightarrow \infty$. It is interesting to note that the vector $\boldsymbol{x_0}$ is not really required in the computation as the simple iteration of powers of the matrix A will result in the columns of A^n aligning with the first eigenvector. The proof of the convergence of the Power Method follows the theoretical description of the powers of a matrix as a linear transformation on a starting vector $\boldsymbol{x_0}$.

Notice that the rate of convergence of the Power Method to the first eigenvector is explicitly seen as the rate in which the terms $c_2(\lambda_2/\lambda_1)^n\boldsymbol{v_2}, \ldots, c_p(\lambda_p/\lambda_i)^n\boldsymbol{v_p}$ vanish. Thus, the rate of convergence is directly related to the ratio between λ_1 and λ_2. The rate of convergence of the Power Method can also be visually represented by an ellipse that circumscribes the data in the direction $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ (Phatak and De Jong 1997, 317-318). When the ellipse is elongated, convergence is quick and when the ellipse looks more like a circle, convergence is slow.

A practical issue is the computation cost for computing powers of a matrix. An equivalent, but more efficient implementation of the Power Method, employs a successive updating of the starting vector until it converges to the first eigenvector. As such, the iterative computation of the Power Method changes from computing powers of a matrix to the multiplication of a matrix by a vector. The iterations are characterized as follows,

$$\begin{aligned} A\,\boldsymbol{x_0} &= \boldsymbol{x_1} \\ A\,\boldsymbol{x_1} &= \boldsymbol{x_2} \\ A\,\boldsymbol{x_2} &= \boldsymbol{x_3} \\ &\vdots \\ A\,\boldsymbol{x_{n-1}} &= \boldsymbol{x_n} \end{aligned}$$

where $\boldsymbol{x_n} \rightarrow \boldsymbol{v_1}$ as $n \rightarrow \infty$. The iteration of the $\boldsymbol{x_n}$ vector can be seen as successive steps toward the first eigenvector. Again, the methods stops when $\|\boldsymbol{x}_n - \boldsymbol{x}_{n-1}\|_2 < \epsilon$ for some $\epsilon > 0$. In this sense, the starting vector undergoes multiple transformation by a linear transformation until the transformation no longer has a directional effect on the vector.

The result is the solution to the eigenvalue problem $A\boldsymbol{x}_{n-1} = k\boldsymbol{x}_n$ where k is some constant. We would like k to be the eigenvalue, but it is not. The iterations of multiplication has caused a scaling problem.

The computation of the Power Method causes a monotonic exponential scaling of the vector $\boldsymbol{x}_n$ as it approaches the first eigenvector. This is shown as the constant c_1 in Theorem 9. Without a rescaling factor, c_1 will grow or shrink very quickly, causing the algorithm to exceed the storage capacity for a numeric value in most computers. Thus, a rescaling or normalization is required at each iteration. There are two commonly used methods for rescaling in the iterations of the Power Method. The methods differ in the choice a norm; the *2-norm* or the *∞-norm. T*he ∞-norm is defined as $\|\boldsymbol{x}_n\|_\infty = max\{|c_1|, |c_2|, \ldots, |c_k|\}$, where c_i are the components of $\boldsymbol{x}_n$ and k is the dimension of $\boldsymbol{x}_n$. As can be seen, the ∞-norm only requires a comparison of p values to find the largest component of $\boldsymbol{x}_n$ where as the 2-norm requires p squares, a summation, and a square root. As the rescaling is required at every iteration, the use of the ∞-norm is more efficient per iteration. Another benefit of using the ∞-norm is that that, as c_1 is rescaled to 1 at every iteration and, when the algorithm converges, is equal to the eigenvalue. However, in practice, using the *∞-norm* causes additional iterations to be required which degrades any benefits. Thus, the two scaling methods are largely equivalent.

An actual implementation of the Power Method using C++ is presented below. A matrix class is used to represent the matrices and vectors A, ***xn***, and ***new_xn***. The A matrix is a real symmetric matrix. The vector ***xn*** is initialized as the first column of A.

```
for (int ICounter=1;ICounter<=ItMax;ICounter++)
{
        new_xn=A*xn;
        lambda=xn.at(ColMax(xn,1),1);
        new_xn=new_xn*(1.0/lambda);
        if (VectorNorm(xn-new_xn,2)<Epsilon)
                break;
        xn=new_xn;
}
```

The convergence criteria of the Power Method is based on a preset value of *Epsilon* that represents an acceptable distance between $\boldsymbol{x}_{n-1}$ and $\boldsymbol{x}_n$. A common value for the *Epsilon* is 10*e*-10. In general, a smaller *Epsilon* is better, limited by the floating point precision of the computer. However, the processing power of the computer is often the practical lower bound. As computation of successive eigenpairs relies on the residual data from the last eigenpair computation, any error is cumulative. Thus, a large value of *Epsilon* can cause instability, especially for smaller eigenvalues.

After convergence to the first eigenvector is achieved, a method of deflation of the data matrix can be applied. Deflation removes the data in the direction of the first eigenvector. Consider a projection of the data onto the first eigenvector vector $\boldsymbol{v}_1$. Then, as $\|\boldsymbol{v}_1\|=1$, the scalar component of the projection would be $A\,\boldsymbol{v}_1$ and the direction component would just be $\boldsymbol{v}_1$. The result is the formula for the deflation as $A_{new} = A - A\boldsymbol{v}_1\boldsymbol{v}_1'$ (Wilkison 1965, 597). Then each column of A_{new} is a residual of a simple linear regression of the corresponding column of A and $\boldsymbol{v}_1$. A simpler form, using the definition of the eigenvalue problem, was suggested by Hotelling as $A_{new} = A - \lambda_1\boldsymbol{v}_1\boldsymbol{v}_1'$. Although computationally more efficient, additional error is introduced as λ_1 is an estimate.

Notice that there is error in the estimation of the eigenvector which results in an error in the deflated matrix. The result is additional error in the that computation of the second eigenpair. As the data matrix is deflated again to find the third eigenpair, additional error is accumulated in the deflated matrix. After many deflations, it is possible that the error may dominate the deflated matrix and may result in a false eigenpair. This is a problem common with all algorithms that compute eigenpairs.

Another problem that may occur with the Power Method is that it may not converge at all. Geometrically, consider the representation of an ellipse that circumscribes the data in the direction of first two eigenvectors. When the two eigenvalues are similar in value, the ellipse looks like a circle. In this case, it is difficult to identify the point on the ellipse that intersects the major axis. Algorithmically, the Power Method has the same problem. In the case when the two largest eigenvalues are equal, the Power Method fails to converge. Although this failure mode exits, it is very rare with real data that any two eigenvalues are exactly equal. The case is usually that the first few eigenvalues are significantly different and the last eigenvalues may be close in value (Jackson 1991, 32). This tends to work well in the context of dimension reduction as the largest eigenpairs are of importance. Notice how the convergence of the Power Method is related to the geometric properties of the data represented by the ratio of the eigenvalues.

A second failure mode of the Power Methods is a poor starting vector. If the starting vector does not have a component in the direction of the eigenvector being sought, then the Power Method fails to converge. This is the case when the starting vector $\boldsymbol{x}_0$ is orthogonal to the eigenvector being computed. Although the Power Method may fail in this situation, it is not a common, even with small dimensional data. In the context of data matrices with a large number of columns, the possibility of selecting a vector orthogonal to another is very small and is often not considered a practical failure mode. However, an upper bound to the number of iterations should be part of any implementation of the Power Method.

Variations have been developed over the years to extend or enhance the Power Method. Two such variations are the Inverse Power Method and the Shifted Power Method. The Inverse Power method is the Power Method applied to A^{-1} instead of A. the result is that the smallest eigenpair is computed instead of the largest. The Inverse Power Method is often used in numerical solutions to partial differential equations where the smaller eigenpairs are desired. The Shifted Power Method performs a shift to the A matrix in order to increase the speed of convergence. The intent is to shift the data to increase the ratio between the first two eigenvalues. Although computing the shift takes additional work, the Shifted Power Method has been shown to increase the speed of convergence from that of the standard Power Method. More on the Inverse Power Method and the Shifted Power Method can be found in (Wilkinson 1965, 572).

The Power Method can be use to compute eigenpairs for a real symmetric matrix. With the use of deflation, the largest eigenpairs are computed one at a time. This successive computation of the eigenpairs avoids unnecessary computations when only the first few eigenpairs of a matrix are desired. Algorithms that only compute a subset of the eigenpairs have gained relevance with the growing size of databases. Another such method for computing eigenpairs is the Nonlinear Iterative Partial Least Squares algorithm.

4.2 Nonlinear Iterative Partial Least Squares Algorithm

The Nonlinear Iterative Partial Least Squares (NIPALS) algorithm is credited to Herman Wold (Wold 1966, 391-420). The NIPALS algorithm computes the eigenpair with the largest eigenvalue one at a time, using deflation to compute additional eigenpairs. It will be shown that the NIPALS algorithm is equivalent to the Power Method (Lorber, Wangen, and Kowalski 1987, 19-31). The NIPALS algorithm differs from the Power Method in convenience of application and as a more explicit computation of an important structure called a *principal component*. The principal component is of particular interest with regards to data dimension reduction.

The NIPALS algorithm begins with an arbitrary starting vector, usually a column directly from the data matrix is selected. The starting vector undergoes two transformations to complete a single iteration of the NIPALS algorithm. At the end of each iteration, convergence is checked. The algorithm is characterized as follows, where D is any $n \times p$

data matrix and $\boldsymbol{t}_0$ is initialized as the starting vector.

$$\begin{aligned}
&\boldsymbol{p}_1 = D'\,\boldsymbol{t}_0\\
&\boldsymbol{t}_1 = D\,\boldsymbol{p}_1\\
&\boldsymbol{p}_2 = D'\,\boldsymbol{t}_1\\
&\boldsymbol{t}_2 = D\,\boldsymbol{p}_2\\
&\quad\vdots\\
&\boldsymbol{p}_n = D'\,\boldsymbol{t}_{n-1}\\
&\boldsymbol{t}_n = D\,\boldsymbol{p}_n
\end{aligned}$$

where $\boldsymbol{p}_n \rightarrow \boldsymbol{v}_1$ as $n \rightarrow \infty$ for the eigenvector $\boldsymbol{v}_1$ corresponding to the largest eigenvalue. Notice that the two step iteration accommodates the the non-square data matrix *D,* thus making it appear that the NIPALS is more capable than the Power Method.

The starting vector $\boldsymbol{t}_0$ is chosen as the first column of the data matrix. The algorithm starts by multiplying $\boldsymbol{t}_0$ by the transpose of the data matrix *D*, which transforms $\boldsymbol{t}_0$ into the $\boldsymbol{p}_1$ vector that exists in the row space of *D*. Then, the $\boldsymbol{p}_1$ vector is pre-multiplied by *D* to obtain a new $\boldsymbol{t}$ vector in the column space of of *D*. The iteration are often characterized as projections onto the row space and column space of *D* (Geladi and Kowalski, 1986, 5-8; Varmuza and Filzmoser 2009, 87-88). However, the NIPALS algorithm is better explained by its equivalence to the Power method. The relationship between the NIPALS algorithm and the Power Method is simply the condensation of the two steps of NIPALS to the single step of the Power Method. Consider the last step of one iteration and the first step of the next iteration of the NIPALS algorithm,

$$\begin{aligned}
&\boldsymbol{t}_n = D\,\boldsymbol{p}_{n-1}\\
&\boldsymbol{p}_n = D'\,\boldsymbol{t}_n
\end{aligned} \tag{2}$$

can be condensed to the single line

$$\boldsymbol{p}_n = D'D\,\boldsymbol{p}_{n-1}.$$

Which is exactly the Power Method. Thus, Theorem 9 can be considered a proof of the convergence of the NIAPLS algorithm and all conclusion concerning the Power Method also apply to the NIPALS algorithm.

To compute the corresponding eigenvalue of an eigenvector found using the NIPALS algorithm, the Rayleigh Quotient may be used. The Rayleigh quotient is easily derived from the definition of an eigenvalue as follows:

$$A\,\boldsymbol{p}_n = \lambda_1\,\boldsymbol{p}_n$$
$$\boldsymbol{p}_n'\,A\,\boldsymbol{p}_n = \boldsymbol{p}_n'\,\lambda_1\,\boldsymbol{p}_n$$
$$\frac{\boldsymbol{p}_n'\,A\,\boldsymbol{p}_n}{\boldsymbol{p}_n'\,\boldsymbol{p}_n} = \lambda_1$$

for *n* sufficiently large where λ_1 is the estimate of the eigenvalue associated with the estimated eigenvector $\boldsymbol{p}_n$. Thus, as $\boldsymbol{p}_n'\boldsymbol{p}_n=1$ and $A=D'D$, the Rayleigh Quotient reduces to

$$\boldsymbol{p}_n'\,A\,\boldsymbol{p}_n = (\boldsymbol{p}_n'\,D')\,D\,\boldsymbol{p}_n = \boldsymbol{t}_n'\,\boldsymbol{t}_n = \lambda_1\,.$$

The vector $\boldsymbol{t}_n$ is referred to as the *principal component* corresponding to the eigenvector $\boldsymbol{p}_n$ and is defines as $\boldsymbol{t}_n = D\boldsymbol{p}_n$. The principal component represents the data in the direction of the eigenvector. That is, if we consider the context of dimension reduction of a data matrix, the $\boldsymbol{t}_n$ vector is the magnitude or scalar component of the data projected onto the direction of the greatest variance. In that sense, the $\boldsymbol{t}_n$ vector can be thought of as a vector that captures the maximum variation or information in the data matrix. The principal component corresponding to the largest eigenpair is a one dimensional representation of the data matrix that retains the most information. Additional principal components can be used to retain information from data remaining in the data matrix after deflation. This is an important concept in the reduction of the dimension of data with eigenpairs.

The principal component may be considered as the scalar component of a projection onto the first eigenvector and the first eigenvector is the direction of the projection. Together, they are a decomposition of a single dimension of the data in the data matrix. As the eigenvectors form a basis of the row space of the data matrix, the columns may be represented as a linear combination of eigenvectors. The coefficients of the linear combination are the magnitudes of the projection in the direction of each eigenvector, which are exactly the corresponding principal components. Thus, for a $n \times p$ data matrix of full rank, let $r = p < n$ and where r is the number of eigenpairs computed. The relationship between the eigenvectors, principal components, and the data matrix is

$$D=TP' \tag{3}$$

where D is the data matrix, T consists of columns $\boldsymbol{t}_1 \ldots \boldsymbol{t}_r$ of principal components and P consists of columns $\boldsymbol{p}_1 \ldots \boldsymbol{p}_r$ of eigenvectors corresponding to the respective principal components of T. Then, because $T=DP$, it can be said that P is the change of basis matrix from D to T. That is, the eigenvectors are used to rotate the data matrix to a set of orthogonal columns ordered by the maximum variance inherent to the data. The result of the change of basis is the T matrix.

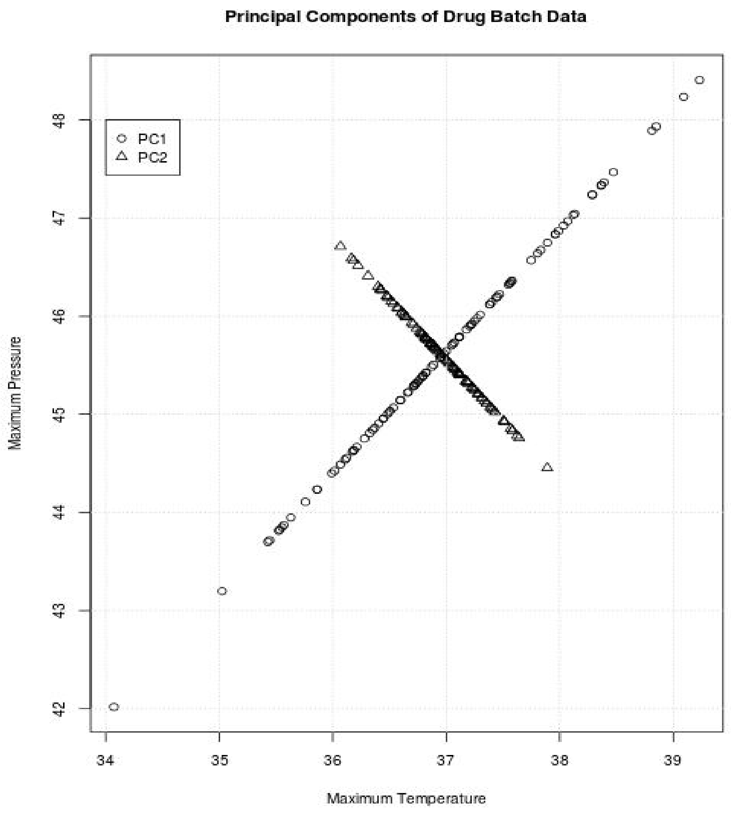

Figure 4.1: Plot of the principal components of the drug batch data.

As an example, consider the simplified drug data from Chapter 2. Figure 4.1 shows the principal components of the data matrix for the same drug batch data. Each original data point can be expressed as the sum of two points, one from each principal component. As the eigenvectors are an orthogonal basis, the columns of *T* are uncorrelated. Thus, the corresponding eigenvalues can also be interpreted as an indication of the strength of the dependency relationship among of the columns of *D* in the direction of the respective eigenvector. That is, if there is a strong dependency relationship among the columns of the data matrix, then many of the columns have a significant correlation in a particular direction and that direction is an eigenvector. As a result of the dependency relationships among the columns of the data matrix, there will also be small eigenvalues that represent the less uncorrelated information in the data. This indicates that most of the variation of the data is captured in the first r principal components where $r < p$. Thus, all p of the columns of T and P may not be required to retain most of the information in the data. Then the relationship describes by (3) becomes

$$D=T_rP_r'+E, \tag{4}$$

where T_r and P_r only contain r corresponding columns of T and P respectively. As the columns of T and P are kept in the order in which they were computed, the first r columns are the best choice to retain the most information. Thus, the dominate linear relationships of the data are represented as the $n \times r$ matrix T_rP_r'. The amount of dependency among the columns of the data matrix determine the amount of information that can be represented in the first r principal components.

As each eigenvalue represents the variability of the data in the direction of the corresponding eigenvector, the sum of the eigenvalues represents the total variability of the data. Thus, to represent the largest amount of variability with the smallest number of principal components, the principal components with the largest corresponding eigenvalues are retained.

The E term in (4) represents the information not included in the first r principal components. Also notice that the existence of multicollinearity will result in a high amount of information being represented by a small number of principal component. Thus, the best value of r is determined by the level of the dependency relationships between the columns of the data matrix.

As the eigenvalues indicates the level of the dependency relationship of the information in the corresponding principal component, it makes sense to analyze the eigenvalues to determine an appropriate value of *r*. Many algorithms have been developed to optimize the number of principal components and extensive information can be found on these formulas in (Jackson 1991, 41-51; Jolliffe 2002, 111-147; Khattree and Naik 2000, 40-91). However, in practice, there is often an obvious drop off point between the large and small eigenvalues. To see this construct a plot of the eigenvalue versus *r;* called a SCREE plot. An example of a SCREE plot is presented in Figure 4.2.

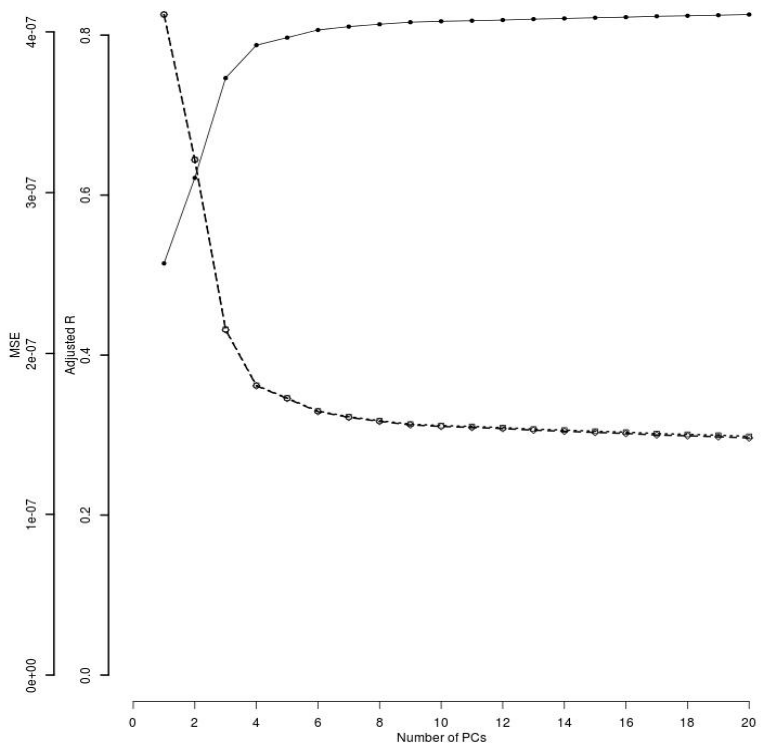

Figure 4.2: An example of a SCREE plot

The notion that the largest eigenvector represent the direction of greatest variance has been asserted many times in this study. This relationship will now be made explicit with Theorem 10 and Corollary 10.1. First we show that the Raleigh Quotient of a matrix is maximized by its first eigenvector.

Theorem 10: The Rayleigh Quotient obtains its maximum value for a given symmetric matrix with its largest eigenvector.

Proof:

Let D be a $n \times p$ data matrix and $A=D'D$ is a $p \times p$ real valued symmetric matrix. Then there exists real eigenvector $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p$ and real eigenvalue $\lambda_1^2 > \lambda_2^2 > \ldots > \lambda_p^2$. By theorem 8, $A = V\Sigma^2 V'$ where the columns of V are $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p$ and Σ^2 is a diagonal matrix with diagonal entries $\lambda_1^2 > \lambda_2^2 > \ldots > \lambda_p^2$. Define $\boldsymbol{y} = V'\boldsymbol{x}$ for any $\boldsymbol{x} \in \mathbb{R}^p : \boldsymbol{x}'\boldsymbol{x} = 1$. Then, the Rayleigh Quotient is

$$\frac{\boldsymbol{x}'A\boldsymbol{x}}{\boldsymbol{x}'\boldsymbol{x}} = \frac{\boldsymbol{x}'V\Sigma^2V'\boldsymbol{x}}{\boldsymbol{x}'VV'\boldsymbol{x}} = \frac{\boldsymbol{y}\Sigma^2\boldsymbol{y}'}{\boldsymbol{y}'\boldsymbol{y}} = \frac{\sum_{i=1}^{p}\lambda_i^2 y_i^2}{\sum_{i=1}^{p} y_i^2} \le \lambda_1^2 \frac{\sum_{i=1}^{p} y_i^2}{\sum_{i=1}^{p} y_i^2} = \lambda_1^2 .$$

Consider $\boldsymbol{x} = \boldsymbol{v}_1$, then $\boldsymbol{y} = V'\boldsymbol{v}_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ and $\frac{\boldsymbol{v}_1'A\boldsymbol{v}_1}{\boldsymbol{v}_1'\boldsymbol{v}_1} = \boldsymbol{y}'\Sigma^2\boldsymbol{y} = \lambda_1^2 .$

Therefore, $\frac{\boldsymbol{x}'A\boldsymbol{x}}{\boldsymbol{x}'\boldsymbol{x}} \le \frac{\boldsymbol{v}_1'A\boldsymbol{v}_1}{\boldsymbol{v}_1'\boldsymbol{v}_1} \quad \forall \boldsymbol{x} \in \mathbb{R}^p : \boldsymbol{x}'\boldsymbol{x} = 1 .$ □

A full discussion of Theorem 10 can be found in (Johnson and Wichern 2007, 80-81).

Corollary 10.1: The variance of a principal component is maximized.

Proof:

Let *D* be a $n \times p$ data matrix and *A=D'D* is a $p \times p$ real valued symmetric matrix. Then there exists real eigenvector $\boldsymbol{v}_1$ with a corresponding real eigenvalue λ_1^2, such that $\lambda_1^2 > \lambda_i^2$ for $i = 2, \ldots, p$. Let $\boldsymbol{t}_1 = D\boldsymbol{v}_1$ be the first principal component of the data matrix *D*. Thus,

$Var(\boldsymbol{t}_1) = \boldsymbol{t}_1{}'\boldsymbol{t}_1 = \boldsymbol{v}_1{}'D'D\boldsymbol{v}_1 = \frac{\boldsymbol{v}_1{}'D'D\boldsymbol{v}_1}{\boldsymbol{v}_1{}'\boldsymbol{v}_1} = \frac{\boldsymbol{v}_1{}'A\boldsymbol{v}_1}{\boldsymbol{v}_1{}'\boldsymbol{v}_1}$ and, from the results of Theorem 10, $\frac{\boldsymbol{x}'A\boldsymbol{x}}{\boldsymbol{x}'\boldsymbol{x}} \le \frac{\boldsymbol{v}_1{}'A\boldsymbol{v}_1}{\boldsymbol{v}_1{}'\boldsymbol{v}_1} \quad \forall \boldsymbol{x} \in \mathbb{R}^p : \boldsymbol{x}'\boldsymbol{x} = 1.$ Therefore, the variance of a principal component is maximized.□

Corollary 10.1 proves the assertion that the most variation of the data matrix is captured in each principal component. As subsequent principal components are computed from the deflated data matrix, each principal component is uncorrelated and contains the maximum amount of variation. Also notice how that the proofs explicitly show the relation between the variance of a principal component of the data matrix D and the eigenvalue of the corresponding eigenpair of the symmetric matrix *D'D*.

The use of principal components to represent data is the basic concept of Principal Component Analysis as introduced by Hotelling in 1933. The idea is that there is a small number of latent or soft variables that describe the underlying relations of the explicit data in the data matrix. To that extent, there is more than dimension reduction occurring, rather a discovery process of the true variables of the phenomena being observed. Using pharmaceutical production as an example, there may be many sensors on a reactor, but there may be only a few underlying semi-observable chemical processes occurring that result in correlated perturbations in the sensor readings. Thus a very large data matrix may be represented by a small number of principal components with very little loss of information. Some excellent references for information on Principal Component Analysis are (Jackson 1991; Jolliffe 2002; Khattree and Naik 2000, 40-91).

The NIPALS algorithm is a conveniently structured implementation of the Power Method, but offers no more functionality than the Power Method. The NIPALS algorithm has been widely used in *chemometrics,* the statistical monitoring of chemical processes (Geladi 1988, 231-246; Roffel and Betlem 2006, 305-316; Qin 2003, 480–502; Lee and Vanrolleghem 2002, 489-497; Jackson and Mudholkar 1979, 341-349). The NIPALS algorithm is presented here for two purposes. The first is a convenient presentation of the concept of a principal component and its use in the dimension reduction of data matrices. The second purpose is to give a historical perspective of the method of Partial Least Squares that will be presented in the next chapter. The NIPALS algorithm and the Partial Least Squares algorithms are both credited to Herman Wold, where the NIPALS algorithm preceded the PLS algorithms. After a discussion on principal component regression, it will be shown in the next chapter that the NIPALS and PLS algorithms (and transitively, the Power Method) are very closely related.

CHAPTER 5

REGRESSION METHODS AND DIMENSION REDUCTION OF DATA

One goal of this study is to present methods for extracting meaningful information from large data sets. It has been shown that eigenpairs can be used to construct principal components that are representations of the dominate linear relationships of the data. Chapter 5 takes the next step from dimension reduction to inferential information extraction in the form of regression.

5.1 Principal Component Regression

The method of Principal Component Regression (PCR) is a simple extension of the concept of principal components. The basic idea is to use the principle components instead of the data matrix in a regression model.

Consider the data matrix D and the the T_r and P_r matrices as in equation (4). The r subscript will be dropped and the T and P matrices will be assumed to have only r columns where r represents the number of principal components retained. The PCR algorithm uses the orthogonal columns of T as regressors on some response variable $\boldsymbol{y}$ represented by a $n \times 1$ vector. PCR uses the familiar ordinary least squares formulas in (5) to compute the coefficients of the regression, $\hat{\boldsymbol{\beta}}$ and fitted values, $\hat{\boldsymbol{y}}$.

$$\hat{\boldsymbol{\beta}} = (T'T)^{-1} T \boldsymbol{y} \qquad (5)$$
$$\hat{\boldsymbol{y}} = T\hat{\boldsymbol{\beta}}.$$

The prediction of the response requires that a new $1 \times p$ observation vector $\boldsymbol{x}_{\boldsymbol{new}}$ be transformed by P to obtain the $1 \times r$ vector $\boldsymbol{t}_{\boldsymbol{new}}$. Then the prediction is calculated as

$$\hat{y}_{new} = \boldsymbol{t}_{\boldsymbol{new}} \hat{\boldsymbol{\beta}}.$$

An obvious benefit of PCR is in the reduction of the dimension of the data. Computing the coefficients of the regression for a data matrix with thousands of columns can be difficult even with the fastest computers. However, computing the coefficients of the regression from a T matrix with a significantly reduced number of columns can make the analysis feasible to compute.

More Importantly, consider an *under-determined* matrix as having less observations (rows) than regressors (columns). An under-determined data matrix D does not have a unique solution to the equation $D\boldsymbol{x}=\boldsymbol{b}$. As a result, a Multiple Least Squares (MLS) regression with an under-determined data matrix will fail because the solution of the formula for coefficients of the regression is also not unique. However, for PCR, when the number of principal components is selected to be less than the number of observations, then the regression becomes possible.

Notice that, with a MLS regression model, every regressor is explicitly correlated to the response by an estimated coefficient of regression. When the columns of the data matrix have some level of dependency among them, the coefficients of the regression reflect this relationship, usually in a canceling effect of positive a negative coefficients. If the MLS regression model is used to make a prediction with new data and the new data does not reflect same dependency relationships between the regressors, then an instability in the prediction can occur. However, as the columns of T are orthogonal, there is no dependent relation among the columns and, as a result, this source of instability is avoided with PCR (Montgomery, Peck, and Vining 2006, 357). In this sense, the calculation of principal components can be thought of as a preprocessing step that uncorrelates the columns of the data matrix. This is a significant benefit of PCR over MLS regression.

Notice, if the columns of the $n \times p$ data matrix are uncorrelated and if $r = p < n$, then PCR is identical to MLS regression. In this case, reducing the dimension of the data by retaining a smaller number of principal components is the same as removing a regressor in a MLS regression. However, when the columns of the data matrix are not orthogonal, then a PCR model with less principal components than columns of the data matrix can actually improve the stability of the the model. The idea is that a subset of principal components retained may better represent the inherent structure or correlations of the data. The intent is that the information discarded in the smaller principal components represent information that is uncorrelated with the rest of the columns. Thus, an artificial correlation to the response from this data may contribute to instability of an estimate. Notice that removing smaller principal components from the model is dissimilar to removing a whole regressor from the data matrix as small principal components represents less information.

Notice that the decomposition of the data matrix into principal components in (4) does not involve the ***y*** vector at all. Thus, the computation of the principal components only reflect the relationships among the columns of D and does not consider strength of any correlation to the response vector ***y***. As a result, the procedure of choosing principal components for PCR, may result is a set of vectors that do not represent a strong correlation to the response. This is a fundamental flaw of PCR.

Regardless of this flaw, PCR can be used as a simple regression method to eliminate the instability of multicollinearity. However, the method may be improved if the algorithm could consider the dominate correlations between the the data matrix and the response variable. This is the idea of the method of Partial Least Squares.

5.2 Partial Least Squares

Herman Wold is credited with the development of the Partial Least Squares (PLS) algorithm. The algorithm is also referred to as Projection to Latent Structures. Although Wold primarily worked in economics, the method of PLS has found wide usage in the areas of chemistry (and chemical engineering), education, psychology, management science, political science, and environmental science (Geladi 1988, 231-246). Much of the focus on PLS in chemical engineering has come from Svante Wold, son of Herman Wold.

5.2.1 The General PLS Algorithm

The popularity of PLS is due to its combined utility of eliminating the effect of multicollinearity, dimension reduction, and the maximization of the correlation between the regressors and the response. The elimination of the effects of multicollinearity and dimension reduction are very similar to PCR. The maximization of the correlation to the response is what sets PLS apart form PCR. The PLS algorithm considers both the data matrix and the matrix of response variables to construct the principal components. As a result, a stable regression model can often be obtained with a small number of principal components. Consider this example of how the dimension reduction with PLS enables the regression of time series data.

The production of pharmaceutical drugs occurs in a reactor as a batch. A reactor may have 30 sensors and each sensor may be read thousands of times per batch. However, each sensor may be read at a different interval and each batch may be a different duration. Thus, the data must be interpolated to a common interval and preprocessed to a uniform length. Suppose the preprocessing reduces the number of sensor readings to a uniform length of 100 readings per batch. Then, the data for each batch is unfolded into a row vector where each time slice of sensor readings is placed end-on-end. As a batch may take days to finish, the number of batches available for building a model may only be 50 or less. Each batch to be included in the training of the model is then stacked to form a matrix. This rearrangement of the data is called *batch-wise unfolding*. As a result, the data matrix has 3000 columns and 50 rows. The time-series nature of the data matrix means that there are 100 column of the same sensor readings at different points of time during a batch. These multiple reading cause multicollinearity and, as a result, there is a small inherent dimension of the data that represents the best correlation to the response. In many cases, only three or four principal components are required to model a pharmaceutical batch. Also notice that the data matrix is under-determined and, therefore, a MLS regression model would not be possible. Thus, PLS is well suited for the analysis of pharmaceutical batch production. A more complete description of the use of PLS for pharmaceutical batch monitoring can be found in (Reiss, Wojsznis, and Wojewodka 2009, 75–82).

For an deeper understanding of how the PLS algorithm works, a comparison will be made with the Power Method. However, it is first noted that the general PLS algorithm has the capability to consider multiple response variables. Define l to be the number of response variables and the matrix Y to be an $n \times l$ matrix of response data. Although a model with multiple response variables has value for some applications, the estimation of a response variable is less accurate than a PLS model with a single response variables. However, the general PLS algorithm will be presented to show the explicit connection with the Power Method. A special case where l=1 will be considered later in this chapter.

Let D be an $n \times p$ data matrix and $\boldsymbol{p}_1$ be a starting vector with some component in the direction of the first eigenvector of $D'D$. It has been shown that the Power Method iteration of

$$\boldsymbol{p}_n = D'D\boldsymbol{p}_{n-1} \tag{6}$$

will converge to the first eigenvector of $D'D$. That is, $\boldsymbol{p}_n = \boldsymbol{p}$ as $n \rightarrow \infty$ and the solution to the eigenvalue problem is $D'D\boldsymbol{p} = \lambda\boldsymbol{p}$. Notice that the Power Method iterations, as in (6), can also be written as $\boldsymbol{p}_n$=Cov(D, $D\boldsymbol{p}_{n-1}$) where, as the vector $\boldsymbol{p}_n$ converges to the first eigenvector $\boldsymbol{p}$, the covariance between D and $D\boldsymbol{p}$ is maximized. This maximization is subject to the constraint that $\boldsymbol{p'p}$=1. This is just another way of saying that $\boldsymbol{p}$ represents the vector in the direction of the greatest covariance of D and $D\boldsymbol{p}$ which is also the direction of the greatest variance of $D'D\boldsymbol{p}$ of which was proven in Chapter 4.

The general PLS algorithm is based on the Power Method (and NIPALS). However, instead of working with one matrix, there are two matrices, the data matrix X and the matrix of response variables Y. Consider the matrix to be decomposed is $Y'X$. Then the Power Method iterates

$$\boldsymbol{w}_n=(Y'X)'(Y'X)\boldsymbol{w}_{n\text{-}1}\,, \tag{7}$$

until the vectors $\boldsymbol{w}_n$ converges. Notice that equation (7) can also be written as

$$w_n=(Y'X)'(Y'X)w_{n-1}=cov(Y'X,Y'X\,w_{n-1})=cov\left[cov(Y,X),cov(Y,X\,w_{n-1})\right].$$

The result is the vector $\boldsymbol{w}$ that maximizes the covariance of the matrices $Y'X$ and $Y'X\boldsymbol{w}$, subject to the constraint $\boldsymbol{w'w}$=1. As $Y'X$ is also a covariance matrix, $\boldsymbol{w}$ is the direction that maximizes the correlation in X that is most correlated to Y.

The iteration of the equation (7) is the core of the general PLS algorithm and this establishes the mathematical equivalence of the general PLS algorithm to the Power Method. However, the algorithm includes more steps as there are important structures that must be computed. The vectors involved in the general PLS algorithm are summarized below.

Table 5.1: Vectors of the general PLS algorithm

$\boldsymbol{t}$	$n \times 1$	*X Scores*
$\boldsymbol{p}$	$1 \times r$	*X Loadings*
$\boldsymbol{u}$	$n \times 1$	*Y Scores*
$\boldsymbol{q}$	$l \times 1$	*Y loading(coefficient of the regression)*
$\boldsymbol{w}$	$p \times 1$	*X loadings with maximized correlation to Y*

The PLS algorithm begins by initializing the ***u*** vector as a column of the *Y* matrix and the ***t*** vector as a column of the *X* matrix, and defining $\epsilon = 10e-10$ or some suitable small value. The General PLS algorithm is executed as follows.

$$\boldsymbol{w} = \frac{X'\boldsymbol{u}}{\|\boldsymbol{w}\|} \tag{8}$$

$$\boldsymbol{t}_{new} = X\boldsymbol{w}$$

$$\boldsymbol{q} = \frac{Y'\boldsymbol{t}_{new}}{\boldsymbol{t}_{new}'\boldsymbol{t}_{new}}$$

$$\boldsymbol{u} = \frac{Y\boldsymbol{q}}{\boldsymbol{q}'\boldsymbol{q}}$$

if $\|\boldsymbol{t} - \boldsymbol{t}_{new}\| > \epsilon$ then $\boldsymbol{t} = \boldsymbol{t}_{new}$ and repeat steps above.

$$p = \frac{X'\boldsymbol{t}}{\boldsymbol{t}'\boldsymbol{t}}$$

Careful examination of algorithm (8) shows, in addition to (7), there are three more similar vector iterations occurring; $\boldsymbol{t}_n=(X'X)'YY'\boldsymbol{t}_{n-1}$, $\mathbf{q}_n=(X'Y)'X'Y\boldsymbol{q}_{n-1}$, and $\boldsymbol{u}_n=(Y'Y)'XX'\boldsymbol{u}_{n-1}$. All of these computation are rolled into the algorithm to produce meaningful structures for the prediction of the response. These structure are now described.

The vectors ***p*** and ***t*** of the algorithm (8) are similar to the ***p*** and ***t*** vectors from the NIPALS algorithm (2) as they represent the eigenvectors and principal components of the *X* matrix. However, notice, as *Y* appears in the computation of ***t*** and, thus, in ***p***, the vectors are not the same as the ***t*** and ***p*** vector in the NIPALS algorithm. The ***t*** and ***p*** vector are often referred to as the *score* and *loading* vectors for the *X* matrix and the ***u*** and ***q*** vectors are referred to as the the *scores* and *loading* vectors for the *Y* matrix. Again notice that *X* is required in the computation of the ***u*** and ***q*** vectors. Thus, each set of principal components take into consideration both the *X* and the *Y* matrices. The ***w*** vector is of primary interest in that it represents the direction of the greatest correlation of the covariance matrices of *X* and *Y*.

After (8) successfully converges, a method of deflation is used to remove the information from *X* and *Y* that is explained by this first component. The deflation for PLS is performed by

$$\begin{aligned} X_{new} &= X - \boldsymbol{tp} \\ Y_{new} &= Y - \boldsymbol{tq}\,. \end{aligned} \tag{9}$$

Notice that the deflation term for *X* is similar to (3) and is the matrix of residuals for the principal component. Notice that ***tq*** is the fitted values of the regression of the principal component and is the proper deflation of the *Y* matrix. Additional components are calculated from the residual matrices X_{new} and Y_{new}.

The ***t***, ***p***, ***u***, ***q*** and ***w*** vectors from each component are stored as columns in the *T, P, U, Q,* and *W* matrices, respectively. After the desired number of principal components have been found, the following are the dimensions of the matrices computed by the PLS algorithm.

Table 5.2: Matrices of the general PLS algorithm.

T	$n \times r$	*X matrix Scores*
P	$p \times r$	*X matrix Loadings*
U	$n \times r$	*Y matrix Scores*
Q	$l \times r$	*Y matrix loading (coefficients of the regression)*
W	$p \times r$	*X loadings with maximized correlation to Y*

Below is a C++ implementation of the general PLS algorithm.

```
t=X.Col(1);

u=Y.Col(1);

for (ICounter=1;ICounter<ItMax;ICounter++)

{

        w=(X.Transpose()*u);

        w=w*(1.0/VectorNorm(w,2));

        t_new=(X*w);

        q=(Y.Transpose()*t_new)*(1.0/(t_new.Transpose()*t_new));

        u=(Y*q)*(1.0/(q.Transpose()*q));

        if (VectorNorm(t-t_new,2)<epsilon)

                break;

        t=t_new;

}

p=(X.Transpose()*t)*(1.0/(t.Transpose()*t).at(1,1));
```

The fitted values of the PLS model are easily calculated by $\hat{Y} = TQ'$. Estimating the response for new data with PLS requires the new observation be projected onto the loadings to obtain the principal component vector $\boldsymbol{t_{new}}$. However, the desired loadings vectors $\boldsymbol{w_i}$, that make up the columns of the W matrix, are not orthogonal. There is also a need to scale the $\boldsymbol{t_{new}}$ vector by $\boldsymbol{w_i'w_i}$. This oblique projection is required to compute the principal components for PLS (Phatak and De Jong 1997, 311–338) and is computed as follows for each principal component *i* where the *i*th component of $\boldsymbol{t_{new}}$ is noted as t_{new_i}

$$t_{new_i} = \frac{X\,\boldsymbol{w_i}}{\boldsymbol{w_i}'\,\boldsymbol{w_i}}. \tag{10}$$

After each component is computed, the deflation

$$X = X_{old} - \boldsymbol{t}_{\boldsymbol{new}} P_i$$

must occur to remove the any overlap (Jackson 1991, 286).

An example of an implementation of C++ code to perform the oblique projection is presented below.

```
for (int RCounter=1;RCounter<=r;RCounter++)
{
        td=W.Col(RCounter).Transpose()*W.Col(RCounter)
        t_new.Assign(RCounter,1,(X*W.Col(RCounter))*(1.0/td));
        X=X-(t_new*P.Col(RCounter).Transpose());
}
```

Once the the $\boldsymbol{t}_{new}$ vector has been properly computed, the estimated response is obtained as

$$\hat{y} = t_{new} Q'.$$

5.2.2 The PLS1 Algorithm

What has been described as the general PLS algorithm is commonly referred to as the PLS2 algorithm and its defining feature is that it can accommodate more than one response variable. Although the functionality of multiple response variables is useful in some applications, it can have a negative effect on a PLS model used to estimate response variables. That is, unless the response variables have a strong linear relationship, the prediction of the PLS2 model will be less accurate than a separate model for each response variable (Hasegawa 2006). The reason is that each coefficients of the regression can only reflect one linear relationship. Thus, if the two response variables do not have a linear relationship, then the coefficients of the regression can only estimate the best linear relationship to represent both response variables. The result is a compromise in the prediction of both response variables.

The PLS1 algorithm is an optimized version of the PLS2 algorithm for the special case where l=1. The PLS1 algorithm is significantly simplified and considerably less expensive to compute. The simplification allows for the $\boldsymbol{w}$ vector to be computed directly as $\boldsymbol{w} = \mathrm{cov}(X,\boldsymbol{y}) = X'\boldsymbol{y}$. Because $X'\boldsymbol{y}$ is a vector, no iteration is required as there is only one possible direction of the greatest covariance of X and $\boldsymbol{y}$.

It is notable that a MLS regression model that requires the computation of $(X'X)^{-1}$ and the PCR model requires both the iterations required to compute the eigenpairs of the data matrix and the computation of $(T'T)^{-1}$. The computation of an inverse of a matrix is bounded by $O(n^3)$ flops of a computer processor and is considered to be an extremely expensive operation (Benner et al. 2012; Higham 1996, 261-285). The direct computation of the $\boldsymbol{w}$ vector in the PLS1 algorithm avoids the need for iterations required to find eigenvectors. Also notice that the PLS1 algorithm computes the T matrix one column vector at a time. Thus, $\boldsymbol{t't}$ is a scalar which implies that its inverse is the scalar $1/(\boldsymbol{t't})$. As a result, the PLS1 algorithm does not require the computation of an inverse of a matrix. Thus, the PLS1 algorithm is significantly less computationally complex than MLS regression or PCR. In fact, computation of the PLS1 algorithm for a 2500×4200 data matrix takes less than five seconds on a desktop personal computer.

The PLS1 algorithm has many variants with slightly different characteristics and performance, but most share the same common functionality (Andersson 2008, 518–529). Below is one such implementation of the PLS1 algorithm in C++ ;

```
w=X.Transpose()*y                                                    (11)
w=w*(1.0/VectorNorm(w,2));
t=Working_X*w;
q=(t.Transpose()*y)*(1.0/(t.Transpose()*t));
p=(Working_X.Transpose()*t)*(1.0/(t.Transpose()*t));
```

The same deflation method is applied to the PLS1 algorithm as is used for the PLS2 algorithm when finding more than one component.

When the columns of a data matrix are uncorrelated and $r = p < n$, then the PLS prediction is identical to MLS prediction. When $r < p < n$, the prediction capability of PLS is dependent on the data. That is, if each variable has some correlation to the response, then the MLS regression may out-perform a PLS model with reduced dimensions as it may omit some contribution from some regressors. However, consider that there exists some number $i<p$ such that, when $r=i$, all the correlation to the response is included in the PLS model. Then, the PLS model is equivalent to the MLS model and the PLS model is the parsimonious model (Martens and Naes 1989, 163-165). Again, this assumes that the regressors are uncorrelated. In practice, there is almost always some level of dependency relationship between the columns of the data matrix which insures that the reduced dimension PLS model will produce a more stable estimation than MLS regression. Further more, a subset of components may also filter out components in X that are uncorrelated to Y and, therefore, the PLS model becomes less susceptible to uncorrelated disturbances.

The conclusion is that the PLS model is often more stable and more parsimonious than the MLS regression or PCR model. In addition, the PLS1 model is significantly faster to compute than the other regression methods. The conclusion is that it would be difficult to describe a situation in which the MLS regression model and PCR model would be a better choice that a PLS model.

5.2.3 Diagnostics for PLS

The implementation of the PLS algorithm shows the explicit relationship between the *Q* and *T* matrices. However, if the entire *T* matrix is assumed to be already calculated, then the *Q* matrix may be obtained by $Q=(T'T)^{-1}T'Y$. This is the familiar formula for the coefficients of a MLS regression (Montgomery, Peck, Vining 2006, 69). Thus, once the components have been computed, PLS is equivalent to a MLS regression of *Y* on *T*. As such, the *hat matrix* of the PLS regression model can be computed as $H=T(T'T)^{-1}T'$. This critical relationship validates the application of many MLS regression diagnostic methods and plots for a PLS model. Note that the residual degrees of freedom of a PLS model is n-r-1, as the number of regressors is the number of components r. Consider the following diagnostics methods as applied to PLS.

1. An *outlier* is an observation with a larger than expected residual where the *residual* is defined as the difference between the actual value of the response and the predicted value from the model. Outliers detection may be implemented using the standard error of the regression d_i computed as

$$d_i=\frac{e_i}{\sqrt{MSE}}=\frac{y_i-\hat{y}_1}{\sqrt{\frac{SS_{res}}{n-r-1}}}=\frac{y_i-\hat{y}_1}{\sqrt{\frac{\sum_{k=1}^{n}(y_k-\hat{y}_k)^2}{n-r-1}}} \tag{12}$$

where $|d_i|>3$ indicates an outlier (Montgomery, Peck, and Vining 2006, 123).

2. The PRESS statistic may be used to indicate the prediction capability of the PLS

model and is computed as

$$PRESS = \sum_{i=1}^{n} \frac{e_i}{\sqrt{MSE}\left(1 - h_{ii}\right)} \tag{13}$$

Where h_{ii} is the scalar at the ith row and ith column of the hat matrix. A small value for the PRESS statistic indicates good prediction capability of the model (Montgomery, Peck, and Vining 2006, 142).

3. A *high leverage* point is a data point that lies a far distance from the center of the data (centroid). A high leverage point has a high influence on the model, thus, is worth investigating. High leverage points of the PLS model are indicated by the value of h_{ii}. The general rule is that if $h_{ii} > 2(r+1)/n$ then the ith observation is a high leverage point (Montgomery, Peck, and Vining 2006, 191).

4. Diagnostic plots commonly used to visualize a MLS regression model are also applicable to PLS. Some common plots that have been successfully used with a PLS model are residual plots, leverage plots, and the normal Q-Q plot.

A PLS model provides an estimate of the mean response by an interpolation of values in the training set. The space defined by the data matrix is called the *scope* of the model. However, what happens if the a new observation is outside of the scope of the model? This is referred to as *extrapolation* and will cause the prediction to be unstable.

Refer back to figure 2.1 and consider the extreme data points in the direction of either principal component. Notice that an extreme value in the direction of one principal component is rarely an extreme value in the direction of the another principal component. That is, the space defined by the data is rarely a box, but, more often, is an ellipse. Thus, it is possible for an observation to have values within the range of variables used to generate the model, but the data point may lie outside the scope of the model. That is, the data point may lie within the box that contains the ellipse that represent the scope of the model, but not lie inside the ellipse itself. *Hidden extrapolation* occurs when such an observation is used to make a prediction of the response.

Hidden extrapolation for a PLS model can be detected with the hat matrix. Consider a new observation $\boldsymbol{x_{new}}$, then the $\boldsymbol{t_{new}}$ vector is obtained by (10). If

$$t_{new}'(T'T)^{-1}t_{new} > h_{max}, \tag{14}$$

where h_{max} is the maximum value of h_{ii} for $i \in \{1, \ldots, n\}$, then $\boldsymbol{t_{new}}$ is considered outside of the scope of the PLS model and the corresponding estimation of the response should be considered an extrapolation (Montgomery, Peck, and Vining 2006, 101-103).

Notice that the larger the number of dimensions of the data matrix, the larger the chance that there may be hidden extrapolation. It becomes hard to represent the entire scope of the model as the number of dimension increases. Consider a relatively small data matrix with 10 variables and each variable is split into five equal size segments of its range (similar to a factor in an ANOVA model). The ideal data matrix has an observation in every segment for every variable. This would require $5^{10} = 976562$ many observations and that assumes perfect control over every observation. Real data often follows a distribution that will result in many duplicates. The result is that most data matrices do not contain enough observations to cover the entire scope of the variable ranges. Thus, detection of hidden extrapolation is beneficial when working with real data of any size.

The formula for hidden extrapolation can also be thought of as a simple convex hull of the data matrix in the shape of a hyper-elipsoid. Consider the matrix *T'T* as the covariance matrix of *T*. As the columns of T are uncorrelated and the eigenvalues from the data matrix are the variances of the columns of *T* we can interpret (14) as

$$\boldsymbol{t}'\Sigma^{-1}\boldsymbol{t} = \frac{t_1^2}{\lambda_1} + \frac{t_2^2}{\lambda_2} + \ldots + \frac{t_r^2}{\lambda_r} > h_{max}, \tag{15}$$

where Σ is the diagonal matrix of eigenvalues, t_i is the *i*th component of the $\boldsymbol{t}$ vector and, any $\boldsymbol{t}$ vector that satisfies (15) is outside the convex hull of the regressor data. Thus, the eigenvalues are the lengths of the half-axis of the hyper-elipsoid that circumscribes the data (Hotelling 1933, 426-429; Rencher 2002, 381). The value of h_{max} represents the furtherest point from the center of the data (centroid) of the data matrix and, therefore, is be used to calibrate the proportions of the half axises lengths identified by the eigenvalues.

The PLS algorithms and associated methods of diagnostics have proved to be valuable regression methods and have been embraced by statisticians and mathematicians alike. Over the last 30 years, Many papers have been published on PLS and many adaptations have been proposed to handle a large range of data. A significant endorsement to the validity of PLS came from the US/FDA in the form of the Process Analytical Technology (PAT) initiative that identified PCA and PLS as statistical methods recommended to pharmaceutical producers for validation of drug production (Montague 2008). As such, there has been a particular interest in statistical process monitoring with PCA and PLS in the pharmaceutical industry.

The next chapter presents an an application of PLS to estimate the parameters of a pharmaceutical bioreactor simulator. The application includes examples of many of the diagnostics methods presented in this chapter.

CHAPTER 6
AN APPLICATION OF PLS FOR PARAMETER ESTIMATION

The proteins from living organisms are an active component of many modern pharmaceuticals. However, the kinetics of mammalian cells, used to produce these proteins, are not well understood and standard production practices are not well established. Making changes to production or performing experiments on a production process is often cost prohibited. Thus, many production facilities are run based on holistic knowledge of on-site operators and engineers. The use of computer simulators can greatly reduce the time and cost of making experimental changes to the production process. Gregory McMillan has produced a general use computer based mammalian cell bioreactor simulator (Boudreau and McMillan 2007). The simulator requires that 22 cell parameters be identified in order to match a particular production facility. A PLS model was generated to estimate each unknown parameter of the simulator. The expectation is that, with production runs data from a particular facility, each model will estimate a single parameter. With the parameters of the simulator identified for the particular production facility, the simulator could be used to explore new production scenarios.

6.1 Mammalian Cell Proteins in Pharmaceutical Drugs

The history of mammalian cell proteins in pharmaceutical drugs goes back to 1798 when

Dr. Edward Jenner's seminal research was published. Dr. Jenner's research offered a conclusive study describing the application of a mammalian protein as a vaccine for smallpox. The smallpox vaccine that Dr. Jenner developed came from pus in blisters on a cow infected with cowpox (Riedel 2005, 21–25). The protein structures in the cow pus are the primary agent of the smallpox vaccine. Modern pharmaceuticals such as insulin for diabetes, erythropoietin (EPO) for anemia, remicade for rheumatoid arthritis, and rituxan for lymphoma all contain a protein as the primary agent (Tomlinson 2004, 521 - 522).

Whereas Dr. Jenner scraped the pus straight from the cow, modern production of proteins (bioprocessing) has developed into a process of growing the living cells of an organism in a bioreactor. The bioreactor is a vessel filled with a solution containing nutrients and components required for the the cells to grow while separated from the living organism of origin. A seed culture of cells is introduced into the bioreactor and the cells grow while consuming the nutrients and secreting proteins and waste. The bioreactor is infused with more nutrients to maintain a balance that supports the growth of the cells and production of the proteins (Boudreau and McMillan 2007). Before the cell growth exceeds the capacity of the bioreactor, the proteins are harvested and the cells are destroyed. The process is extremely complex and not well understood. It is not yet possible to measure the protein concentration of the bioreactor without destroying the cells. It is even difficult to measure the cell concentration of the bioreactor during the processing, thus making it a problem to control nutrients levels or to know when to stop the batch.

6.2 Bioreactor Simulators and Parameter Estimation

The use of mammalian cells proteins is at the forefront of pharmaceutical production. Greg McMillan's research on mammalian cell bioprocessing has lead to his development of a comprehensive computer based mammalian cell bioreactor simulator. The simulator is a based on the first principles of physics that include the laws of conservation, energy balances and mass balances (Roffel and Betlem 2006, 20). The simulator mimics the behavior of a bioreactor through the phases of cell growth from the charging of the bioreactor with the initial solution, introduction of the seed culture, the growth of the cells, and the final protein content. The simulator requires a set of parameters of the cell kinetics that effect the bioreactor process. The idea is, given the correct values of the parameters, the simulator will mimic the actual production of a particular mammalian cell bioreactor. Once the parameters are identified, the simulator can be used to test production scenarios without costly and time consuming trials.

The task of identifying the parameters of the simulator has been a manual precess of trial and error and is often heavy on the error side. The ultimate goal is to develop an analytical method of estimating the parameters of a particular processing facility using actual data from batches run at that particular facility. The method being presented involves a separate PLS model to estimate each parameter. To generate the data for the model development, the parameters values of the simulator are randomly varied across the range of each parameter and simulated batched are run. The data collected from the simulated batches are used to generate the models. An iterative method will be employed that successively estimates each parameter and updates the values of the regressor for the subsequence estimation. It is expected that iterations of estimations will converge to reasonable estimates of the parameters. The method is called successive parameter estimation (F'ukunaga 1990, 385).

However, for this to work, it must be exhibited that such an estimation is possible. Specifically, given a set of sensor data with all the parameters known except one, can a model estimate the last parameter? This is a necessary requirement to show that such an iterative estimate of all the parameters will converge. This study will consider the results of the PLS models generated to estimate each parameter with all other parameters known.

6.3 Data Collection and Preprocessing

The data from each simulated production batch is stored sequentially in a database. During the batch there are 30 different sensors values that are calculated based on cell kinetics and first principles of physics. Sensor data is recorded at different rates and different intervals. Thus, the resultant data of one batch is 30 vectors of varied length. Each vector starts and ends at the same time, but the readings are not aligned. A standard time interval is chosen and the data is interpolated across the raw data vectors. The result is a k by p matrix of data representing the batch sensor readings where k is the number of evenly spaced intervals in the batch and p=30 is the number of sensors. Notice that, depending on the interval of the interpolation, this may also have the effect of reducing the size of the data.

The stopping criteria for a production bioreactor is not well established across the industry.

The simulator stops a batch when the cell concentration reaches a certain level. It is assumed that the real bioreactor is equipped with an at-line analyzer that estimates cell concentrations from a sample taken every four hours. As the cell growth varies based on the level of nutrients, oxygen level, and cell kinetics (parameters of the simulator), each batch runs for a different duration of time. Thus, with n many batches, there are n many k_i by p matrices of data where each k_i is different.

To align the lengths of each batch, a method of dynamic time warping (DTW) is employed. DTW uses a distance calculation as a metric and Dijkstra's lowest cost path algorithm to minimize the adjustments required to transform each batch to a uniform length. Adjustments are made by averaging or duplicating observations. The DTW process can cause some loss or mutation of the data, however results are generally a good alignment of the data based on similar features. The implementation used follows the details given in the excellent papers (Kassidas, MacGregor, Taylor 1998, 864-875; Ramaker et al. 2003, 133–153). The results of DTW are n many data matrices of uniform size $k \times p$.

The simulator parameters are merged into each batch data matrix as constant values across the k many intervals. The parameters augment the dimension of each observation (row) to $30+22=52=p$. Notice that this count includes the variable to be estimated as it will be separated later for each particular model. Because each parameter has a constant value across the entire batch, they can be added to the data matrix before or after the DTW without a change in the results. In order to minimize processing time, they are added after the DTW step.

Each batch data is then unfolded *batch-wise* to form a vector. The vector is constructed by taking each of the 52 variables for each time slice and lining them up, end to end, to form a $1 \times kp$ vector. The chronological order of the time slices are maintained and the order of each sensor within each time slice is maintained. The vectors for each batch are then stacked in a matrix to form a $n \times kp$ data matrix (Camacho and Pico 2006 1021-1035; Lee, Yoo, and Lee 2004, 119–136).

The data matrix is preprocessed to be mean centered by column. The data in each column is also normalized by the standard deviation of the column. The mean centering simplifies the computations and the scaling removes bias from different measurement scaling. The columns for the response variable is extracted to form the $X_{parameter}$ data matrix and the $\boldsymbol{y}_{parameter}$ vector of the response variable. The results are 22 data matrices and 22 corresponding response vectors. The matrices are now ready for the PLS1 algorithm.

In total, the data used to generate the models contains 2366 batches from the simulator. Each batch has 51 regressors and 81 time slices. Thus, the $X_{parameter}$ data matrix for each model is 2366×4131 and each $\boldsymbol{y}_{parameter}$ vector is 2366×1. The size of the data matrix used to generate each model is 111.6 megabytes.

A remark is made on the size of the data. The primary focus of this study is dimension reduction of data. The data matrix describes has a large number of column and, thus, is appropriate for this study. However, overall, the data set is not too large to fit into the memory of a computer and would not be classified as big data. For big data, some form of sampling would be required to reduce the size of the data to fit into the memory of a computer in order to generate a PLS model. It is noted that the the PLS1 algorithm is very efficient and computed the final model for each parameter in about 5 seconds on an Intel® i7 processor running at 4GHz. Thus, a much larger data matrix could be processed. The limiting factor of the size of the data for this study was the time required for the simulator to run the batches. At an average rate of 50 batches per day, it took over 48 days to generate the final data that was used in this study.

It is also noted that an MLS model would fail for this application as the number of regressors (columns) is larger than the number of observations (rows or batches). The PLS model will rely on the dimension reduction capability and multicollinearity of the data. The multicollinearity comes from the repeated sensor readings and that multiple sensors are reading the same physical phenomenon. The result is that the model will have a significant reduction is dimension that will permit a regression.

6.4 PLS Model Building and Diagnostics

There are two major steps involved in constructing the regression model. The first step involves the analysis of data to identify erroneous observations that can affect the model performance. The second step concerns which information to include in the model to best estimate the parameter.

6.4.1 Outliers and High-Leverage points

The fitted values of the model are calculated as the estimated response of the model for the observations used to generate the model. As the actual values of the response are represented in the $\boldsymbol{y}_{parameter}$ vector, the difference between the two is calculated and is called the *residuals* of the model. The residuals are standardized as in (12) and an *outlier* is identified as having a standardized residual of larger than ±3. An outlier represents an observation that does not conform to the linear relationships of the model. As the data for the models in this study were computer generated, there were no outliers that were determined to represent erroneous data.

The influence of each data point was evaluated to identify *high-leverage* points as described in Chapter 5. As high-leverage points have a high influence on the model, each point identified as a high-leverage was investigated, but none were determined to be erroneous. There should be special attention paid to high-leverage point with high dimensional data as it may be difficult for the training set to represent the entire intended scope of the model. As a result, there may be many observations identified as high leverage. Only points determined to be errors should be removed. Figure 6.1 is a plot of the observation influence versus the standardized residual for the cell hydrolysis rate parameter.

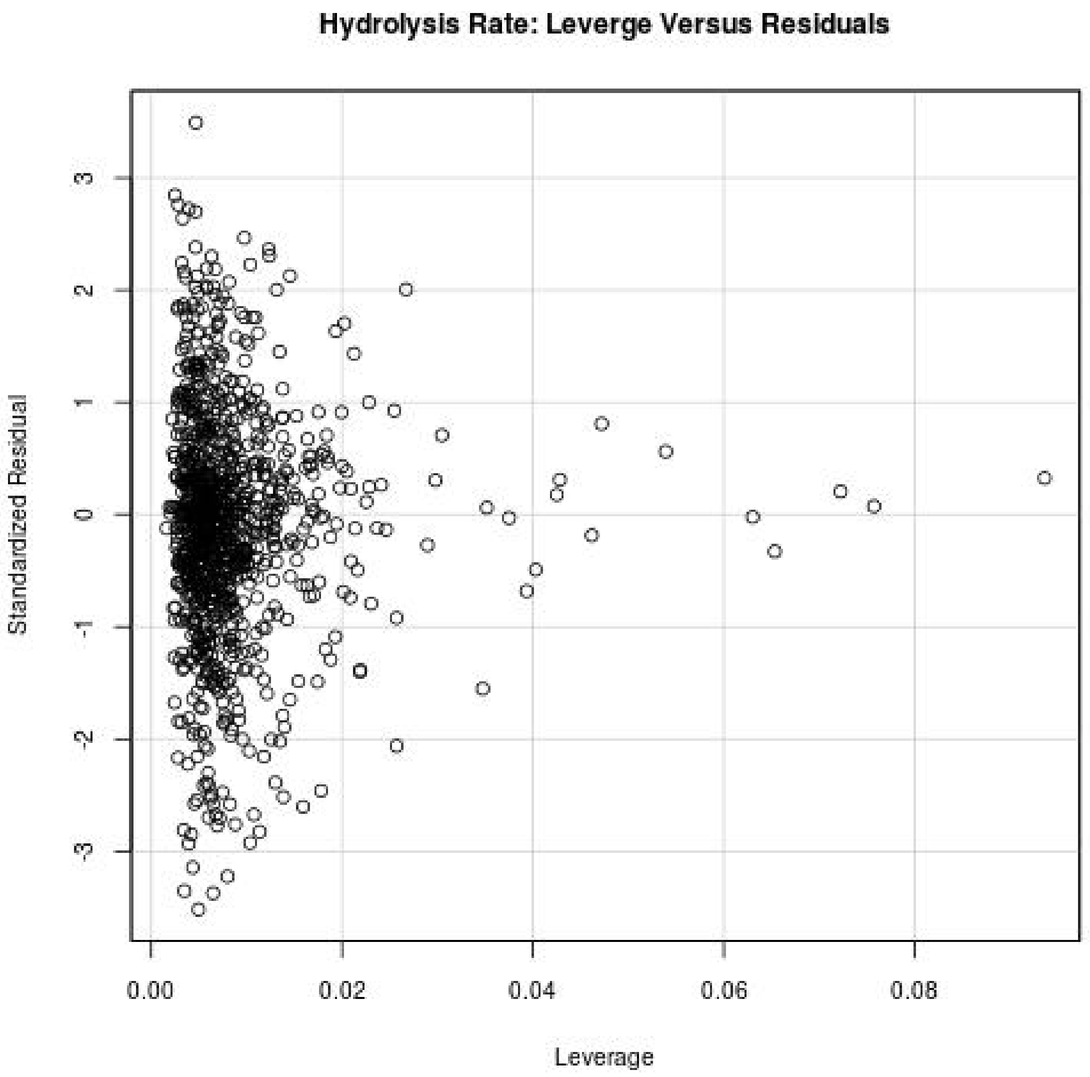

Figure 6.1: PLS leverage versus standardized residuals for hydrolysis rate.

The high-leverage points are represented as points on the right side of the plot in Figure 6.1. Notice that the high-leverage points are not outliers and that the plot has the form of a right pointing isosceles triangle about the x-axis. This triangle shape indicates that the high-leverage points do not contradict the majority of the data in the model. Thus, there is not problem with high-leverage points in the model.

6.4.2 Model Building

Building a MLS regression model is a matter of selecting which regressors to include in the model and which to omit. For a PLS model, all regressors are included in the model and the selection of the number of principal components determines which information is retained and which is omitted.

As mentioned in Chapter 4, there are many formulas for estimating the optimal number of principal components for a model. The motivation for most of these methods is that it was very time consuming to generate a model with a large numbers of principal components. However, the speed of modern computers now makes this possible and, along with a computational trick, models can be constructed with a range of principal components. As a result, the different models can be evaluated directly.

The trick is to compute one model with a high number of principal components and then use subsets of the matrices to assemble the other models. For this study, a model was computed with 20 principal components for each parameter. A model with 19 principal components was assembled by removing the last column of the matrices from the 20 principal component model. Then, an 18 principal component model was assembled by removing one column from the 19 principal component model, and so on. The result is an efficient method to generate twenty models from one large model that can be quickly compared for performance.

Table 6.1 shows the comparison of models with one through twenty principal components for the cell hydrolysis rate parameter. Notice that that all four statistics in Table 6.1 show a diminishing returns or a leveling out at the six to seven principal components range. This indicates that the optimal principal components is in this range.

The better comparison of models can be made with the use of test sets. The set of observations used to generate a model is called the *training set*. In contrast, a *test set* is a set of observations withheld from the training set and used to simulate new observations for the purpose of evaluating model performance. Recall that the goal of the models is to estimate the simulator parameters for set of batches from a production facility. The set of batches from the production facility will have base set of parameter values that are reflected in each batch. However, each batch will also have some amount of error and variation from the base values. Thus, the best test set to reflect this is a set of batches generated with similar parameters values.

To construct such a test set, base values of the parameter were established. Then, a set of batches were generated with a slight random deviation from the base parameter values. The result is a set of batches with similar parameters. Different base values were used to generate multiple test sets with similar parameter. Each test set was constructed with 50 batches and a total of 20 tests sets were generated.

Table 6.1: Comparison of models and number of principal components.

# PCs	MSE	R^2	Adjusted R^2	PRESS
1	4.10798157E-007	0.5149139681	0.514708771	0.0009718525
2	3.20372809E-007	0.621851675	0.6215316172	0.0007583655
3	2.14708063E-007	0.7466790792	0.7463573337	0.0005085692
4	1.79901253E-007	0.7878353693	0.7874759206	0.0004263011
5	0.000000172	0.7972727552	0.7968432483	0.0004076684
6	1.63840097E-007	0.806940602	0.806449565	0.000388486
7	1.60306765E-007	0.8111841444	0.8106236224	0.0003802205
8	1.5777305E-007	0.8142472668	0.8136167951	0.000374325
9	1.5559499E-007	0.8168893073	0.8161898182	0.0003693048
10	1.54632067E-007	0.8180997555	0.8173273553	0.0003671678
11	0.000000154	0.8188933985	0.8180471059	0.0003658197
12	1.53304639E-007	0.8198144192	0.8188954957	0.0003641961
13	1.52316975E-007	0.8210513474	0.8200622604	0.0003619846
14	1.51621723E-007	0.8219438958	0.8208835872	0.0003604833
15	1.50888277E-007	0.8228805843	0.8217500348	0.0003588369
16	1.50259489E-007	0.8236937393	0.8224928453	0.0003574352
17	1.49316224E-007	0.8248750994	0.8236071594	0.0003553342
18	1.48814607E-007	0.8255377535	0.8241997388	0.0003541984
19	1.48218041E-007	0.8263111715	0.8249044845	0.0003528893
20	1.4749672E-007	0.8272301245	0.8257566074	0.0003513036

The performance of each model was evaluated using the average of the mean squared error of then batches in each test set. The test set evaluation for the hydrolysis rate parameter is presented in table 6.2.

Notice that the lowest average mean squared error in table 6.2 occurs for the model with six principal components. The final model for the parameter hydrolysis rate was selected to have six principal components.

Table 6.2: Test set statistic comparison by principal component.

# of PCs	Average MSE
1	8.144570346E-006
2	8.905167916E-006
3	0.000003758
4	4.261417573E-006
5	4.310589688E-006
6	3.737604055E-006
7	3.843223187E-006
8	4.013847955E-006
9	4.115186803E-006
10	4.357300507E-006
11	4.427686347E-006
12	0.000004295
13	4.381356925E-006
14	4.543801186E-006
15	4.523543118E-006
16	4.924752642E-006
17	5.097811953E-006
18	5.456273425E-006
19	0.000005876
20	6.232219059E-006

6.4.3 Hidden Extrapolation Detection

As discussed in Chapter 5, the larger the dimension of the data, the greater the chance that a data point will lie outside of the scope of the model. If a test set contains a point outside of the scope of the model, then the model's ability to predict a response will be better than reported. In the case of estimation of the response from a new observation that is outside of the scope of the model, the estimate will be unstable. How unstable can the estimation be? Consider Figure 6.2 as an example of an estimation of the response with a point outside of the scope of the model.

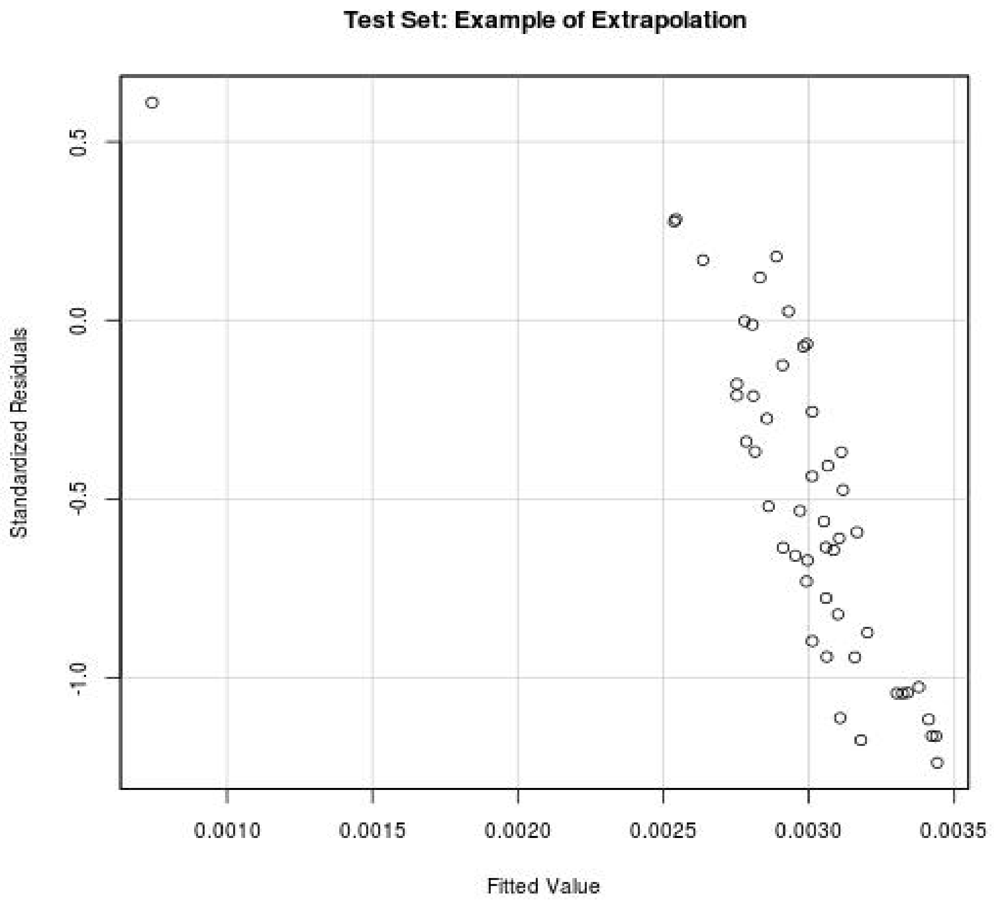

Figure 6.2: An example of the effect of extrapolation.

The calculation used to identify hidden extrapolation was implemented as in Chapter 5. Of the test sets, a few test sets were found to have a single batch with observations that were outside of the scope of the model. Data from one such test set is presented in Figure 6.2. The point on the plot that represents the observation that was detected as outside of the scope of the model is in the upper left corner, significantly away from the grouping on the right. Notice that the largest standardized residual is not the extrapolated fitted value. Thus, the plot could be misread concerning the meaning of the deviated point. Knowing that the fitted value is a result of extrapolation adds insight that the particular fitted value is not to be trusted.

The batches detected with observations outside of the scope of the model in each test set were removed as to not skew the evaluation of the models.

6.5 Parameter Estimation Results

Of the 22 PLS models generated, 15 adequately estimated the parameter. Thus, the method of PLS appears to have performed well and, in the cases where the PLS model did not perform well, the data did not appear to have any observable correlation to the response.

The choice of the number of principal components in 18 of the models was 1. Thus, there is a single relationship that underlies many of the parameters. The four other models had 6, 6, 4, 3 as the number of principal components best suited to for the model. There did not appear to be any correlation between the models that performed well and the number of principal components used in the model.

The use of diagnostics methods adapted from MLS regression was instrumental in evaluation the models. Detection of hidden extrapolation allowed for the removal of batches from the test sets to ensure an accurate assessment of the model performance.

CHAPTER 7

CONCLUSIONS, REMARKS AND DIRECTION OF FUTURE RESEARCH

The use of eigenvalues and eigenvectors for the reduction of the dimension of data is based on a solid foundation of mathematical theory. The original method for finding eigenvectors as proposed by Harold Hotelling in 1933 is the the Power Method and is the basis of many data dimension analysis and reduction algorithms. As a result, the methods are mathematically stable and exhibit well understood behavior.

The PLS algorithm includes a useful set of features for the dimension reduction and regression of data. In particular, PLS performs very well with data sets with multicollinearity, especially data resulting from time series observations. As a result, PLS is well suited for the analysis of industrial production data.

As the correlation of the data matrix to the response is maximized with the PLS algorithm, the pitfall of PCR is avoided. The results of a PLS regression are either more stable or equivalent to a MLS regression.

Once the principal components of a PLS methods are computed, the PLS regression coefficients can be computed as the MLS regression coefficients of the principal components and the response. This connection allows for of a rich set of diagnostics tools and data visualization be used to evaluate a PLS model.

The PLS1 algorithm is more efficient than PCR and MLS regression algorithms. As a result, the PLS1 algorithm enables very large data sets to be modeled in a matter of seconds. The speed of the algorithm and some matrix manipulation also allow for a direct comparison of models with different numbers of principal components by only computing one large model.

Many of the bioreactor simulator parameters were estimated well with a linear PLS model. Non-linear adaptations to PLS were investigated during this study and, in general, did not perform any better than the linear PLS. It is concluded that a linear model is the best model for data with very little correlation to the response.

Additional research is required to estimate the remaining seven parameters before the iterative successive estimation can be implemented. Although there is always an interest in non-linear models, it would seem that working on algorithms that can extract linear relationships from data with little apparent correlation would have value in this situation.

As the each parameter will be estimated with a set of batches, possibly the model could identify which batches that do estimate the parameter well and focus on the internal structure of those batches. A possible way to do this would be to examine the variances of the regressors of the models that do predict well to find a subset of the data that has as stronger correlation to the response. These relationships may be represented by eigenpairs with smaller corresponding eigenvalues.

There are many possibilities for future research and a more thorough understanding of eigenpairs, the Power Method, and the PLS algorithms is an excellent base for a continued study into more advanced linear and non-linear regression algorithms.

BIBLIOGRAPHY

Agneeswaran, Vijay Srinivas. 2012. Big-data: Theoretical, engineering and analytics perspective. In *Big Data Analytics First International Conference, BDA 2012, New Delhi, India, December 24-26, 2012. Proceedings.* ed. Srinath Srinivasa and Vasudha Bhatnagar: 8–15. New York: Springer.

Andersson, Martin. 2008. A comparison of nine pls1 algorithms. *Journal of Chemometrics 23:* 518–529.

Beezer, Robert A. 2004. *A first course in linear algebra*. GNU Free Document License, http://linear.ups.edu/download/fcla-electric-2.99.pdf(accessed April 22, 2013).

Bernstein, Dennis S. 2009. *Matrix mathematics: Theory, facts, and formulas*. Princeton, NJ: Princeton University Press.

Benner, Peter, Pablo Ezzatti, Enrique S. Quintana-Ortí, and Alfredo Remón. 2012. Matrix inversion on cpu–gpu platforms with applications in control theory. *Concurrency and computation: practice and experience.* ed. Geoffrey C. Fox and David W. Walker. New York: John Wiley & Sons, Inc.

Bergen, Jeffery. 2010. *A concrete approach to abstract algebra: From the integers to the unsolvability of the quintic.* Boston: Elsevier.

Boudreau, Michael and Gregory McMillan. 2007. *New directions in bioprocess modeling and control.* Research Triangle Park, NC: ISA.

Bronson, Richard. 1989. *Shaum's outline of theory and problems of matrix operations.* New York: McGraw-Hill.

Camacho, Jose and Jesus Pico. 2006. Online monitoring of batch processes using multi-phase principal component analysis. *Journal of Process Control* 16: 1021–1035.

Corrochano, Eduardo Bayro. 2005. *Handbook of geometric computing: Applications in pattern recognition, computer vision, neural computing, and robotics*. Berlin : Springer.

Fine, Benjamin and Gerhard Rosenberger. 1997. *The fundamental theorem of algebra.* New York: Springer.

Friedberg, Stephen H., Arnold J. Insel and Lawrence C. Spence. 1989. *Linear algebra.* New Jersey: Prentice Hall.

F'ukunaga, Keinosuke. 1990. *Introduction to statistical pattern recognition.* San Diego: Academic Press.

Geladi, Paul and Bruce R. Kowalski. 1986. Partial least-squares regression: A tutorial. *Analytica Chimica Acta* 186: l-17.

Geladi, Paul. 1988. Notes on the history and nature of partial least squares (pls) modelling. *Journal Of Chemometrics* 2: 231-246.

Gentle, James, E. 2007. *Matrix algebra.* New York: Springer.

Golub, Gene H. and Charles F. Van Loan. 1996. *Matrix computations*. 3rd ed. Baltimore: Johns Hopkins University Press.

Good , Phillip I. 2006. *Resampling methods: A practical guide to data analysis*. Boston: Birkhuser.

Halmos, Paul R. 1974. *Finite-dimensional vector spaces.* New York: Springer.

Han, Jiawei and Micheline Kamber. 2006. *Data mining: Concepts and techniques*. Amsterdam: Elsevier.

Hasegawa, Takeshi. 2006. Principal component regression and partial least squares modeling. In *Handbook of vibrational spectroscopy.* New York: John Wiley & Sons, Inc.

Higham, Nicholas J. 1996. *Accuracy and stability of of numeric algorithms.* Philadelphia: Society for Industrial and Applied Mathematics.

Hirsch, Morris W. and Stephen Smale. 1974. *Differential equation, dynamic systems, and linear algebra*. San Diego: Academic Press.

Hoffman, Kenneth and Ray Kunze. 1971. *Linear algebra.* 2nd ed. New Jersey: Prentice-Hall.

Horn, Roger A. and Charles R. Johnson. 1985. *Matrix analysis.* Cambridge: Cambridge University Press.

Hotelling, Harold. 1933. Analysis of complex statistical variables into principal components. *Journal of Educational Psychology* 24, no 6: 417-441.

Householder, Alston. 1964. *The theory of matrices in numeric analysis.* New York: Dover Publications, inc.

Jackson, J. Edward. 1991. *A user's guide to principal components*. New York: John Wiley & Sons, Inc.

Johnson, Richard A. and Dean W. Wichern. 2007. *Applied multivariate statistical analysis*. 6th ed. New Jersey:Pearson Prentice-Hall.

Jolliffe, I.T. 2002. *Principal component analysis*. 2nd ed. New York: Springer.

Kassidas Athanassios, John F. MacGregor, and Paul A. Taylor. 1998. Synchronization of batch trajectories using dynamic time warping. *AIChE Process Systems Engineering* 44, no.4: 864-875.

Khattree, Ravindra and Dayanand N. Naik. 2000. *Multivariate data reduction and discrimination with sas® software.* Cary, NC: SAS Institute Inc.

Kuttler, Kenneth. 2004. *Elementary linear algebra*. http://www.math.byu.edu/~klkuttle/0000ElemLinearalgebratoprint.pdf(accessed March 21, 2013).

Lavallee, Pierre. 2007. *Indirect sampling.* New York: Springer.

Lay, David C. 2003. *Linear algebra and its applications.* Boston :Addison Wesley.

Lee, Dae Sung, and Peter A. Vanrolleghem. 2002. Monitoring of a sequencing batch reactor using adaptive multiblock principal component analysis. *Biotechnology and Bioengineering* 82, no.4: 489–497.

Lorber, A., L. E. Wangen and B. R. Kowalski. 1987. A theoretical foundation for the pls algorithm. *Journal of Chemometrics* 1, no.19: 19-31.

Lütkepohl, H. 1996. *Handbook of matrices.* New York: John Wiley & Sons.

Mathews, John H. and Kurtis D. Fink. 1999. *Numerical methods using matlab.* 3rd ed. New Jersey: Prentice-Hall.

Montague, Jim. 2008. Is pat a silver bullet. *Control Global*(online) June. http://www.controlglobal.com/articles/2008/185.html(accessed March 26, 2013).

Montgomery, Douglas, Elizabeth Peck, and G. Geoffrey Vining. 2006. *Introduction to linear regression analysis.* New York: John Wiley & Sons, Inc.

Office of the White House, Press Secretary. (2012). Obama administration unveils "big data" initiative: announces $200 million in new r&d investments[press release]. http://www.whitehouse.gov/sites/default/files/microsites/ostp/big_data_press_release_final_2.pdf(accessed March 28th, 2013).

Parlett, Beresford N. 1998. *The symmetric eigenvalue problem.* Philadelphia: Society for Industrial and Applied Mathematics.

Phatak, Aloke and Sijmen De Jong. 1997. The geometry of partial least squares. *Journal of Chemometrics* 11: 311–338.

Piziak, Robert and P. L. Odell. 2007. *Matrix theory: From generalized inverses to jordan form*. New York: Chapman & Hall/CRC.

Qin, S. Joe. 2003. Statistical process monitoring: basics and beyond. *Journal of Chemometrics* 17: 480–502.

Ramaker, Henk-Jan, Eric N.M. van Sprang, Johan A. Westerhuis, and Age K. Smilde. 2003. Dynamic time warping of spectroscopic batch data. *Analytica Chimica Acta* 498: 133–153.

Reiss, Randolf, Willy Wojsznis and Robert Wojewodka. 2010. Partial least squares confidence interval calculation for industrial end-of-batch quality prediction. *Chemometrics and Intelligent Laboratory Systems* 100: 75–82.

Remmert, Reinhold. 1991. *Theory of complex functions.* New York: Springer.

Rencher, Alvin C. 2002. *Methods of multivariate analysis*. New York: John Wiley & Sons, Inc.

Riedel, Stefan. 2005. Edward jenner and the history of smallpox and vaccination. *Proceeding (Bayl Univ Med Cent), National Center for Biotechnology Information. U.S. National Library of Medicine* 18, no.1: 21–25.

Roffel, Brian and Ben Betlem 2006, Process dynamics and control: Modeling for control and prediction. New York: John Wiley & Sons, Inc.

Rudin, Walter. 1976. *Principles of mathematical analysis.* New York: McGraw-Hill.

Saad, Yousef. 2011. *Numerical methods for large eigenvalue problems*. Philadelphia: Society for Industrial and Applied Mathematics.

Schilling, Otto F. G. 1975. *Basic abstract algebra*. Boston: Allyn and Bacon.

Schott, James R. 1997. *Matrix analysis for statistics.* New York: John Wiley & Sons, inc.

Shilov, Georgi E. 1977. *Linear algebra*. New York: Dover Publications.

Shores, Thomas S. 2007. *Applied linear algebra and matrix analysis*. New York: Springer.

Singh, Sachchidanand and Nirmala Singh. 2012. Big data analytics. In *2012 International conference on communication, information & computing technology, oct: 19-20*. Mumbai, India: IEEE.

Stewart, G. W. 1998. *Matrix algorithms: Volume 1:basic decomposition.* Philadelphia: Society for Industrial and Applied Mathematics.

Sumathi, S. and S. N. Sivanandam. 2006. *Introduction to data mining and its applications.* New York: Springer.

Thomas, George B. 2005. *Thomas' Calculus.* 11th ed. Boston: Pearson Addison Welsey.

Tomlinson, Ian M. 2004. Next-generation protein drugs, *Nature Biotechnology* 22, doi:10.1038/nbt0504-521: 521-522. http://www.nature.com/nbt/journal/v22/n5/full/nbt0504-521.html[accessed March 25, 2013].

Trefethen, Llyod N. and David Bau III. 1997. *Numerical linear algebra.* Philadelphia: Society for Industrial and Applied Mathematics.

Truss, J. K. 1997. *Foundations of mathematical analysis*. New York: Oxford University Press.

Varmuza, Kurt and Peter Filzmoser. 2009. *Introduction to multivariate statistical analysis in chemometrics*. Boca Raton: CRC Press.

Wasserman, Larry. 2004. *All of statistics: A concise course in statistical inference*. New York: Springer.

Wegman, Edward J. and Jeffrey L. Solka. 2005. *Statistical data mining: Handbook of statistics vol 24.* Amsterdam: Elsevier B.V.

Wilkinson, J. H. 1965. *The algebraic eigenvalue problem.*, London: Clarendon Press.

Wold, Herman. 1966. Estimation of Principal Components and related models by iterative least squares. In *Multivariate Analysis: Proceedings of an international Symposium held in Dayton, Ohio, June 14-19, 1965.* ed. Paruchuri R. Krishnaiah: 391-420. New York: Academic Press.

Wold, Herman. 1973. Nonlinear iterative partial least squares (NIPALS) modeling: Some current developments. In *Multivariate Analysis II: Proceedings of an International Symposium on Multivariate Analysis held at Wright State University, Dayton, Ohio, June 19-24, 1972.* ed. Paruchuri R. Krishnaiah: 383-407. New York: Academic Press.

Wold, S., J. Cheney, N. Kettaneh, and C. McCready. 2006. The chemometric analysis of point and dynamic data in pharmaceutical and biotech production (pat): Some objectives and approaches. *Chemometrics and Intelligent Laboratory Systems* 84: 159–163.

Xuefeng, Lü, Cheng Chengqi, Gong Jianya and Guan Li. 2011. Review of data storage and management technologies for massive remote sensing data. *Science China Technological Sciences* 54, no.12: 3220–3232 .

VITA

Randolf Reiss was born February 16th, 1970 to Joann F. Reiss of Lincoln, Nebraska. He graduated from Urbana High School in Urbana, Illinois on May 1988. He moved to Austin, Texas in 1994 and worked for Emerson Process Management as a programmer for 13 years. Randy returned to school to earn his Bachelor of Science in Applied Mathematics from Texas State University-San Marcos in December of 2007. In 2012, He enrolled in the graduate program to pursue a Masters of Science in Mathematics, also at Texas State.

Address: 113 Twin Creek Circle, Dripping Springs, Texas 78620

This thesis was typed by Randolf H. Reiss.