5-2011

Classification System for Impedance Spectra

Carl Gordon Sapp
csapp@utk.edu

To the Graduate Council:

I am submitting herewith a thesis written by Carl Gordon Sapp entitled "Classification System for Impedance Spectra." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

J. Douglas Birdwell, Major Professor

We have read this thesis and recommend its acceptance:

Tse-Wei Wang, Roger Horn

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:
I am submitting herewith a thesis written by Carl Gordon Sapp entitled "Classification System for Impedance Spectra." I have examined the final paper copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

J. Douglas Birdwell, Major Professor

We have read this thesis
and recommend its acceptance:

Tse-Wei Wang

Roger Horn

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

Classification System for Impedance Spectra

A Thesis Presented for

The Master of Science

Degree

The University of Tennessee, Knoxville

Carl Gordon Sapp

May 2011

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

Abstract

This thesis documents research, methods, and results to satisfy the requirements for the M.S. degree in Electrical Engineering at the University of Tennessee. This thesis explores two primary steps for proper classification of impedance spectra: data dimension reduction and effectiveness of similarity/dissimilarity measure comparison in classification. To understand the data characteristics and classification thresholds, a circuit model analysis for simulation and unclassifiable determination is studied. The research is conducted using previously collected data of complex valued impedance measurements taken from 1844 similar devices. The results show a classification system capable of proper classification of 99% of data samples with well-separated data and approximately 85% using the full range of data available to this research.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This report documents research and results to satisfy the requirements for the M.S. degree in Electrical Engineering at the University of Tennessee. The research developed here is shown to be unique among published research in this topic, and results can be beneficial to many active industries. The background information provides an overview of the various approaches published in the field and places what has been accomplished in context.

The research method described later in this thesis provides a two-step process to determine the optimal decision process for classification of complex-valued electrical impedance spectra gathered from hardware devices of similar type but with differing makeup of component parts, grouped according to similar characteristics. Dimensionality reduction is used to reduce the number of dimensions to be analyzed due to redundancy and correlation embedded in the data. The samples are compared with a pair-wise comparator function to determine the degree of similarity, or dissimilarity, between two objects. Other topics investigated include origin of circuit model parameters, tolerances on category assignment, and a comparison of similarity measures in the correct classification of objects. The classification system proposed in this report is capable of correctly classifying up to 99%

of well-seperated data and 85% of the provided data samples. The system's error rate (misclassification of samples) can in some cases be less than 1%.

1.1 Problem Statement

We wish to develop a method to classify electrical impedance spectra efficiently and effectively by comparing each spectrum to a database of spectra with known classification by using applicable measures of spectral similarity, effective dimension reduction methods, and establishment of optimum classification thresholds.

Chapter 2

Background

To understand the methods of this research, it is important to understand the research performed in the past that has led to the current state of information. These topics will give the reader a solid understanding of the background material required and show current research that has been conducted on similar topics.

2.1 Characteristics of Impedance Spectra

Electrical impedance is defined as "A measure of the complex resistive and reactive attributes of a component in an alternating-current circuit" by the Institute of Electrical and Electronics Engineers (IEEE) [1]. Impedance measurements can be represented in several different forms. The most popular methods involve representing the phase angle and magnitude, in a manner similar to measurements in polar coordinates, or as a complex number, in cartesian coordinates [2]. In this research, the impedance will be represented as a complex number.

Electrical impedance is a well-researched and documented property of conductive materials. This section gives the reader a solid understanding of the background material

required to understand the measurements provided in the data and the steps involved in model analysis. An overview of impedance properties, including the effects of basic electrical components on the model's overall impedance along with single/multiple resonant circuits, is detailed. With this information, an understanding of the traits of an electrical impedance spectrum will be used to refine the methods used in comparison of the spectra.

2.1.1 Alternating Current

The use of alternating current (AC) became popular in the late 1800s after limits on the transmission and delivery of direct current (DC) were realized. By this time, many of the basic concepts of direct current were well researched and documented. Therefore, many of the properties of alternating current are derived from the same principles used in direct current circuit analysis. AC differs from DC in that the electric current continuously reverses direction in a sinusoidal pattern. For the purposes of this discussion, only AC voltage sources, or sources where the voltage is regulated to be consistent, will be reviewed; however, the same principles can be applied to AC current sources.

AC voltage is most commonly described in either its time-domain or phasor representation. In the time-domain, the voltage v, as a function of time t, is represented as $v(t) = V_m sin(wt + \theta)$ where V_m is the peak amplitude of the waveform, w is the frequency of the waveform, and θ is the phase shift of the wave. These properties are described in figure 2.1. More often, AC is described using its phasor representation due to the simplicity in calculations in subsequent circuit analysis. The phasor representation makes use of the amplitude, or magnitude, and phase angle to represent the electric current as a function of frequency. The phasor representation may be written using the magnitude and phase in a method similar to the polar coordinate system or as a complex number in a method similar to the cartesian coordinate system. Conversion between the two methods is illustrated in figure

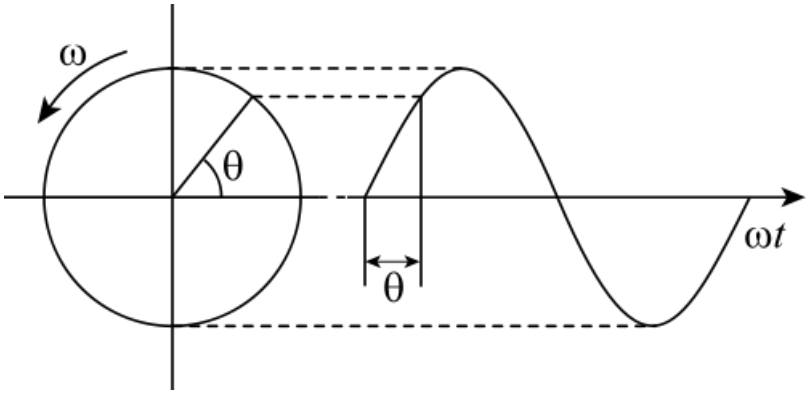


Figure 2.1: Phasor Diagram [3]

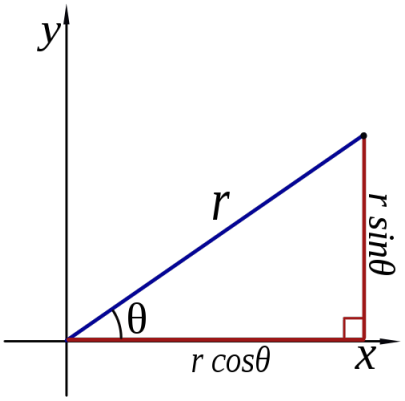


Figure 2.2: Conversion of Polar to Cartesian Coordinates [4]

2.2. The phasor representation in its complex form will be used from this point forward to describe the electric voltage, current, and impedance in the models.

2.1.2 Basic Impedance of Resistors, Inductors, and Capacitors

In DC circuit analysis techniques, calculations involving resistors are not time-dependent. Therefore, the translation of resistance to impedance in AC is a straight-forward one-to-one transformation where the ratio of voltage V to current I still follows Ohm's law, $V = IR$ [5], and R is the impedance. In DC circuit analysis, inductors and capacitors vary with time and therefore have special properties in AC circuit analysis. In AC, an inductor has a voltage to

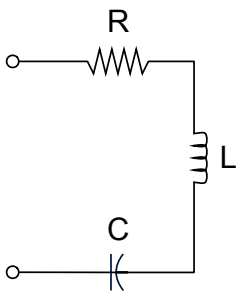


Figure 2.3: RLC Series Circuit

current ratio of $\frac{V}{I} = jwL$ where j is $\sqrt{-1}$, w is the AC frequency, and L is the inductance in Henry. The impedance of the inductor is jwL. A capacitor has a voltage to current ratio of $\frac{V}{I} = \frac{1}{jwC}$ with the impedance being $\frac{1}{jwC}$. Notice that the impedance of a resistor is purely real, resulting in the magnitude equalling the impedance with a zero degree phase shift. In addition, the inductor and capacitor have purely imaginary impedance resulting in the magnitude equalling the impedance but with a 90 degree phase shift.

2.1.3 Resonance Circuits

Resonance can occur in an electrical circuit just as it can occur in most natural materials. In signal and system analysis, resonance can occur "in any system that has a complex conjugate pair of poles [6]", or any system that has at least one inductor and capacitor. To illustrate a simple resonance circuit, the impedance of a simple RLC series circuit, illustrated in figure 2.3, is analyzed here. As in DC, the impedance of components connected in series can be added to determine the total impedance of the system. Therefore, the total impedance of an RLC series circuit would be

$$Z = R + jwL + \frac{1}{jwC} \tag{2.1}$$

$$= R + j\left(wL - \frac{1}{wC}\right) \tag{2.2}$$

From the equation above, there are several circumstances that are interesting. To determine the frequency at which the circuit has a minimum impedance, solve for the imaginary part equal to zero because the real part is not dependent on the frequency.

$$0 = wL - \frac{1}{wC} \tag{2.3}$$

$$\frac{1}{wC} = wL \tag{2.4}$$

$$\frac{1}{LC} = w^2 \tag{2.5}$$

$$w = \sqrt{\frac{1}{LC}} \tag{2.6}$$

As is well known in circuit analysis, this is the basic equation for the resonant frequency in a series RLC circuit. Calculation of the frequency where the system has the maximum impedance is not as straightforward because the system does not have a local maxima where the slope is zero and the second derivative of the impedance with respect to frequency is concave. If we look at the basic case where $L = 1$ and $C = 1$, the magnitude of the impedance becomes dependent on

$$\left(w - \frac{1}{w}\right) \tag{2.7}$$

The maxima therefore occur at zero and infinite frequency.

2.1.4 Multiple Resonance Circuits

Multiple resonance is a phenomenon that can occur with circuits involving many inductor and capacitor components arranged in loop structures. With a circuit that exhibits multiple resonance, more than one frequency exists that give local impedance maxima or minima. A circuit that exhibits multiple frequencies of zero impedance is illustrated in figure 2.4. Smith and Alley show this circuit along with an analysis showing the algebraic equation with quadratic terms created by the overall impedance equation that causes the multiple

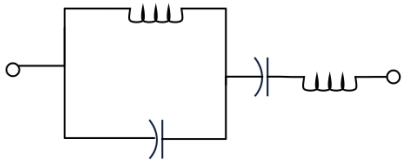


Figure 2.4: A circuit exhibiting multiple resonance frequencies

resonance frequencies [7]. Smith and Alley show the calculations required to obtain the overall impedance equation of

$$\|z\| = \frac{w^4 - w^2\left(\frac{1}{L_1C_1} + \frac{1}{L_2C_2} + \frac{1}{L_1C_2}\right) + \frac{1}{L_1C_1L_2C_2}}{\frac{w}{L_1}\left(w^2 - \frac{1}{L_2C_2}\right)} \tag{2.8}$$

which reduces to the form

$$\|z\| = \frac{(w^2 - w_1^2)(w^2 - w_3^2)}{\frac{w}{L_1}(w^2 - w_2^2)} \tag{2.9}$$

indicating two frequencies that can cause a zero in the numerator and an overall impedance of zero. Notice that this circuit cannot be decoupled with each section analyzed separately; it must be analyzed as one system.

2.2 Dimensionality Reduction

Modern network analyzers that measure the impedance of a system over a range of frequencies are capable of taking measurements from 10 MHz (10^6) to 1.05 THz (10^{12}) with over 20,000 sample points [8]. Analysis of all 20,000 points may be time consuming and unneeded because initial investigations show that the similarity of two impedance curves can be determined by their general shape, which can be represented with a lot less measurements at selected frequencies.

For analysis, the data used in this research will be represented as vectors of complex floating point values, for instance, a vector of 20,000 elements representing the real and

imaginary parts of the impedance at 20,000 frequencies. If correlation exists among the data variables, then one of several techniques can be used to reduce the dimensionality of a set of data while still retaining most of the information represented in the data. In this section, several documented techniques will be reviewed. Using knowledge of the data and an analysis of the loss of information, one may be able to reduce the data to less than 1/1000th of the original size, allowing much more efficient calculation of results, avoiding instability of solution results, and rendering more possibilities for application. In the subsections below, two methods of dimension reduction are reviewed: principal component analysis and peak binning.

2.2.1 Principal Component Analysis

Principal component analysis (PCA) identifies the principal components of the data that exhibit the most variability. With PCA, the data are represented with respect to a new ordered set of orthogonal bases that capture successively less variability in the data. In many cases, 90% of the variability in the data can be represented in the first few principal components.

One of the first publications on PCA was in Philosophical Magazine by Karl Pearson in 1901, where he describes his techniques by choosing a line to best represent a series of data points in a method where the sum of the error distance perpendicular to the best fit line is minimized [9]. Since 1901, PCA has been an integral part of multivariate analysis and may possibly be the most used data analysis techniques in dimensionality reduction. Several books are dedicated to the subject and explain many different applications of the same techniques with many great advantages [10, 11, 12].

One method of performing PCA is with singular value decomposition (SVD). With SVD, the data matrix of size $m \times n$ with rank r is factored into three matrices that have unique

properties.

$$X = U\Sigma V' \tag{2.10}$$

The V matrix is of size $n \times r$, and the columns of V are right singular vectors of X. The columns of V represent a set of basis where each column shows the directions onto which the sum of squares of the projection of the rows of X is of decreasing magnitude. The columns of V are orthonormal.

The U matrix is of size $m \times r$ and the columns of U are orthonormal, and are the left singular vectors of X.

The Σ matrix is a diagonal $r \times r$ matrix. The values in the Σ matrix are referred to as the singular values and are the positive square roots of the eigenvalues of both XX' and $X'X$. The singular values are in decreasing order and can be used to determine the real, or effective, rank of the original data matrix by looking at the number of non-zero singular values. To determine the effective rank, the ratio of each singular value to the maximum singular value is calculated and low ratios below a threshold are typically taken to be zero. The number of non-zero singular values above the threshold represent the 'effective rank'.

In figure 2.5, 1,000 two-dimensional points from the multivariate normal distribution are plotted. The SVD of the mean-centered data matrix was calculated, and both columns of the V matrix are shifted by the mean of the data and plotted in red. The vector of the first column of V is the longer red vector and shows the direction of most variance. The vector from the second column is orthogonal to the first and shows the direction of least variance.

The data used here have many traits that work well with PCA. The impedance spectra for the circuits used have several maxima and minima with steep slope transitions. This type of data, when analyzed with PCA, should produce a change of basis that exemplifies the variability in the maxima vs minima.

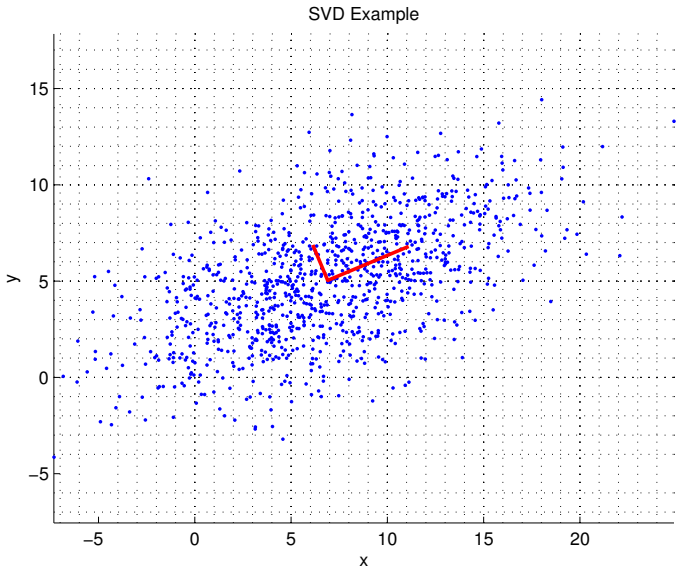


Figure 2.5: SVD of two-dimensional data from a correlated multivariate normal distribution.

2.2.2 Peak Analysis - Binning

This method comes from Puryear et al. where the data were binned and multiple peaks falling in one bin were combined [13]. This method may have problems if the bins are selected to be too large and combining peaks that should have remained separate or selecting bins that are too small and obscure the natural variability in the data peaks. The results of this research show the effect of bin size on the loss of data during reduction.

2.3 Spectral Similarity/Dissimilarity Metrics

Comparison of objects represented as data vectors is a well-researched and documented topic. The most popular methods that are pertinent to impedance spectra are reviewed here. This involves most of the metrics covered in McGill's research [14]. While studying the equations and background for each similarity measure, a geometric approach will also be applied to visualize the contours of points 'equally' distant from a specific point. Note that the shapes

of the contours depend upon the selected point, so this is only an illustration. This geometric approach is taken from the examples in Jones and Fumas [15].

Vector comparisons have been performed in mathematics for many years. The topic is heavily documented in fields ranging from computerized text searches [14] to speech recognition [16], comparing music [17] and mass spectrometry [18, 13, 19]. The most complete publication on the topic found was documented by Noreault, McGill, and Koll under a NSF grant in 1979 where a list of 67 similarity measures were reduced to 24 based on algebraic equivalence of statistical similarity [20]. Here, we examine the utility of the similarity measures on the data to be used in the proposed research.

The notation in table 2.1 and the following sections is similar to the notation used in Norealt [20] and McGill [14]. Two vectors, X and Y, of any dimension are compared. X_i refers to the i^th element of the X vector.

2.3.1 Inner Product

The inner product has some interesting geometric features that make it a very good basis for many of the equations used here. The inner product of a vector with itself is equivalent to the length of the vector squared. The inner product between two orthogonal vectors is equal to zero.

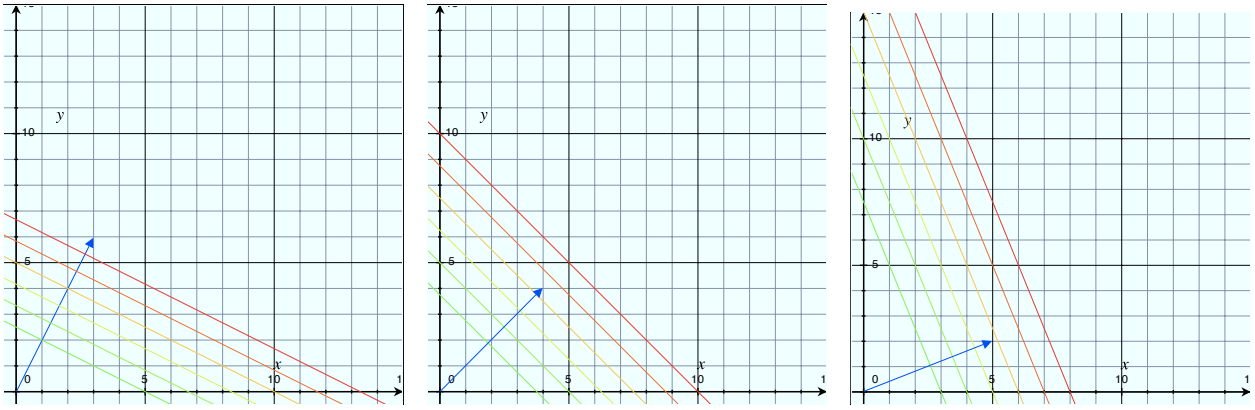
$$\sum X_i Y_i \tag{2.11}$$

Jones and Furnas geometrically illustrate four important properties of using an inner product for comparing two vectors. The inner product between a vector and another vector grows as the angle between the two vectors grows. If two vectors are being compared to a query vector and they both have the same angle of separation from the query vector but are of different lengths, the longer vector will have a larger inner product value [15].

One problem with the inner product as a similarity measure is that the value is unbounded. The inner product of two vectors can grow arbitrarily large based on the

Table 2.1: Listing of metrics analyzed.

Metric	Distance Equation
Inner Product	$\sum X_i Y_i$
Euclidean distance	$\sqrt{\sum (X_i - Y_i)^2}$
Mahalanobis distance	$\sqrt{(x-\mu)S^{-1}(x-\mu)'}$
Manhattan distance	$\sum \lvert X_i - Y_i \rvert$
Average	$\frac{1}{M} \sum X_i - Y_i$
Squared Chord	$\sum \left(\sqrt{X_i} - \sqrt{Y_i}\right)^2$
Canberra	$\sum \left(\frac{\lvert X_i - Y_i \rvert}{X_i + Y_i}\right)$
Coefficient of Divergence	$\frac{1}{M} \sum \left(\frac{X_i - Y_i}{X_i + Y_i}\right)^2$
Modified Boolean Correlation	$\frac{1}{M}(\sum X_i Y_i + \sum X_i' Y_i')$
Avg Weight of Shared Terms	$\frac{\sum (X_i + Y_i)\beta_i}{2N}$
Overlap	$\frac{\sum \min(X_i, Y_i)}{\min(\sum X_i, \sum Y_i)}$
Cosine	$\frac{\sum X_i Y_i}{(\sum X_i^2 \sum Y_i^2)^{\frac{1}{2}}}$
Similarity Index	$\sqrt{\frac{\sum \left(\frac{\lvert X_i - Y_i \rvert}{\min(X_i, Y_i)}\right)^2}{M}}$
Tanimoto's	$\frac{\sum X_i Y_i}{\sum X_i^2 + \sum Y_i^2 - \sum X_i Y_i}$



(a) Vector (3,6) (b) Vector (4,4) (c) Vector (5,2)

Figure 2.6: Contour graphs of equivalent inner product distances. All graphs show contours for distances of 15, 20, 25, 30, 35, and 40 with more similar contours in green.

length. If all values in the vectors are bounded to be positive, a lower bound on the inner product exists at 0; but, without the bound to be positive values, the inner product could be unbounded in the positive and negative direction. This problem will be solved later with variations better suited for similarity comparison.

In two dimensions, the inner product is evaluated as

$$X_1 Y_1 + X_2 Y_2 \tag{2.12}$$

and the inner product distance d to a location (X_1, X_2) is evaluated as

$$d = X_1 Y_1 + X_2 Y_2 \tag{2.13}$$

$$X_2 = -\frac{Y_1}{Y_2} X_1 + \frac{d}{Y_2} \tag{2.14}$$

which is the equation for a line with a slope of -$\frac{Y_1}{Y_2}$ and a y-intercept at $\frac{d}{Y_2}$. The contour lines of equal distance to locations (3,6), (4,4), and (5,2) are shown in figure 2.6.

2.3.2 Euclidean Distance

Euclidean distance is the standard metric used in most distance measurements because in $\Re^2$, the distance can be measured with any standard ruler.

$$\sqrt{\sum (X_i - Y_i)^2} \tag{2.15}$$

If the equation for Euclidean distance is evaluated to find all locations that have a distance of d to the location (4,4), the following equation is formed

$$d = \sqrt{(X_1 - 4)^2 + (X_2 - 4)^2} \tag{2.16}$$

$$d^2 = (X_1 - 4)^2 + (X_2 - 4)^2 \tag{2.17}$$

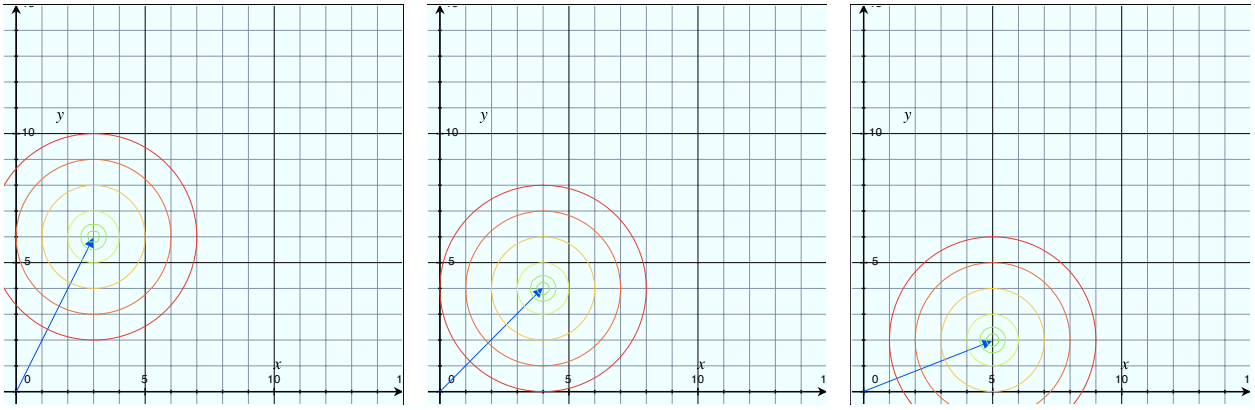
To understand what the equation above looks like graphically, the equation for a circle is reviewed below. In the equation below, the circle is centered at location (a, b) and has radius of r.

$$(x - a)^2 + (y - b)^2 = r^2 \tag{2.18}$$

Contour graphs for the euclidean distance to locations (3,6), (4,4), and (5,2) are shown in figure 2.7.

2.3.3 Mahalanobis Distance

The Mahalanobis distance was first introduced by Prasanta Chandra Mahalanobis in 1936 with his publication, "On the generalized distance in statistics" [21]. The metric is very similar to the Euclidean distance with the modification that it takes into account the density and dispersion of known members in a group. This metric is different from many of the metrics listed here in that it will calculate the distance to the center of a cluster of points based on the dispersion of existing members in the group.



(a) Point (3,6) (b) Point (4,4) (c) Point (5,2)

Figure 2.7: Contour graphs of equivalent Euclidean distances. All graphs show contours for distances of .25, .5, 1, 2, 3, and 4 from the given point, with more similar contours in green.

The Mahalanobis distance equation uses the mean (μ) and covariance matrix (S) of the current group members to calculate the distance from the mean to a given point (x).

$$\sqrt{(x-\mu)S^{-1}(x-\mu)'} \tag{2.19}$$

where each element of the covariance matrix, S, is the variance between samples of feature i and of feature j defined as

$$S(i,j) = \frac{(X_i - \bar{X}_i)'(X_j - \bar{X}_j)}{n-1} \tag{2.20}$$

where X_i is a vector of the i^{th} column/feature values of the group members, $\bar{X}_i$ is the mean of the i^{th} column/feature values of the group members, X_j is a vector of the j^{th} column/feature values of the group members, $\bar{X}_j$ is the mean of the j^{th} column/feature values of the group members, and n is the number of group members. The covariance matrix can also be calculated as $X'X$.

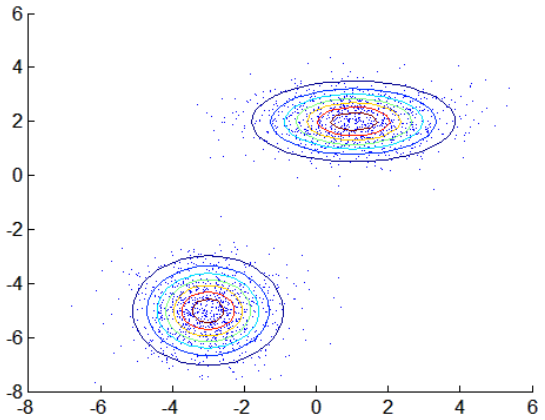


Figure 2.8: An example distribution with Mahalanobis distance contours from the mean value of the group, (-3,-5) and (1,2). [23].

Figure 2.8 is a contour graph for an example group of two dimensional data. Notice how the contours follow the natural dispersion and density of the members of the group.

Mahalanobis Distance Calculation using SVD

The Mahalanobis distance scores can also be calculated using SVD [22]. The proof is shown in equation 2.21. In equation 2.21, X_c is a matrix with each row representing a sample belonging to group X with the mean of group X subtracted. d is the mahalanobis distance to the mean of group X. Y_c is a matrix with each row representing a mean-centered point. Note that the covariance matrix S is equivalent to $X_c'X_c/(n-1)$, where n is the number of group members. Also note that section 2.2.1 describes how SVD decomposes a matrix into

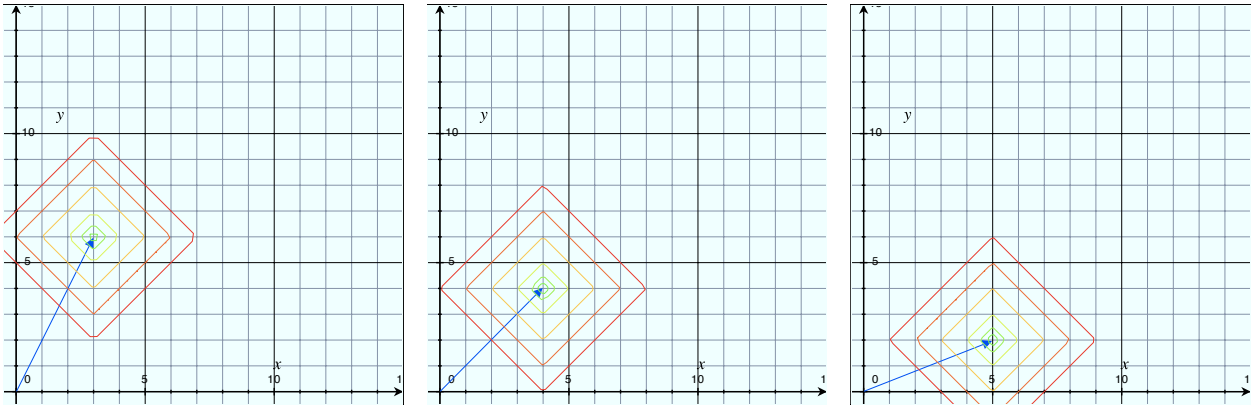
three matrices, $U\Sigma V'$, with unique properties.

$$
\begin{aligned}
d^2 &= Y_c S^{-1} Y_c' && (2.21)\\
&= Y_c \left(\frac{X_c' X_c}{n-1} \right)^{-1} Y_c' && (2.22)\\
&= Y_c \left(\frac{(U\Sigma V')'(U\Sigma V')}{n-1} \right)^{-1} Y_c' && (2.23)\\
&= Y_c \left(\frac{V\Sigma U' U\Sigma V'}{n-1} \right)^{-1} Y_c' && (2.24)\\
&= Y_c \left(\frac{V\Sigma\Sigma V'}{n-1} \right)^{-1} Y_c' && (2.25)\\
&= Y_c \left(\frac{V\Sigma^2 V'}{n-1} \right)^{-1} Y_c' && (2.26)\\
&= Y_c \left(\frac{n-1}{V\Sigma^2 V'} \right) Y_c' && (2.27)\\
&= Y_c V \Sigma^{-2} V' Y_c' (n-1) && (2.28)\\
& && (2.29)
\end{aligned}
$$

d^2 is the sum of the squares of normalized distances of a point to its mean of the group, normalized by the respective standard deviation of the spread of the points along the V vectors (or new basis vectors).

2.3.4 Manhattan Distance

The Manhattan distance measure is often called the city block, or taxicab, distance because it measures the distance between points in space if the path of travel is only able to follow directions parallel to the coordinate space, as a taxicab driver in Manhattan, NY would have to do when traveling between two points in the city where the streets are only North-South



(a) Point (3,6) (b) Point (4,4) (c) Point (5,2)

Figure 2.9: Contour graphs of equivalent Manhattan distances. All graphs show contours for distances of .25, .5, 1, 2, 3, and 4 with more similar contours in green.

or East-West. The Manhattan distance is defined as

$$\sum |X_i - Y_i| \tag{2.30}$$

where X and Y are two given points, and X_i refers to the i^{th} coordinate of the point X.

When the equation is evaluated in two dimensions to find all locations that have a distance of d to the location (4,4), the following equation is formed

$$d = |X_1 - 4| + |X_2 - 4| \tag{2.31}$$

Some intuitive thinking and evaluation of the equation above gives the contours in figure 2.9. The advantages and disadvantages of using the Manhattan distance are similar to those of the Euclidean distance with the exception that similar vectors form a diamond shape around the query point.

2.3.5 Average Distance

The average distance is defined as

$$\frac{1}{M}\sum X_i - Y_i \tag{2.32}$$

where M is the number of coordinates in X. Y contains the same number of points as X.

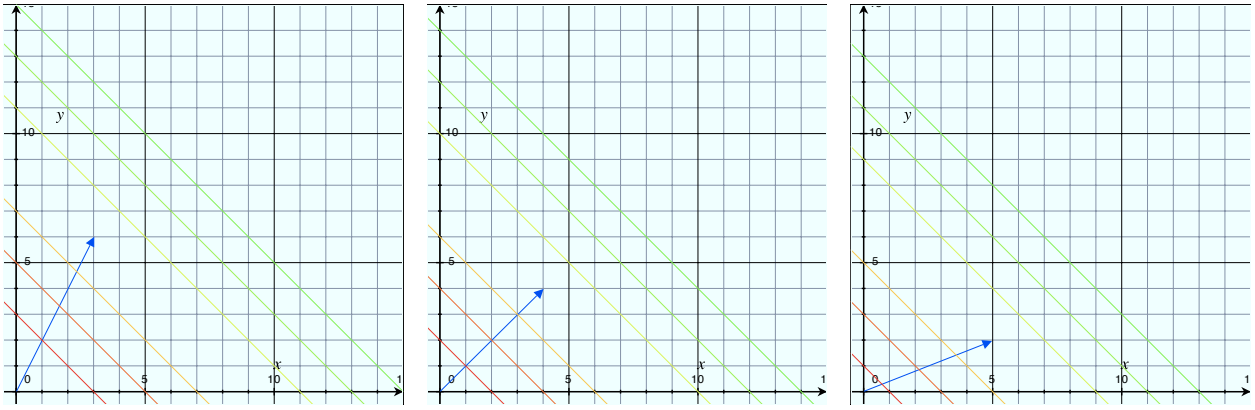
A look at this equation in two dimensions reveals the contour structure that gives us similar values. We will determine all points with a distance value of d to the location (4,4).

$$\begin{aligned} d &= \frac{1}{2}\left[(4 - Y_1) + (4 - Y_2)\right] && (2.33)\\ 2d &= (4 - Y_1) + (4 - Y_2) && (2.34)\\ 2d &= 8 - Y_1 - Y_2 && (2.35)\\ Y_2 &= -Y_1 + (8 - 2d) && (2.36) \end{aligned}$$

Therefore, the following vectors will have the same average distance to the vector (4,4): (1,3), (2,2), (3,1), (4,0), (5,-1)... creating a straight contour line at a negative 45° angle to the query point. The contour graph in figure 2.10 illustrates several distances to the locations (3,6), (4,4), and (5,2).

2.3.6 Squared Chord Distance

The squared chord distance has been actively used in palynology for pollen assemblage comparison based on the work by Gavin et al. and Overpeck et al. in comparing distance metrics with respect to pollen assemblages [24] [25]. The equation only allows comparisons of vectors with positive elements and produces a shape similar to Euclidean distance, but



(a) Point (3,6) (b) Point (4,4) (c) Point (5,2)

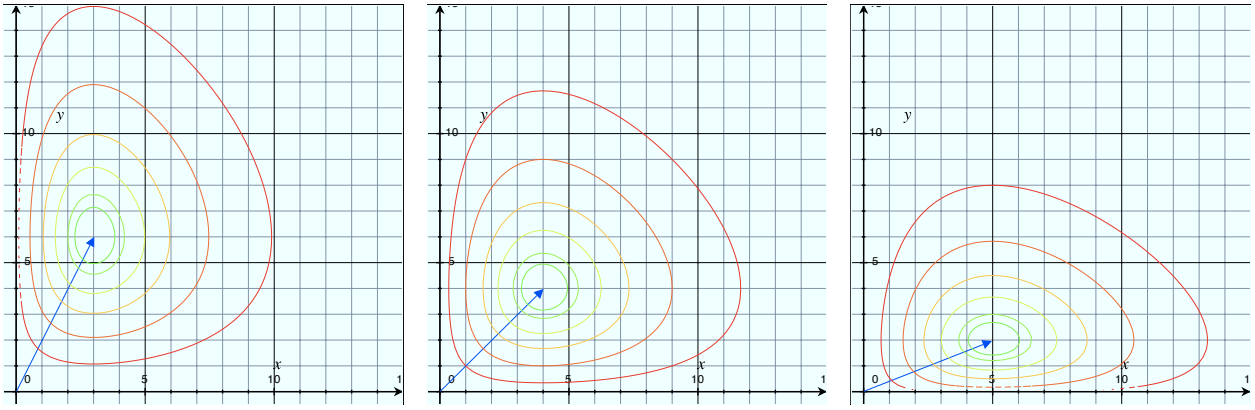
Figure 2.10: Contour graphs of equivalent Average distances. All graphs show contours for distances of -3, -2, -1, 1, 2, and 3 with more similar contours in green.

stretched in the direction of the axis.

$$\sum \left(\sqrt{X_i} - \sqrt{Y_i}\right)^2 \tag{2.37}$$

A look at this equation in two dimensions reveals the contour structure that gives us similar values. We will determine all points with a similarity distance value of d to the location (4,4).

$$\begin{aligned}
d &= \left(\sqrt{4} - \sqrt{Y_1}\right)^2 + \left(\sqrt{4} - \sqrt{Y_2}\right)^2 && (2.38)\\
d &= \left(2 - \sqrt{Y_1}\right)^2 + \left(2 - \sqrt{Y_2}\right)^2 && (2.39)\\
\left(2 - \sqrt{Y_2}\right)^2 &= d - \left(2 - \sqrt{Y_1}\right)^2 && (2.40)\\
2 - \sqrt{Y_2} &= \pm\sqrt{d - \left(2 - \sqrt{Y_1}\right)^2} && (2.41)\\
\sqrt{Y_2} &= 2 \pm \sqrt{d - \left(2 - \sqrt{Y_1}\right)^2} && (2.42)\\
Y_2 &= \left(2 \pm \sqrt{d - \left(2 - \sqrt{Y_1}\right)^2}\right)^2 && (2.43)
\end{aligned}$$



(a) Point (3,6) (b) Point (4,4) (c) Point (5,2)

Figure 2.11: Contour graphs of equivalent Squared Chord distances. All graphs show contours for distances of .05, .1, .25, .5, 1, and 2 with more similar contours in green.

Graphing the above equation for several point values yields the contour structure in figure 2.11.

2.3.7 Canberra Distance

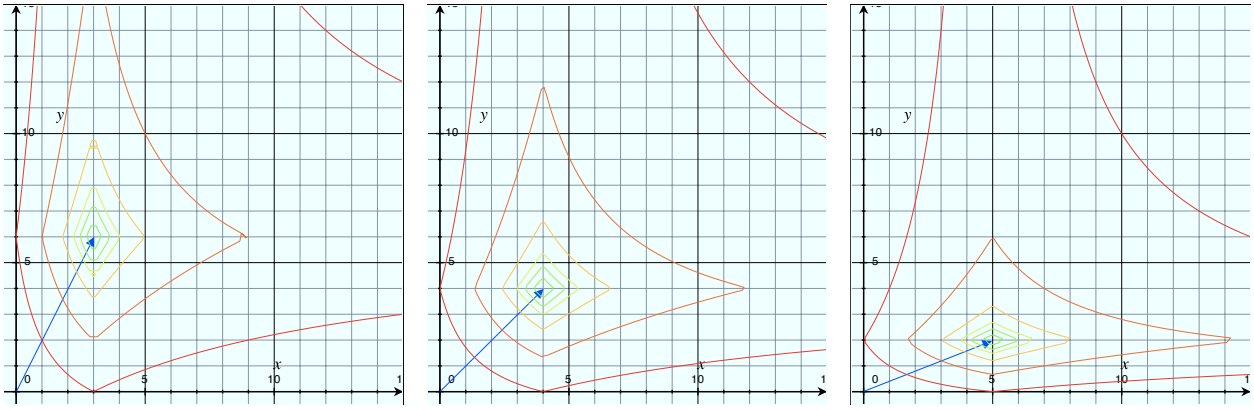
The canberra distance was first published in 1966 [26] and then refined in 1967 by the same authors, Lance and Williams.

$$\sum \left(\frac{|X_i - Y_i|}{X_i + Y_i} \right) \tag{2.44}$$

A look at this equation in two dimensions reveals the contour structure that gives us similar values. We will determine all points with a similarity distance value of d to the location (4,4).

$$d = \left(\frac{|X_1 - 4|}{X_1 + 4} \right) + \left(\frac{|X_2 - 4|}{X_2 + 4} \right) \tag{2.45}$$

Graphing the above equation for several point values yields the contour structure in figure 2.12.



(a) Point (3,6) (b) Point (4,4) (c) Point (5,2)

Figure 2.12: Contour graphs of equivalent Canberra distances. All graphs show contours for distances of .05, .1, .15, .25, .5, and 1 with more similar contours in green.

2.3.8 Coefficient of Divergence

The coefficient of divergence was introduced by Sneath and Sokal [27], studied by McGill [14], and defined as

$$\frac{1}{M}\sum\left(\frac{X_i - Y_i}{X_i + Y_i}\right)^2 \tag{2.46}$$

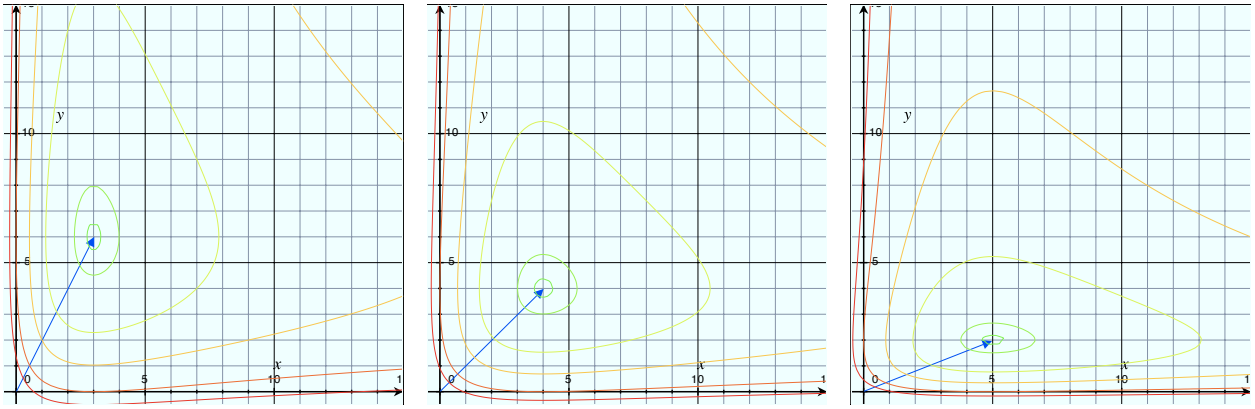
where M is the number of coordinates in X. Y contains the same number of points as X.

In two dimensions, this equation is expanded to

$$\frac{1}{2}\left[\left(\frac{X_1 - Y_1}{X_1 + Y_1}\right)^2 + \left(\frac{X_2 - Y_2}{X_2 + Y_2}\right)^2\right] \tag{2.47}$$

where X is a vector and is compared to Y, which is also a vector. The contour graphs of similar distances d to the locations (3,6), (4,4), and (5,2) is shown in figure 2.13.

$$d = \frac{1}{2}\left[\left(\frac{4 - Y_1}{4 + Y_1}\right)^2 + \left(\frac{4 - Y_2}{4 + Y_2}\right)^2\right] \tag{2.48}$$



(a) Point (3,6) (b) Point (4,4) (c) Point (5,2)

Figure 2.13: Contour graphs of equivalent coefficient of divergence distances. All graphs show contours for distances of .001, .01, .1, .25, .5, and .7 with more similar contours in green.

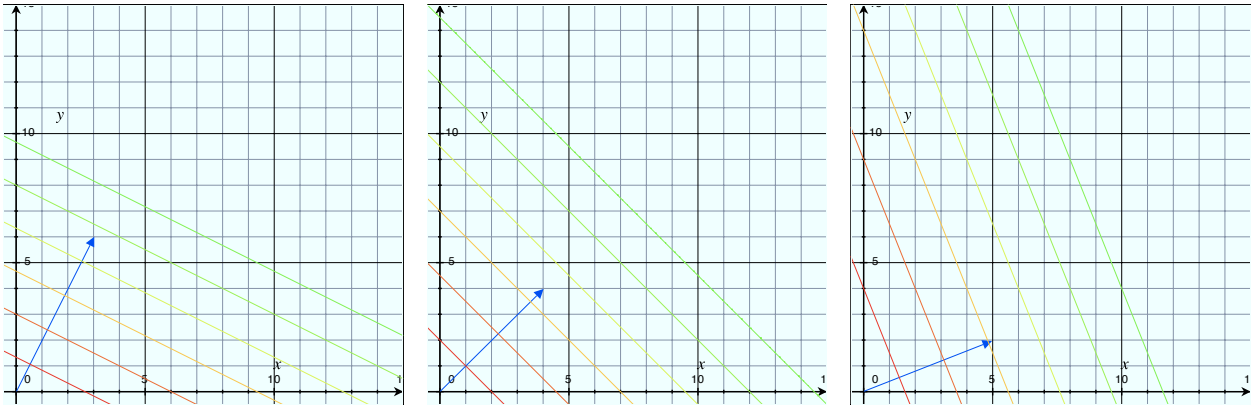
2.3.9 Modified Boolean Correlation

The modified boolean correlation was introduced in Sager [28] and is almost the same as the arithmetic mean of the two vectors. Its modified from the arithmetic mean to include another term that is a value of zero or one depending on if the terms in the vector are positive or negative (if either term is negative, X_i and Y_i equal zero, X_i and Y_i equal one otherwise). This other term becomes more and more negligible as the elements of the two vectors are larger and larger.

$$\frac{1}{M}(\sum X_i Y_i + \sum X_i' Y_i') \tag{2.49}$$

In our case, all terms will be positive and therefore the second term will always be one. If we expand the equation to show equal distances in two dimensions from the location (4,4), the following relationship is formed.

$$d = \frac{1}{2}(4X_1 + 4X_2 + 2) \tag{2.50}$$



(a) Point (3,6) (b) Point (4,4) (c) Point (5,2)

Figure 2.14: Contour graphs of equivalent modified boolean correlation distances. All graphs show contours for distances of 5, 10, 15, 20, 25, and 30 with more similar contours in green.

Rearranging the above equation, it can be shown that this is an equation for a line with a $-\frac{Y_2}{Y_1}$ slope and y-intercept at $\frac{2(d-1)}{Y_1}$.

$$d = \frac{1}{2}(X_1Y_1 + X_2Y_2 + 2) \tag{2.51}$$

$$2d = X_1Y_1 + X_2Y_2 + 2 \tag{2.52}$$

$$X_1Y_1 = -X_2Y_2 + 2(d-1) \tag{2.53}$$

$$X_1 = -\frac{Y_2}{Y_1}X_2 + \frac{2(d-1)}{Y_1} \tag{2.54}$$

$$\tag{2.55}$$

The contours of similar distances are graphed in figure 2.14 and reveal a structure very similar to the average distance measure but shifted with a different slope and y-intercept.

2.3.10 Average Weight of Shared Terms

The average weight of shared terms metric was introduced in Reitsma and Sagalyn's report in 1967 [29] and is equivalent to the average value of all of the terms in both vectors, excluding

any dimensions with negative values.

$$\frac{\sum (X_i + Y_i)\beta_i}{2N} \tag{2.56}$$

$$\beta_i = \begin{cases} 1 & : \quad X_i > 0 \, and \, Y_i > 0 \\ 0 & : \quad Otherwise \end{cases}$$

The data used in this research always has positive terms, causing the β_i term to fall off and have no effect. In two dimensions, the equation is expanded to be the following.

$$\frac{1}{4}(X_1 + Y_1 + X_2 + Y_2) \tag{2.57}$$

An analysis of this equation to reveal contours of equal distance d to the vector (X_1, Y_1) reveals another distance metric that simplifies to the equation of a line with a -1 slope and y-intercept at $4d - Y_1 - Y_2$, as shown in figure 2.15.

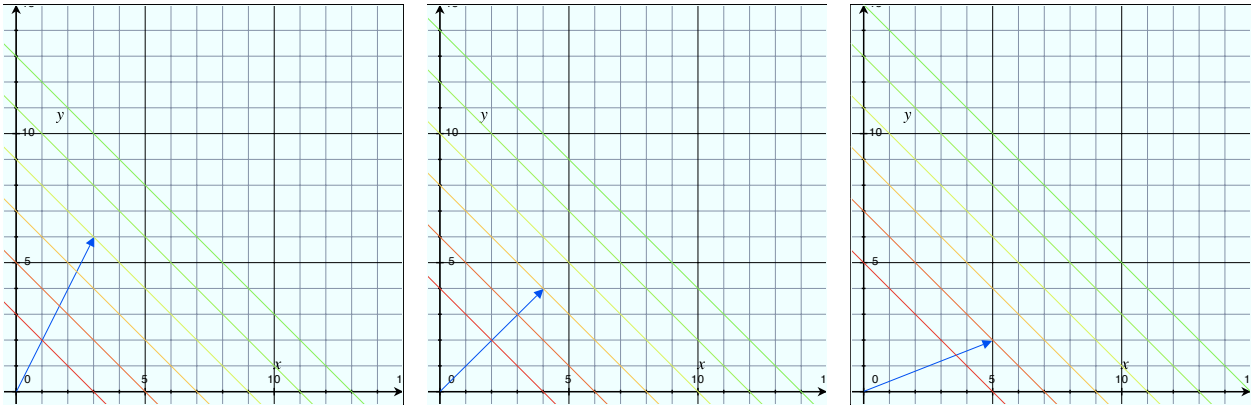
$$d = \frac{1}{4}(X_1 + Y_1 + X_2 + Y_2) \tag{2.58}$$

$$4d = X_1 + Y_1 + X_2 + Y_2 \tag{2.59}$$

$$X_1 = -X_2 + (4d - Y_1 - Y_2) \tag{2.60}$$

2.3.11 Overlap

$$\frac{\sum \min(X_i, Y_i)}{\min(\sum X_i, \sum Y_i)} \tag{2.61}$$



(a) Point (3,6) (b) Point (4,4) (c) Point (5,2)

Figure 2.15: Contour graphs of equivalent average weights of shared terms. All graphs show contours for distances of 3, 3.5, 4, 4.5, 5, and 5.5 with more similar contours in green.

Expanding the equation in two dimensions to find contours of equal distance d from a vector (4,4) reveals the following equation.

$$d = \frac{\min(X_1, 4) + \min(X_2, 4)}{\min(X_1 + X_2, 8)} \tag{2.62}$$

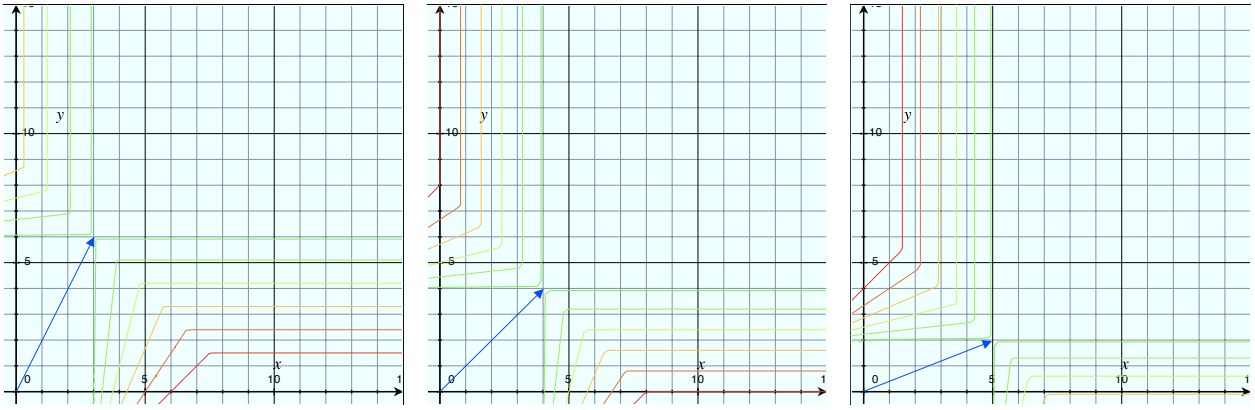
This equation is difficult to graph and doesn't give much meaning. Instead, we will evaluate the function for several different scenarios. If $X_1 < 4$ && $X_2 > 4$, the equation evaluates to the following and the distance is always 1.

$$d = \frac{4 + 4}{8} \tag{2.63}$$

If $X_1 \gg 4$ && $X_2 < 4$ or $X_1 < 4$ && $X_2 \gg 4$, the distance evaluates to the following and will be in the range of .5-1.

$$d = \frac{4 + X_2}{8} \tag{2.64}$$

$$d = \frac{X_1 + 4}{8} \tag{2.65}$$



(a) Point (3,6) (b) Point (4,4) (c) Point (5,2)

Figure 2.16: Contour graphs of equivalent overlap distances. All graphs show contours for distances of .99, .9, .8, .7, .6, and .5 with more similar contours in green.

If $X_1 < 4$ and $X_2 < 4$ the distance evaluates to the following and will always be 1.

$$d = \frac{X_1 + X_2}{X_1 + X_2} \tag{2.66}$$

If we look at the case where $X_1 < 4$ and $X_2 = 4$ or close to it, it becomes obvious that this metric is bounded by 0.5-1.

$$d = \frac{[0 - 4] + 4}{[4 - 8]} = \frac{[4 - 8]}{[4 - 8]} \tag{2.67}$$

Graphing several contours of distance 0.5-1 to the points (3,6), (4,4), and (5,2) shows the structure in figure 2.16. Notice how any points in the upper right or lower left quadrants result in a distance of 1.

2.3.12 Cosine Similarity

The cosine similarity distance metric is equivalent to the cosine of the angle between two vectors and also the inner product divided by the norm/length of the inner product.

$$\frac{\sum X_i Y_i}{(\sum X_i^2 \sum Y_i^2)^{\frac{1}{2}}} \tag{2.68}$$

This metric has been used in many areas due to the easy and intuitive interpretation of the similarity. The metric is also bounded on the interval zero to one with a value of zero indicating the vectors are perpendicular and a value of one indicating the vectors are collinear. Noreault and McGill cite Torgerson's 1958 book [30] as the origin of the metric [20]; however many linear algebra texts show the proof of this metric [31]. This metric has the benefit of being scale independent. A contour graph showing the general structure of the contours is shown in figure 2.17 and can be applied to locating similar distances to a point at locations (3,6), (4,4), and (5,2).

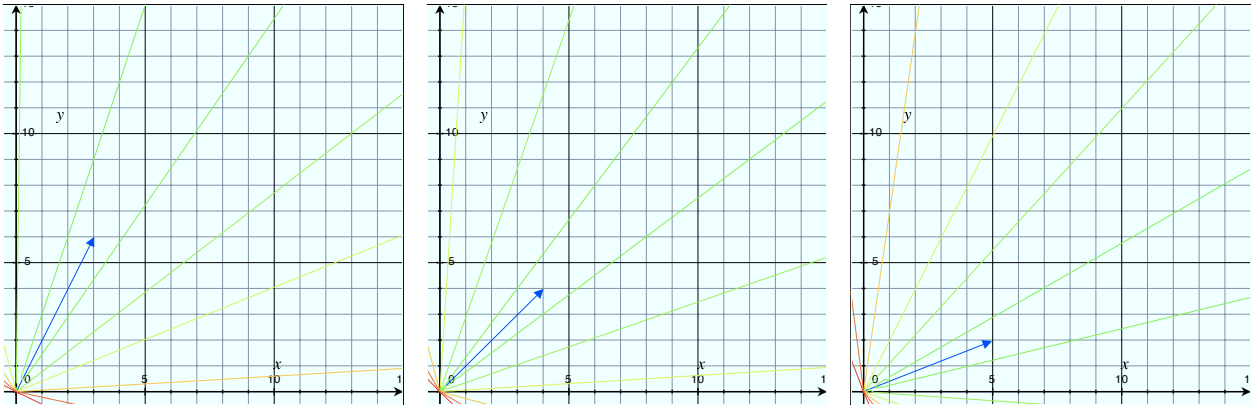
$$d = \frac{X_1 Y_1 + X_2 Y_2}{\sqrt{(X_1^2 + X_2^2)(Y_1^2 + Y_2^2)}} \tag{2.69}$$

2.3.13 Similarity Index

The similarity index is a metric introduced by Lay, Gross, Zwinselman, and Nibbering in 1983 [32] and later refined by Wan, Vidavsky, and Gross in 2002 [33]. The equation is defined as

$$\sqrt{\frac{\sum \left(\frac{|X_i - Y_i|}{\min(X_i, Y_i)}\right)^2}{M}} \tag{2.70}$$

where M is the number of coordinates in X. Y contains the same number of points as X.



(a) Vector (3,6) (b) Vector (4,4) (c) Vector (5,2)

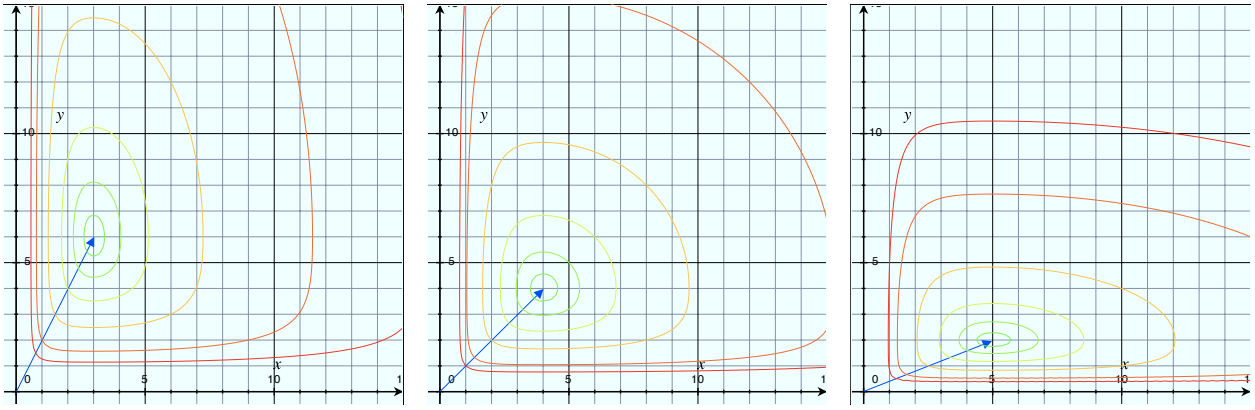
Figure 2.17: Contour graphs of equivalent cosine distances. All graphs show contours for distances of .99, .9, .75, .5, .25, and .01 with more similar contours in green.

The metric was compared to the cosine similarity in 2002 by Wan et. al. [33] and determined to be less effective, but Wan's study was determined to be invalid by Alfassi 2003 [19] because of the lack of scaling performed on the similarity index. The similarity index is an unbounded metric where a value of 0 indicates an exact match and the value increases as the two vectors become less and less similar. The contour graph in figure 2.18 illustrates the contours to locations (3,6), (4,4), and (5,2).

2.3.14 Tanimoto's

The Tanimoto coefficient is an extension of the cosine similarity distance that is documented in Tanimoto's internal IBM memos [34] and Rogers [35]. The calculation is equivalent to the Jaccard coefficient [36] when all elements of the vectors are binary values.

$$\frac{\sum X_i Y_i}{\sum X_i^2 + \sum Y_i^2 - \sum X_i Y_i} \tag{2.71}$$



(a) Point (3,6) (b) Point (4,4) (c) Point (5,2)

Figure 2.18: Contour graphs of equivalent similarity index distances. All graphs show contours for distances of .1, .25, .5, 1, 2, and 3 with more similar contours in green.

In two dimensions, the equation becomes

$$d = \frac{X_1Y_1 + X_2Y_2}{X_1^2 + X_2^2 + Y_1^2 + Y_2^2 - X_1Y_1 - X_2Y_2} \tag{2.72}$$

$$d = \frac{X_1Y_1 + X_2Y_2}{(X_1^2 - X_1Y_1 + Y_1^2) + (X_2^2 - X_2Y_2 + Y_2^2)} \tag{2.73}$$

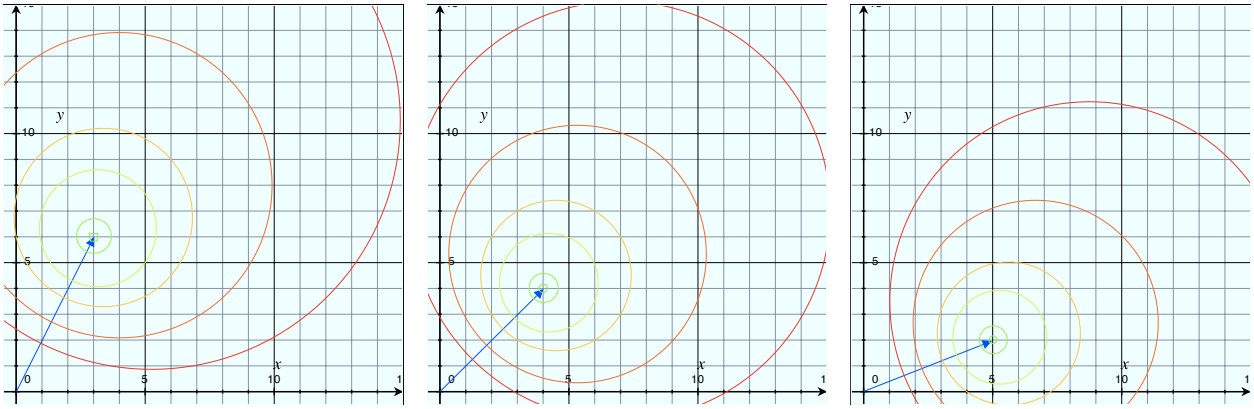
Figure 2.19 shows contours to the locations (3,6), (4,4), and (5,2).

2.3.15 Metric Use in Complex Vector Spaces

The data used in this research are in complex vector spaces, causing a slight modification of the similarity/dissimilarity metrics. In most cases, the inner product is replaced with the Hermitian product. As an example, Scharnhorst illustrates how the cosine similarity is used in complex vector spaces with the following two equations [37].

$$cos\theta(a,b) = \frac{(a,b)_R}{|a||b|} \tag{2.74}$$

$$cos\theta_C(a,b) = \frac{(a,b)_C}{|a||b|} \tag{2.75}$$



(a) Point (3,6) (b) Point (4,4) (c) Point (5,2)

Figure 2.19: Contour graphs of equivalent Tanimoto's distances. All graphs show contours for distances of .999, .99, .9, .8, .6, and .4 with more similar contours in green.

where $(a,b)_R$ is the scalar product and $(a,b)_C$ is the Hermitian product.

$$(a,b)_R = \sum_{k=1}^{m} a_k b_k \tag{2.76}$$

$$(a,b)_C = \sum_{k=1}^{n} \bar{a}_k b_k \tag{2.77}$$

where a and b in equation 2.77 are complex vectors and $\bar{a}_k$ denotes the complex conjugate of a_k.

2.4 Summary

This chapter has explained the background material and previously published research required to implement the model analysis, dimension reduction, and similarity equations on the impedance spectral data for this thesis. Basic impedance characteristics of single and multiple resonance circuits were explained. Two dimension reduction techniques were

detailed: principal component analysis and binned peak analysis. Fourteen similarity/dissimilarity metrics were explained: inner product, Euclidean, Mahalanobis, Manhattan, average, squared chord, Canberra, coefficient of divergence, modified boolean correlation, average weight of shared terms, overlap, cosine, similarity index, and Tanimoto's. These techniques have been adapted to several fields other than the field that they originated in. Several of those applications are detailed in chapter 3.

Chapter 3

Existing Applications

Research has been performed in many related fields that are incorporated into the methodology used here. The basic components of this research involve applying a comparison of two data vectors using a method that fits the data model and show how the samples are related. Vector comparison has been performed in many seemingly unrelated fields on different types of data. Several fields have modified the original methods to allow the experimenters to work with the data being analyzed.

3.1 Mass Spectrometry

The data used in this research have many characteristics that match those of mass spectrometry. In mass spectrometry, an unknown compound is analyzed by looking at the mass to charge (m/z) ratio of the individual components creating an output graph similar to our data where peaks exist at very specific m/z values. These output graphs can then be represented as a vector of values showing the intensity at every m/z value and compared to other compounds that have been analyzed [13]. In some methods, the spectra are reduced to a binary vector indicating if a peak exists at a specific ratio or does not [38]. The extent

of comparison has even been extended to develop a database of compound values where unknowns can be compared quickly and identified as compounds that exist in the database [39]. Many of the methods for comparison have been included in this research.

3.2 Music and Speech Recognition

Music and sound in almost all cases is digitized by analyzing the frequency and amplitude of the incoming sound waves over time. The resulting data consist of a vector of intensity values with a time constant spacing that can be displayed as a continuous waveform. The comparison of two sounds has been a tough problem to solve for many researchers. Two people could say the same phrase and the resulting digitized waveform could be drastically different but with distinct characteristics that give them a related strength [40]. Work has been performed in the music industry to quickly and effectively compare two sounds to give a measure of similarity between the two digitized waveforms [17].

3.3 Medical

Biological impedance tomography has been used to investigate and map the tissue composition of organisms within the medical field for many years. Blad and Baldetorp show how the impedance spectra of a tissue sample can be used to quickly identify the existence of cancerous tissue by analyzing the characteristic frequencies. They show that tumorous tissue exhibit a larger permittivity and conductivity than normal tissues and analyze the complex impedance of the tissue in the frequency range of 1.5–700 kHz. Their preliminary results "show that this method can be extended to a new application for detection of tumour tissue by electrical impedance tomography" [41].

3.4 Document and Text Searching

Words, phrases, sentences, or entire documents can be compared and evaluated for a degree of similarity using many of the techniques in this research. A word could be represented as a vector of ASCII numerical values or as a 26 element vector with the frequency of each letter. Documents can be represented as a multi-dimensional vector, the size of a dictionary, with each element representing the frequency of the existence of the dictionary word in the document. Martinez and Marinez illustrate an example of calculating the cosine similarity between several book titles using binary vectors illustrating if a word is present or not [42]. They also perform a comparison after using SVD to reduce the dimensions of the book vectors. Extensive research has also been conducted to maximize the efficiency of the search to perform the document searching quickly and in parallel [14].

3.5 Numerical Taxonomy

The classification of plants and animals into categories, or taxa, with other plants or animals that share like characteristics can be modeled as a vector of traits with the most similar beings having similar vectors. The techniques used in biological classification systems have been extended into many fields and are the basis of many of the techniques used here [43].

3.6 Hyperspectral Imagery

Van de Meer explains how similarity measures can be used to compare spectra acquired through remote sensing operations to determine the surface composition and properties. He uses several spectral measures in the comparison of the imagery data to known samples including spectral angle, Euclidean distance, and spectral correlation [44]. Some of these comparison techniques will be used in this research.

3.7 Summary

This chapter has shown many of the existing fields of research that have used techniques similar to the methods in this research. Chemists working with mass spectrometry have used similarity coefficients to compare the mass to charge ratios of compounds. Music and speech recognition have analyzed the sound waveforms for similarities between sounds. The medical industry has been using impedance tomography to detect cancerous tissue using the impedance spectra output of tissue samples. Biologists have compared organism traits using similarity coefficients in numerical taxonomy. Geologists have analyzed spectral data in the analysis of hyperspectral imagery to determine ground composition from remote sensing. In this research, the techniques in existing applications will be combined to form methods of automatically comparing electrical impedance spectra.

Chapter 4

Approach/Methods

4.1 Overview

The approach to solution will follow a simple four step process to determine the optimal steps in comparing impedance spectra.

1. The measured data are fit to a circuit model to form an estimated transfer function and circuit parameters.

2. Reduce the dimension of the impedance data with retention of important characteristics of the samples.

3. Use different similarity/dissimilarity metrics to determine which perform more correct classifications.

4. Determine if a threshold exists to determine when a sample does not belong to any of the current categories.

Each step may have its own individual analysis, results, and conclusion.

4.2 Research Questions

- Does a circuit model exist that simulates the data and does analysis reveal any characteristics important to classification?
- Does a reduction of data cause degradation in the classification of impedance spectra?
- What methods of data reduction are most suitable for this type of data?
- What similarity metrics are most suitable for this type of data?
- Do any of the similarity metrics correctly classify more samples than using the Mahalanobis distance?
- Is it possible to determine when an unknown sample does not belong to any of the known categories?

4.3 Available Data

Experiments conducted for this research will make use of impedance data from 1024 devices originating from 13 separate categories. The data were acquired using an Agilent network analyzer with 1601 sample points evenly spaced throughout the acquisition frequency range. The data were then scaled to a range of 1-1.6 Khz with a step impedance of 1 ohm. Analyzing the output graphs, shown below, several characteristics are desirable that distinguish each of the separate categories. The group membership of the devices was retained during acquisition and the characteristics of the data from each device group is detailed in these subsections. A plot of the magnitudes of all 13 groups is shown in figure 4.14 and the phase is shown in figure 4.15.

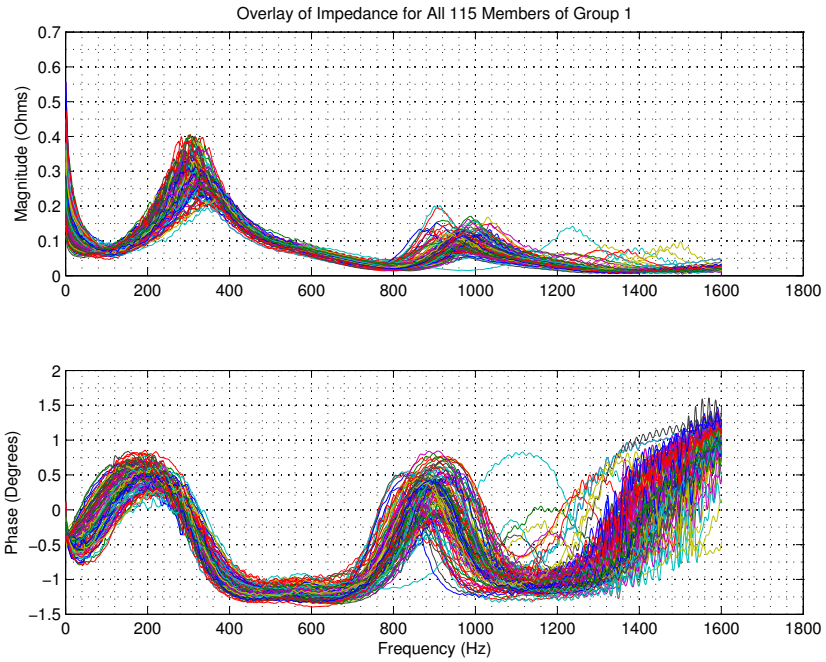


Figure 4.1: Impedance spectra from 115 members of group 1

4.3.1 Group 1

Group 1 spectra, shown in figure 4.1, consists of 115 samples and shows two well-defined peaks around frequencies of about 300Hz and 900Hz for the magnitude of the data. There are other frequencies that may have importance and will be investigated in the model analysis. There is also a high level of noise and variance in the higher frequencies. The spectra of all members of group 1 are overlaid in figure 4.1.

4.3.2 Group 2

Group 2 spectra, shown in figure 4.2, consists of 71 samples and shows three well-defined peaks around frequencies of about 175Hz, 550Hz, and 900Hz for the magnitude of the data. There is also a high level of noise and variance in the higher frequencies. The spectra of all members of group 2 are overlaid in figure 4.2.

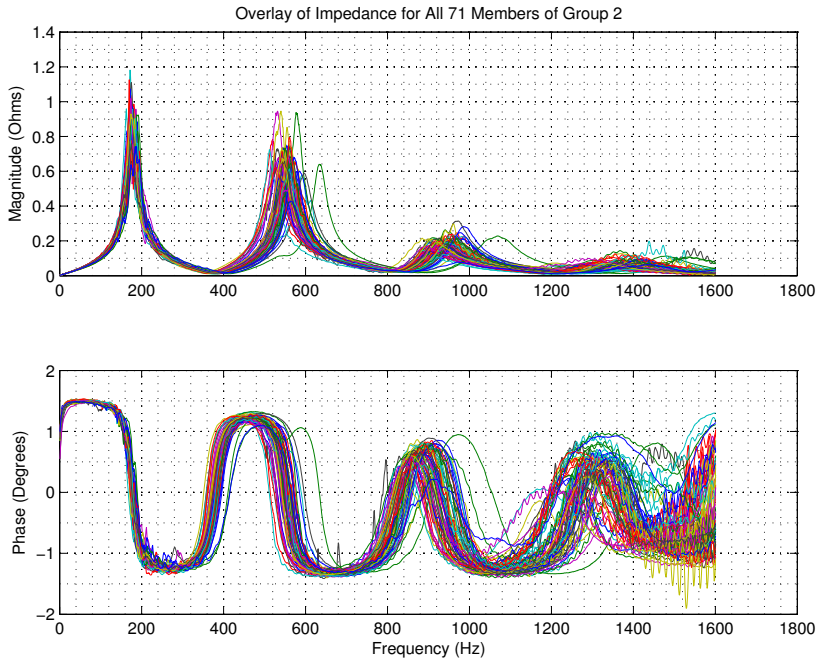


Figure 4.2: Impedance spectra from 71 members of group 2

4.3.3 Group 3

Group 3 spectra, shown in figure 4.3, consists of 52 samples and shows two peaks around frequencies of about 300Hz and 900Hz. There are other frequencies that may have importance and will be investigated in the model analysis. There is also a high level of noise and variance in the higher frequencies. The characteristics of this groups seem very similar to that of group one; however, initial classification results show that they can be effectively distinguished from each other. The spectra of all members of group 3 are overlaid in figure 4.3.

4.3.4 Group 4

Group 4 spectra, shown in figure 4.4, consists of 516 samples and shows two well-defined peaks and valleys around frequencies of about 300Hz and 900Hz with one major peak in the

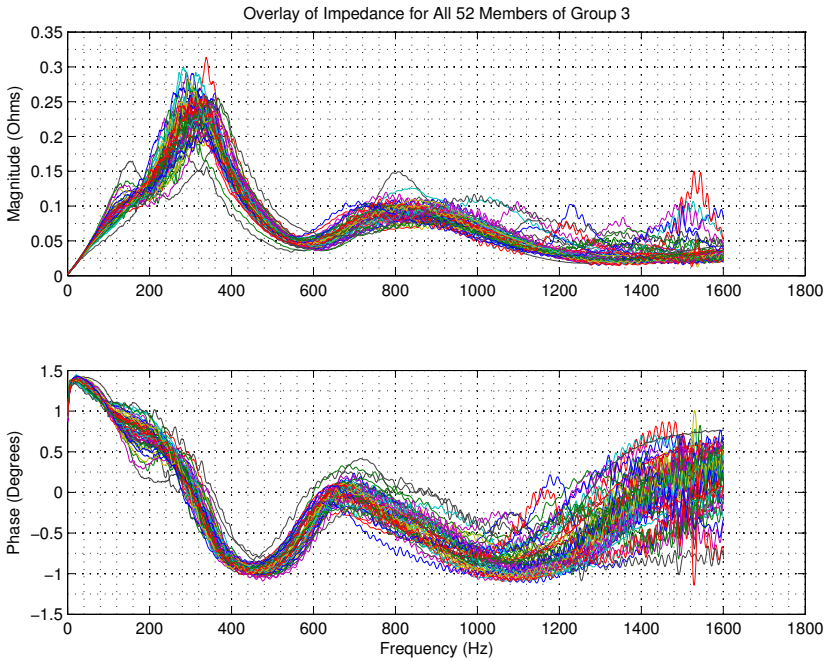


Figure 4.3: Impedance spectra from 52 members of group 3

magnitude. There is also a high level of noise and variance in the higher frequencies. The spectra of all members of group 4 are overlaid in figure 4.4.

4.3.5 Group 5

Group 5 spectra, shown in figure 4.5, consists of 176 samples and shows two well-defined peaks around frequencies of about 600Hz and 1200Hz, despite the high level of variance in the last interesting frequency. The spectra of all members of group 5 are overlaid in figure 4.5.

4.3.6 Group 6

Group 6 spectra, shown in figure 4.6, consists of 169 samples and shows two well-defined peaks around frequencies of about 250Hz and 900Hz. There is also a high level of noise and

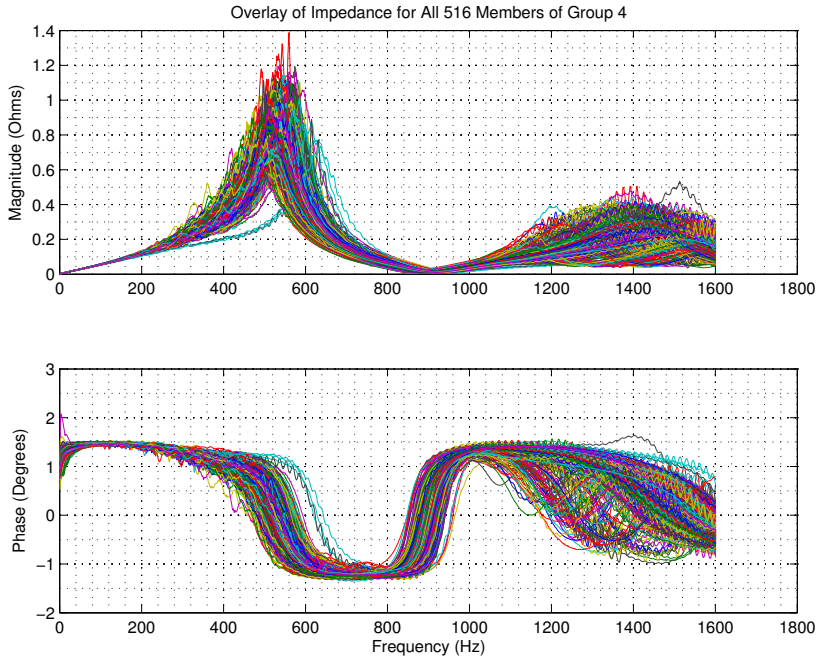


Figure 4.4: Impedance spectra from 516 members of group 4

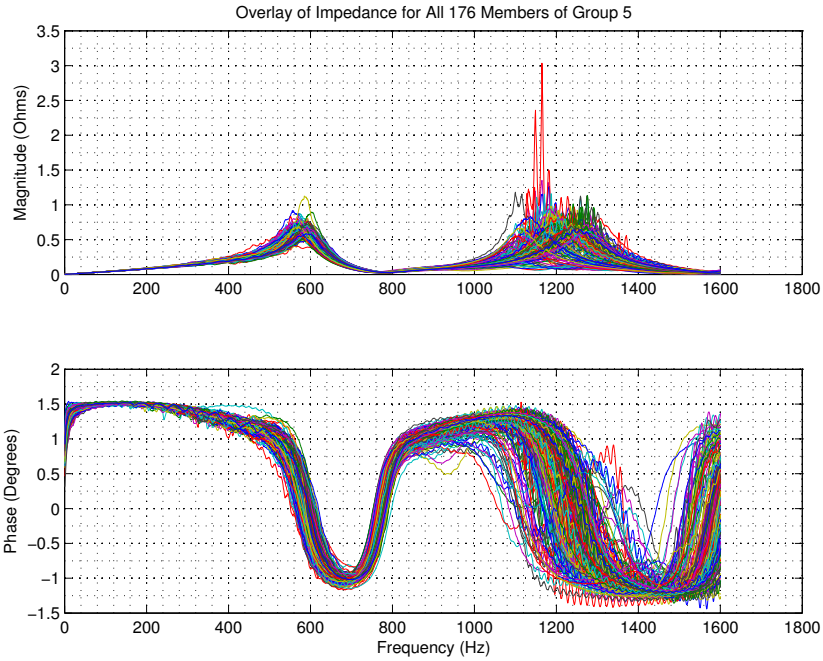


Figure 4.5: Impedance spectra from 176 members of group 5

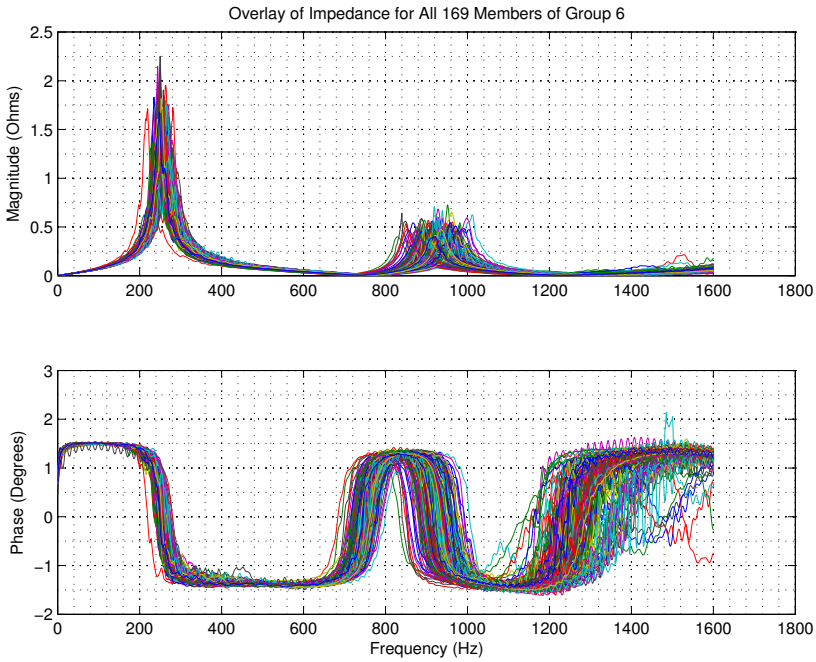


Figure 4.6: Impedance spectra from 169 members of group 6

variance in the higher frequencies. The spectra of all members of group 6 are overlaid in figure 4.6.

4.3.7 Group 7

Group 7 spectra, shown in figure 4.7, consists of 188 samples and shows two well-defined peaks around frequencies of about 250Hz and 950Hz. There is also a high level of noise and variance in the higher frequencies. The spectra of all members of group 7 are overlaid in figure 4.7.

4.3.8 Group 8

Group 8 spectra, shown in figure 4.8, consists of 91 samples and shows two well-defined peaks around frequencies of about 200Hz and 600Hz. There are other frequencies that may have

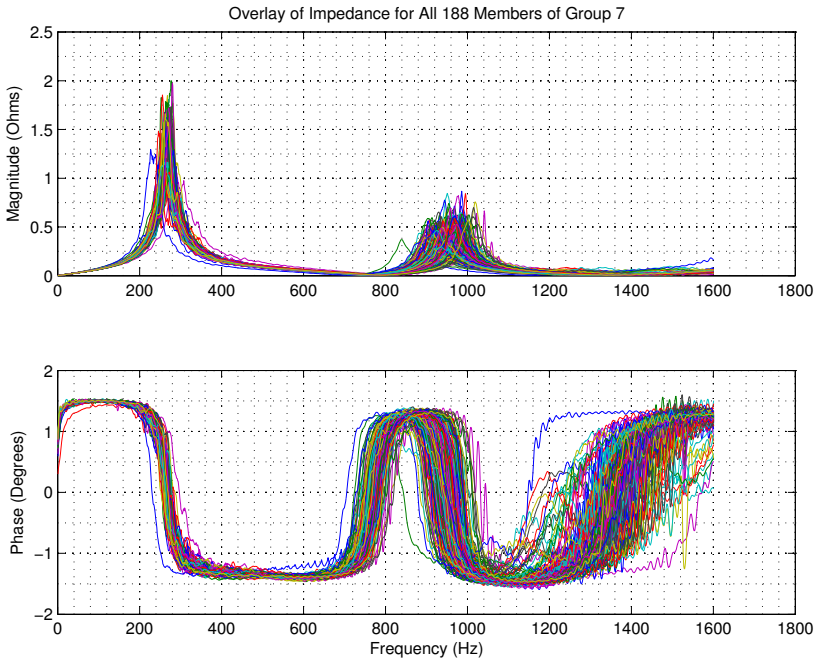


Figure 4.7: Impedance spectra from 188 members of group 7

importance and will be investigated in the model analysis. There is also a high level of noise and variance in the higher frequencies. The spectra of all members of group 8 are overlaid in figure 4.8.

4.3.9 Group 9

Group 9 spectra, shown in figure 4.9, consists of 131 samples and shows two well-defined peaks around frequencies of about 200Hz and 600Hz. There are other frequencies that may have importance and will be investigated in the model analysis. There is also a high level of noise and variance in the higher frequencies. The spectra of all members of group 9 are overlaid in figure 4.9.

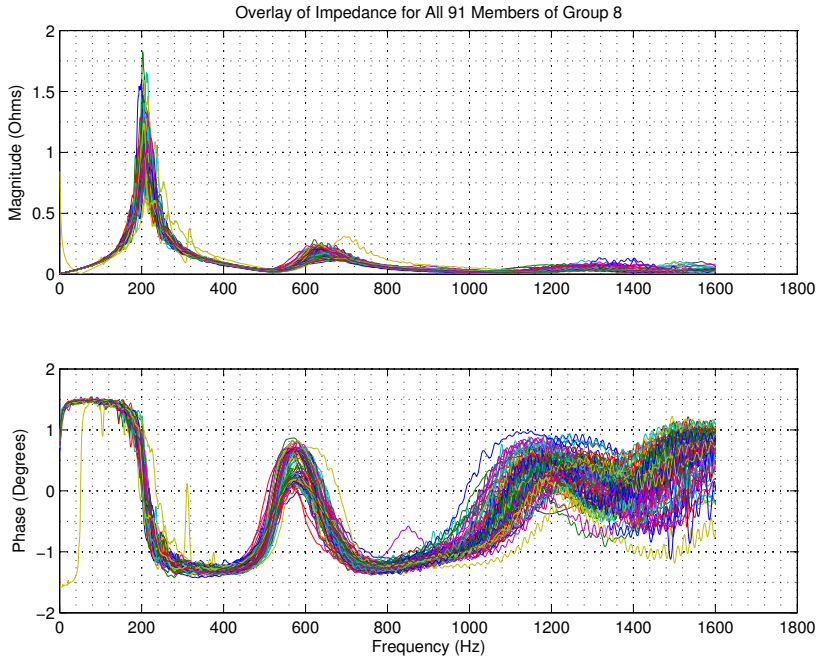


Figure 4.8: Impedance spectra from 91 members of group 8

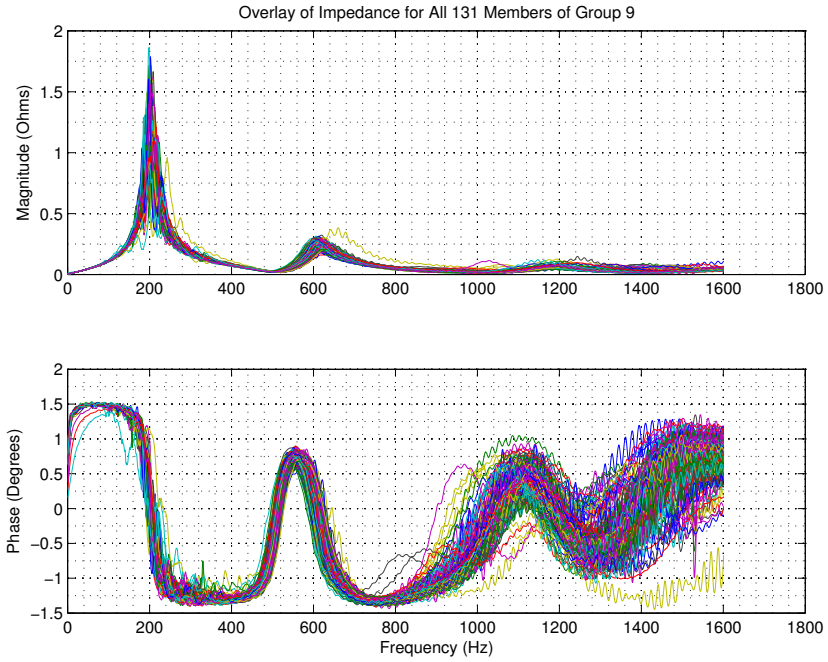


Figure 4.9: Impedance spectra from 131 members of group 9

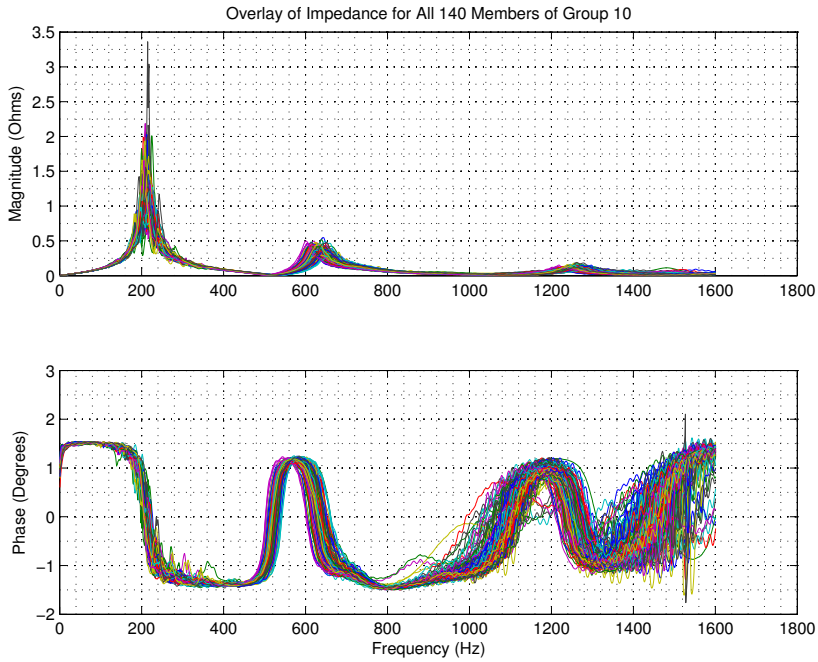


Figure 4.10: Impedance spectra from 140 members of group 10

4.3.10 Group 10

Group 10 spectra, shown in figure 4.10, consists of 140 samples and shows two well-defined peaks around frequencies of about 200Hz and 600Hz. There are other frequencies that may have importance and will be investigated in the model analysis. There is also a high level of noise and variance in the higher frequencies. The spectra of all members of group 10 are overlaid in figure 4.10.

4.3.11 Group 11

Group 11 spectra, shown in figure 4.11, consists of 47 samples and shows two well-defined peaks around frequencies of about 200Hz and 600Hz. There are other frequencies that may have importance and will be investigated in the model analysis. There is also a high level

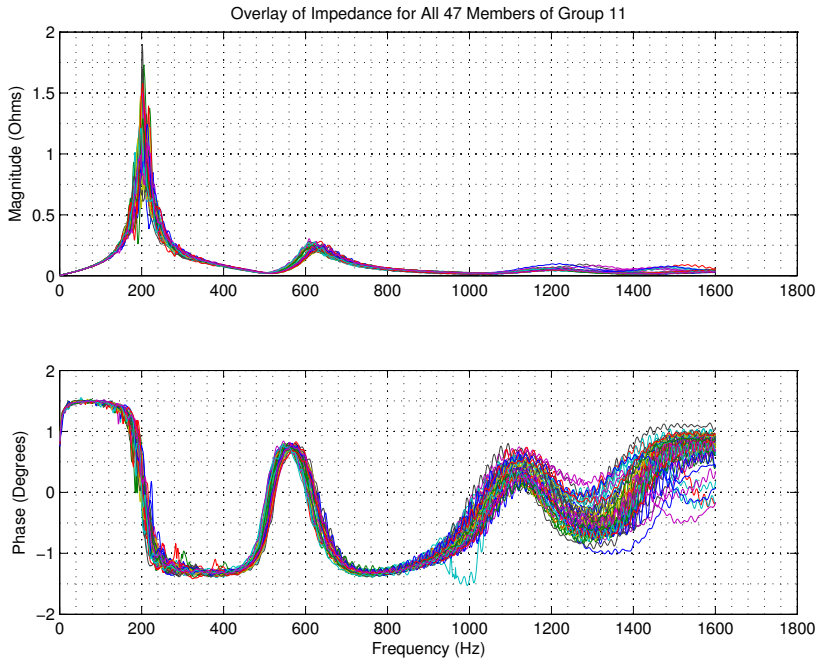


Figure 4.11: Impedance spectra from 47 members of group 11

of noise and variance in the higher frequencies. The spectra of all members of group 11 are overlaid in figure 4.11.

4.3.12 Group 12

Group 12 spectra, shown in figure 4.12, consists of 49 samples and shows two well-defined peaks around frequencies of about 200Hz and 600Hz. There are other frequencies that may have importance and will be investigated in the model analysis. There is also a high level of noise and variance in the higher frequencies. The spectra of all members of group 12 are overlaid in figure 4.12.

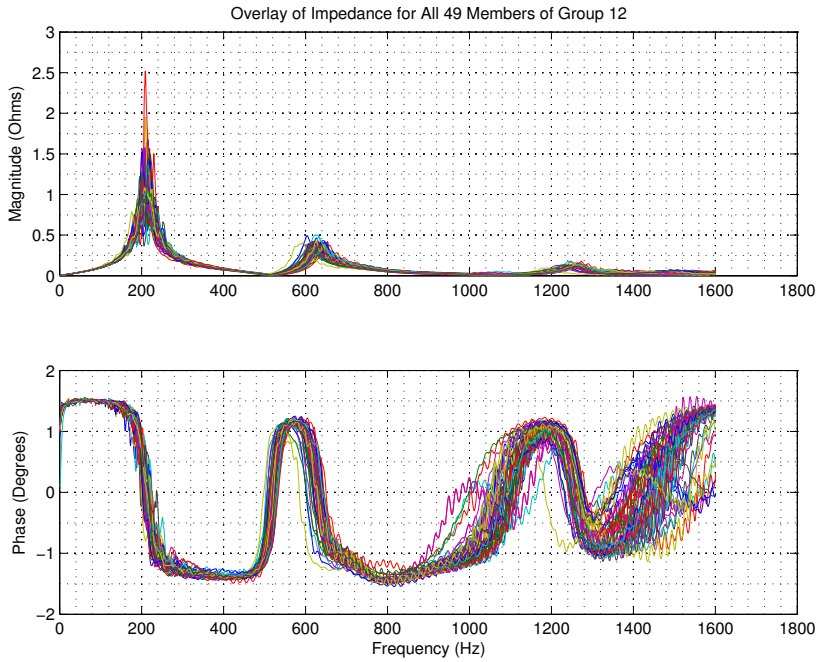


Figure 4.12: Impedance spectra from 49 members of group 12

4.3.13 Group 13

Group 13 spectra, shown in figure 4.13, consists of 99 samples and shows two well-defined peaks around frequencies of about 200Hz and 600Hz. There are other frequencies that may have importance and will be investigated in the model analysis. There is also a high level of noise and variance in the higher frequencies. The spectra of all members of group 13 are overlaid in figure 4.13.

4.4 Model Analysis

The first step to formulating a solution will involve understanding the available data. The equivalent circuit model in figure 4.16 is used to model the characteristics of the objects analyzed. Every data sample was taken from a unique object that could have different component values but still follows the equivalent circuit model. The goal of these first steps

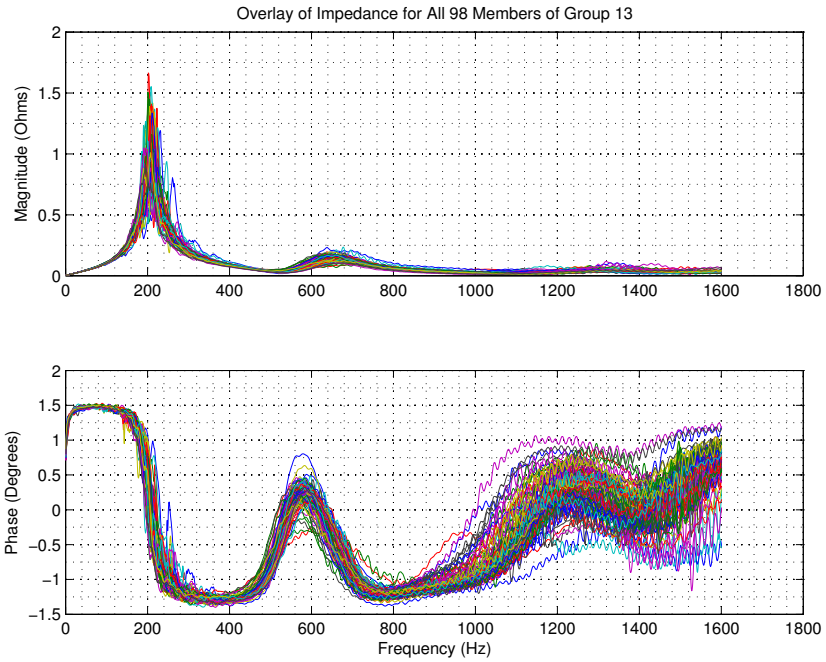


Figure 4.13: Impedance spectra from 99 members of group 13

are to analyze the model to understand the type of data that could be produced and the importance of characteristics within the data.

Performing a frequency analysis of the data plots may show an understanding of the steep peaks and long valleys within the data. The resonant frequency analysis will be obtained using an equivalent circuit impedance and basic circuit analysis techniques.

Detailed analysis steps include:

1. Develop an equivalent circuit impedance equation based on the model provided.

2. Perform a Bode plot analysis to determine corner frequencies and transfer function characteristics.

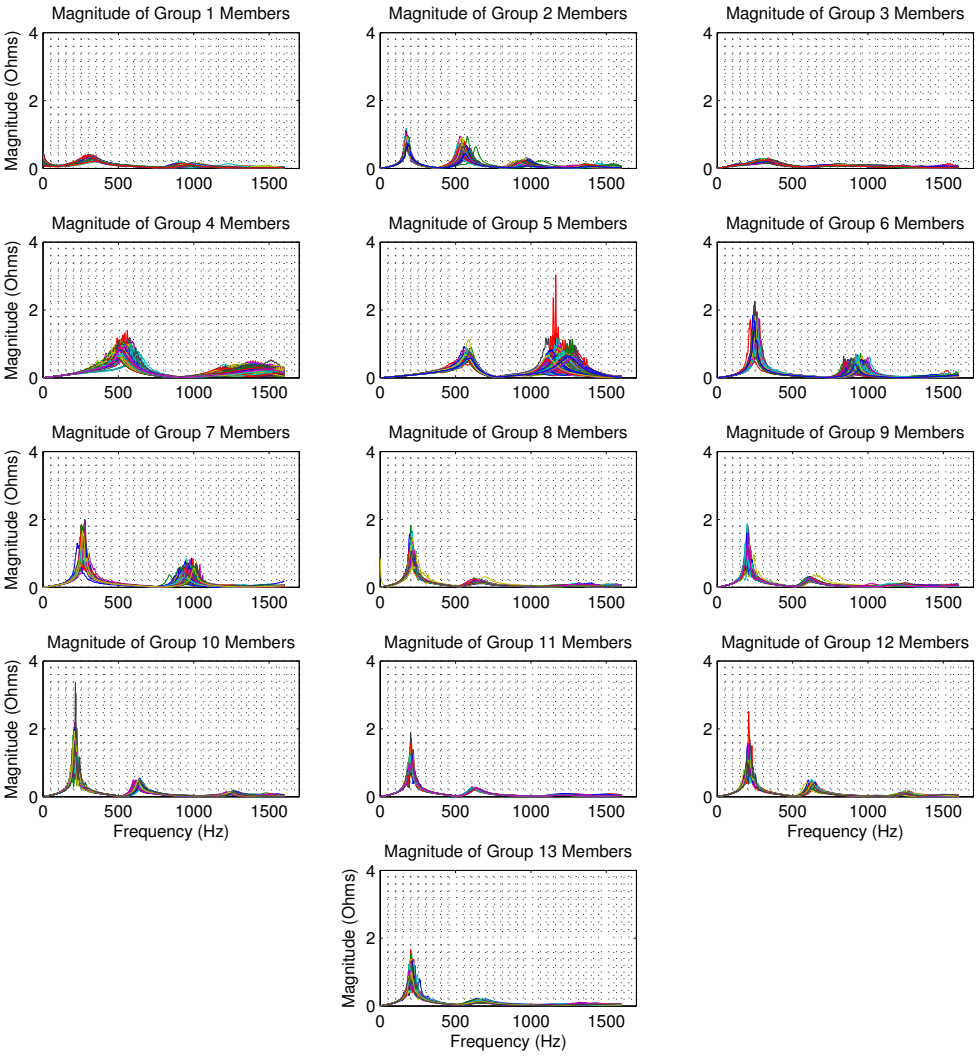


Figure 4.14: Impedance magnitude of all 13 groups.

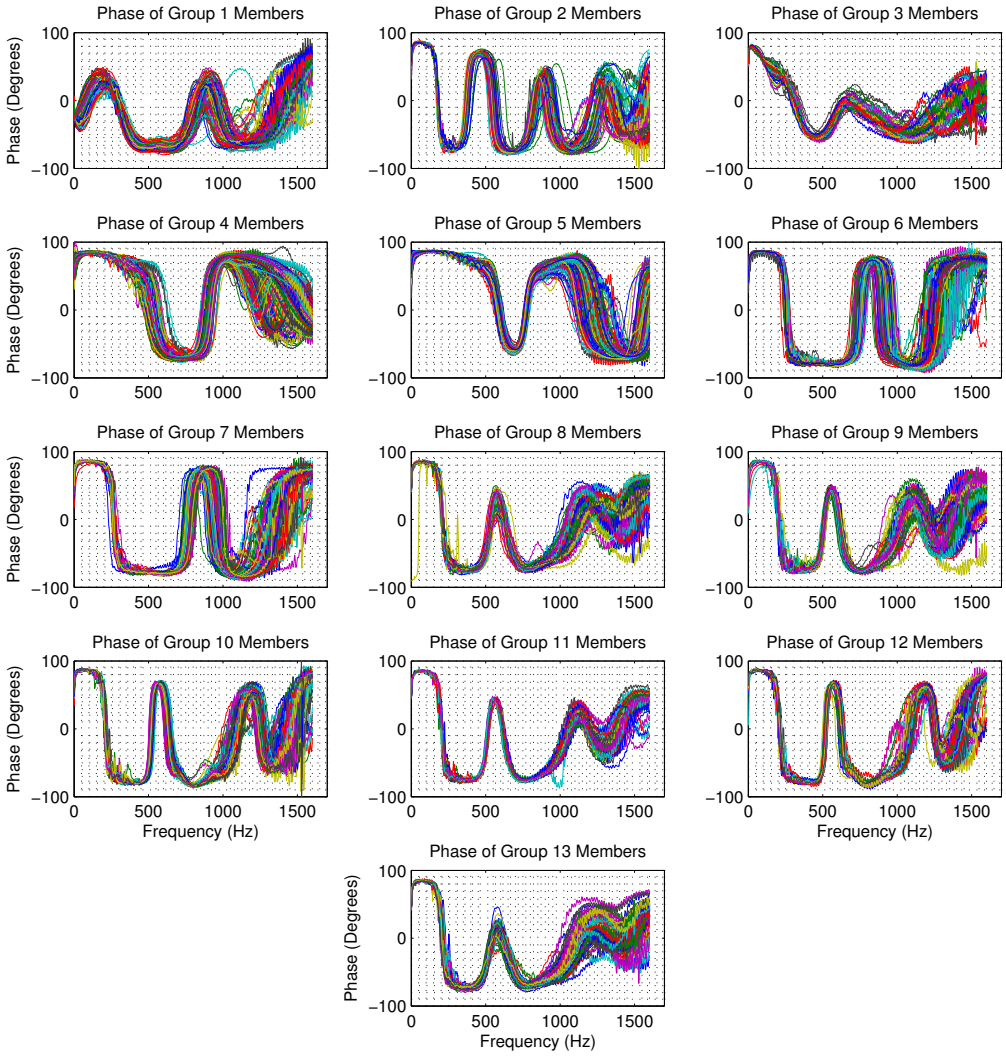


Figure 4.15: Impedance phase of all 13 groups.

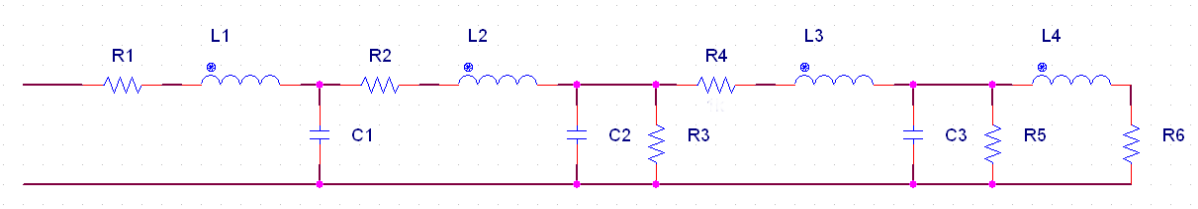


Figure 4.16: Model Circuit for Data

4.5 Comparison of Techniques for Dimension Reduction

The data will be reduced using the previously described techniques and the techniques' effectiveness in retaining information necessary for classification will be compared. The techniques will be analyzed to determine each reduced data set's effectiveness in retaining the most information with the least amount of data. It is expected that all three approaches will have similar results showing that spectra peak information provides the greatest discrimination across groups.

After dimension reduction, an attempt will be made to effectively graph the data showing the types of structures that related samples create. This step will prepare the data for analysis by similarity measures. Three methods will be compared. The steps for each method are outlined below.

1. Method 1: Principal Component Reduction.

   A. Perform SVD on the data matrix to obtain the U, Σ, and V matrices.

$$X = U\Sigma V' \tag{4.1}$$

   B. Analyze the scree plot to determine what should be the optimal number of dimensions.

C. Use the new UΣ data matrix (also referred to as scores), or the equivalent XV matrix, for the similarity analysis.

D. Perform classification using 1-1601 features from the scores matrix and determine the number of features that give the highest number of correct classifications.

2. Method 2: Binning

A. Find the peaks of the impedance spectrum by looking at the zero crossings of the phase. Note that this may not work in all cases.

B. Combine peaks in a single bin by taking an average of the impedance at each peak frequency.

C. Perform classification using 1-1601 bins and determine the number of bins that give the highest number of correct classifications.

3. Analyze the differences between SVD and binned peak analysis. Determine if one is better, and whether both should be used individually, none should be used, or a hybrid should be investigated.

4.6 Similarity/Dissimilarity Measure Effectiveness Comparison

The purpose of this metric comparison is to define the metric that most accurately, based on the associated group assignments, classifies the impedance spectra data used in this research. During this iterative process, the reduced data will be used with a portion being appropriated as training data and the remainder as test data.

1. The following steps will be repeated 20 times.

A. Randomly select 60 percent of the samples from each group to become training samples and the other 40 percent of each group to become test samples.

B. Assign the training samples their pre-defined categories.

C. Determine a virtual centroid sample for each pre-defined sample category by taking an average of all members of the category created in the previous step.

D. For each similarity comparator:

    i. For each test sample:

        a. Calculate the similarity value between the test sample and each category centroid.

        b. Classify the test sample by choosing the category with the most similar value.

        c. Compare the similarity value to the threshold calculated using the methods described in section 4.7. Make a note if the similarity value is less than the threshold.

        d. Determine if the test sample has a similarity value to another category that is higher than the category's threshold. If so, make a note that the test sample could be similar to another category.

    ii. Create a classification confusion matrix for analysis of the results and comparison with actual classification.

    iii. Keep the classification confusion matrix of this comparator with other runs on this comparator for later analysis.

2. Compare the classification confusion matrices for the comparators and determine the comparator with the greatest number of correct classifications and also the comparator with the lowest number of incorrect classifications.

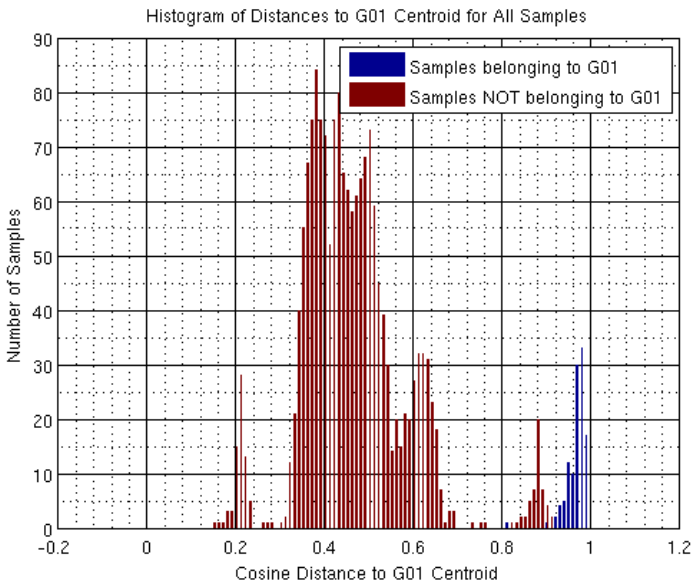


Figure 4.17: Histograph of distances to group G01 centroid.

4.7 Unclassifiable Determination

During the classification process, there is a possibility that samples may not belong to any of the known categories and must be allocated to a new category. This determination will not be a straightforward decision and will straddle the boundaries of unsupervised learning. The goal of this method is to choose a threshold that minimizes the classification error. The methods used here closely follow with the principals of bayesian decision theory [45, p. 20].

An example is shown here. Figure 4.17 shows a histogram of cosine similarity distances for all samples to the centroid (average of the group) of group G01. The samples that belong to group G01 are shown in a different color than those that do not belong to group G01 based on the pre-assigned group numbers. Based on the histogram, a clear separation appears between the non-members and members around a distance of 0.9. A Receiver Operating Characteristic (ROC) curve is shown in figure 4.18 but does not provide much information because the separation is very good between the members and non-members. To obtain a threshold, the histogram was analyzed at every bin (.01 separation) for the

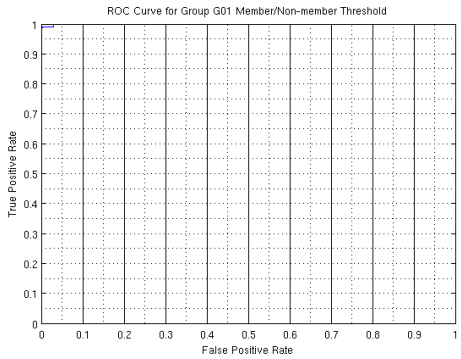


Figure 4.18: ROC curve for member/non-member classification of group G01.

total error classification based on that value of a threshold. For group G01, the threshold of 0.92 has the lowest number of misclassifications to be either a member or non-member with 2 misclassifications. Therefore, we would determine that any sample that has a cosine similarity distance less than 0.92 could possibly be unclassifiable.

These methods will need to be integrated into the similarity/dissimilarity measure comparison and recalculated with each new training set and metric. The following procedure summarizes this method:

1. Determine the threshold limits of classification

    A. Calculate the threshold that minimizes the error classification rate.

4.8 Detailed Steps

This section is a repeat of the steps from the previous sections combined into a single step-by-step procedure.

1. Develop an equivalent circuit impedance equation based on the model provided.

2. Perform a Bode plot analysis to determine corner frequencies and transfer function characteristics.

3. Method 1: Principal Component Reduction.

   A. Perform SVD on the data matrix to obtain the U, Σ, and V matrices.

$$X = U\Sigma V' \tag{4.2}$$

   B. Analyze the scree plot to determine what should be the optimal number of dimensions.

   C. Use the new UΣ data matrix (also referred to as scores), or the equivalent XV matrix, for the similarity analysis.

   D. Perform classification using 1-1601 features from the scores matrix and determine the number of features that give the highest number of correct classifications.

4. Method 2: Binning

   A. Find the peaks of the impedance spectrum by looking at the zero crossings of the phase. Note that this may not work in all cases.

   B. Combine peaks in a single bin by taking an average of the impedance at each peak frequency.

C. Perform classification using 1-1601 bins and determine the number of bins that give the highest number of correct classifications.

5. Analyze the differences between SVD and binned peak analysis. Determine if one is better, and whether both should be used individually, none should be used, or a hybrid should be investigated.

6. The following steps will be repeated 20 times.

   A. Randomly select 60 percent of the samples from each group to become training samples and the other 40 percent of each group to become test samples.

   B. Assign the training samples their pre-defined categories.

   C. Determine a virtual centroid sample for each pre-defined sample category by taking an average of all members of the category created in the previous step.

   D. Determine the threshold limits of classification

      i. Calculate the threshold that minimizes the error classification rate.

   E. For each similarity comparator:

      i. For each test sample:

         a. Calculate the similarity value between the test sample and each category centroid.

         b. Classify the test sample by choosing the category with the most similar value.

         c. Compare the similarity value to the threshold calculated using the methods described in section 4.7. Make a note if the similarity value is less than the threshold.

d. Determine if the test sample has a similarity value to another category that is higher than the category's threshold. If so, make a note that the test sample could be similar to another category.

ii. Create a classification confusion matrix for analysis of the results and comparison with actual classification.

iii. Keep the classification confusion matrix of this comparator with other runs on this comparator for later analysis.

7. Compare the classification confusion matrices for the comparators and determine the comparator with the greatest number of correct classifications and also the comparator with the lowest number of incorrect classifications.

Chapter 5

Results

The results of this research are organized to show details of each of the three phases of the methods. First, the intermediate steps of classification for one similarity function are shown for the cosine similarity function. Then, the classification results for all similarity functions are shown along with statistics for multiple repetitions. Next, the intermediate steps of data dimension reduction using SVD and peak binning are shown. Finally, the results of classification using dimension reduction of all sizes with all similarity functions are shown.

5.1 Data Preprocessing

Graphing of the group members at the beginning of the investigation revealed that some of the samples were acquired in a non-standard process, causing abnormalities in the output that did not fit with the other members of the group. These abnormal samples were removed prior to any operations on the data. Leaving these samples in the data set would result in an improper analysis of the methods of the research. Training samples must be guaranteed to be correctly acquired and correctly classified.

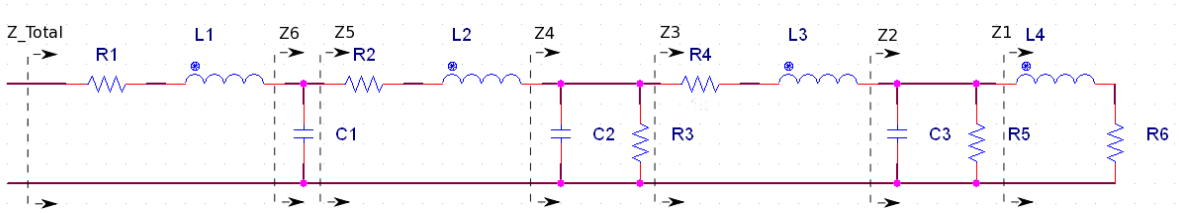


Figure 5.1: Model Circuit for Data with Parts Illustrated

5.2 Model Analysis

5.2.1 Equivalent Circuit Impedance

The first step in the model analysis was to create an equivalent circuit impedance equation from the circuit in figure 4.16. Calculation of an equivalent circuit impedance was done in stages. Figure 5.1 illustrates the breakdown of the circuit.

$$Z_1 = jwL_4 + R_6 \tag{5.1}$$

$$\frac{1}{Z_2} = \frac{1}{\frac{1}{jwC_3}} + \frac{1}{R_5} + \frac{1}{Z_1} \tag{5.2}$$

$$= jwC_3 + \frac{1}{R_5} + \frac{1}{Z_1} \tag{5.3}$$

$$Z_2 = \frac{1}{jwC_3 + \frac{1}{R_5} + \frac{1}{Z_1}} \tag{5.4}$$

$$Z_3 = R_4 + jwL_3 + Z_2 \tag{5.5}$$

$$Z_4 = \frac{1}{jwC_2 + \frac{1}{R_3} + \frac{1}{Z_3}} \tag{5.6}$$

$$Z_5 = R_2 + jwL_2 + Z_4 \tag{5.7}$$

$$Z_6 = \frac{1}{jwC_1 + \frac{1}{Z_5}} \tag{5.8}$$

$$Z_{total} = R_1 + L_1 + Z_6 \tag{5.9}$$

Solving the equations above using MATLAB's COLLECT function in the symbolic toolbox gives a rational transfer function with a form similar to equation 5.10 where $s = jw$.

$$Z = \frac{(A)s^7 + (B)s^6 + (C)s^5 + (D)s^4 + (E)s^3 + (F)s^2 + (G)s + (H)}{(I)s^6 + (J)s^5 + (K)s^4 + (L)s^3 + (M)s^2 + (N)s + (O)} \tag{5.10}$$

This transfer function indicates seven zeros and six poles exist in the system. These values are compared to transfer function estimation using the measured data in the next section.

5.2.2 Transfer Function Estimation

A bode plot of the measured data from sample one of group one is shown in figure 5.2. The bode plot shows at least eight zeros and at least eight poles. Using MATLAB's INVFREQS command with eight zeros and eight poles, the transfer function in equation 5.11 was acquired. The estimated transfer function is plotted in figure 5.2.

$$\frac{0.01s^8 - 1.8s^7 + (6.1E4)s^6 - (2.4E7)s^5 + (6E10)s^4 - (4.2E13)s^3 + (1.4E16)s^2 - (1.4E19)s + 1.4E20}{s^8 + 111s^7 + (E6)s^6 + (1.7E8)s^5 + (2.9E12)s^4 - (1.7E10)s^3 + (7E17)s^2 - (1.8E19)s + 2.2E22} \tag{5.11}$$

The resulting output bode plot in figure 5.2 validates eight zeros and eight poles in the measured data. Notice that the characteristic frequencies of both bode plots are aligned. The characteristic frequencies are shown to be important parts of the model and the dimension reduction portion of this research will test that conclusion.

5.3 Classification Using Cosine Similarity

The most significant part of the classification system is the algorithm for determination of similarity between an unknown sample and the known groups. This section shows the

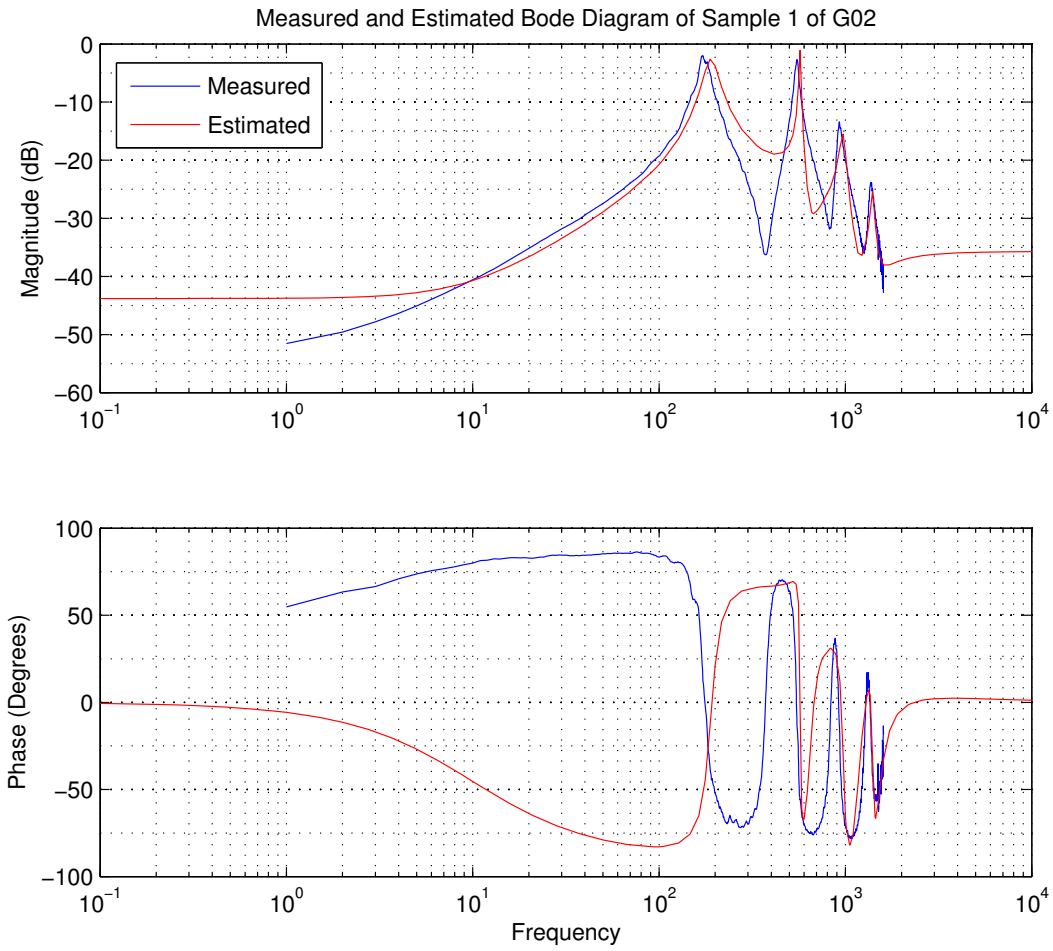


Figure 5.2: Bode plot of measured and estimated transfer functions from sample 1 of group 01.

Table 5.1: Number of total samples split into training and test categories for each group.

Group	# of Members	# Training	# Test
G01	115	69	46
G02	71	43	28
G03	52	31	21
G04	516	310	206
G05	176	106	70
G06	169	101	68
G07	188	113	75
G08	91	55	36
G09	131	79	52
G10	140	84	56
G11	47	28	19
G12	49	29	20
G13	98	59	39
Total	1843	1107	736

classification results using 1844 impedance spectrum samples from 13 predefined groups for one run of one similarity function, cosine similarity.

5.3.1 Separation of Training and Test Samples

Table 5.1 shows the number of samples in each predefined group. Each group was divided by random assignment into 60% training and 40% test samples using MATLAB's random functions. Table 5.1 also shows the number of samples from each group that were placed into the training and test categories.

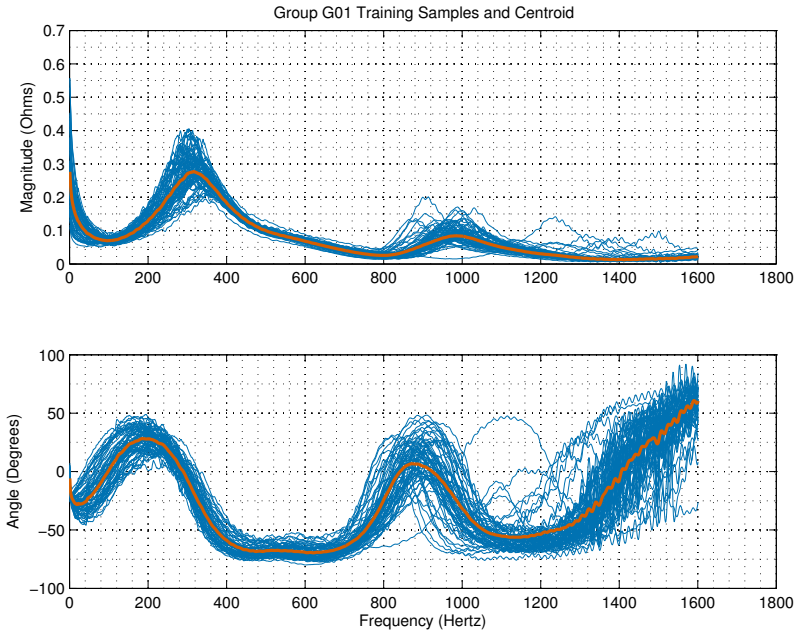


Figure 5.3: Group G01 training samples drawn in blue and the centroid of the training samples drawn in orange.

5.3.2 Centroid Calculation

The training samples from each group were used to determine the centroid and threshold of classification. The centroid of each group was calculated using the average of each dimension of all training samples in the group. Figure 5.3 shows the magnitude and phase of all training samples of group G01 with their centroid. Figure 5.4 shows the centroids of all 13 groups.

5.3.3 Threshold Determination

The similarity distances of all group members to the group's centroid were calculated for each group 01-13. Figure 5.5 shows a binned histogram of the distances of group G01 training samples to the G01 training samples' centroid in blue, calculated using the cosine similarity function. Each blue bar represents the percentage of total group G01 training samples that had a similarity distance in that bin. The sum of all blue bars is equal to 100 percent. The

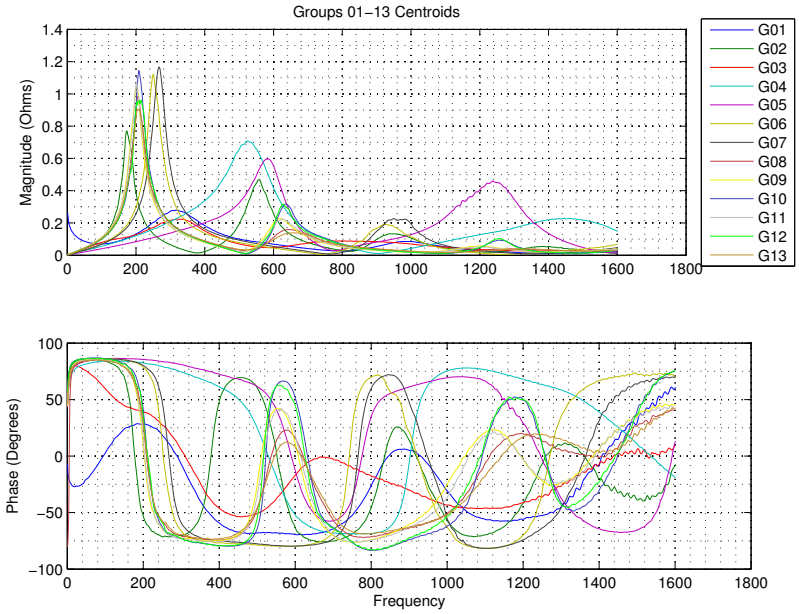


Figure 5.4: Centroids of groups 01-13

cosine similarity distance of all non-group samples (Groups 02-13) to group G01 training samples' centroid was also calculated. Figure 5.5 also shows a binned histogram of the distances of all non-G01 training samples to the G01 training samples' centroid with red bars. Each red bar represents the percentage of total non-group training samples that had a similarity distance in that bin. The sum of all red bars is equal to 100 percent.

The error rate was calculated as the sum of the percentage of group members to the left of the similarity value and the percentage of non-group members to the right of the similarity value. Figure 5.5 shows the error rate, or rate of misclassification as a function of the similarity value if that similarity value is used as the threshold to determine group membership. The optimum threshold is shown with a orange line and matches the location of the minimal error rate in the figure. The threshold for each group was recorded for classification of the test samples.

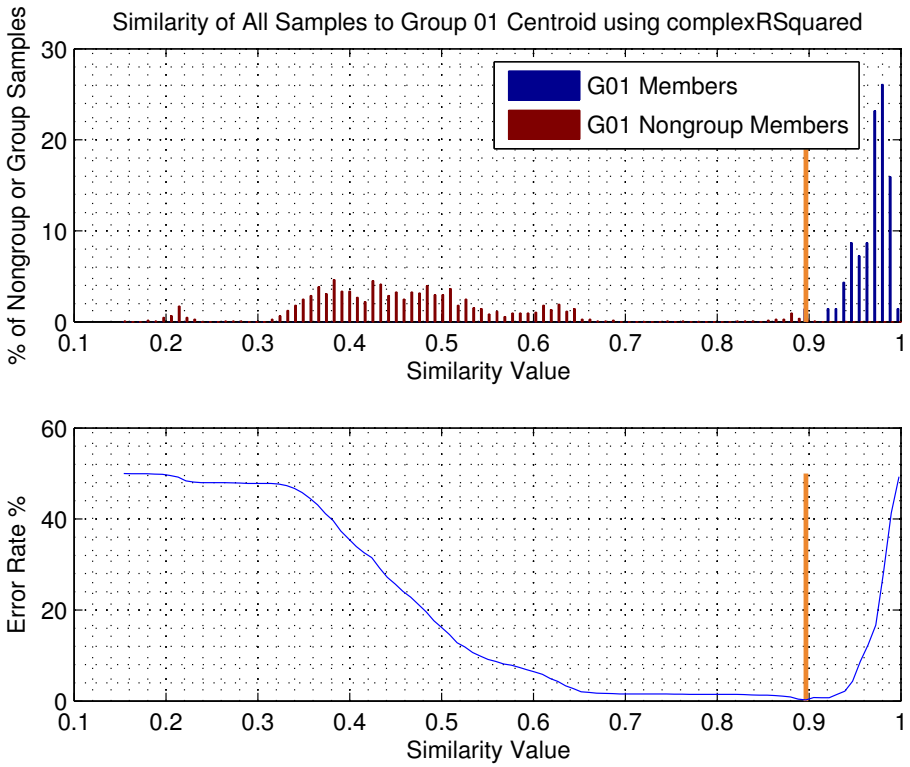


Figure 5.5: Histogram of group 01 test samples' cosine similarity distance to the training samples' centroid. Calculated threshold drawn in orange.

5.3.4 Theshold Scenarios

For classification, the similarity value between each test sample and each group's centroid was calculated and the test sample is then assigned to the group with the most similar centroid. Three scenarios were studied:

1. A test sample was classified to the group with the most similar centroid regardless of threshold value.

2. A test sample was classified to the group with the most similar centroid only if the similarity value was above the threshold for that group. If the sample was not above the threshold, it was placed in a special, unclassifiable group called Un2 in the classification confusion matrices.

3. A test sample was classified to the group with the most similar centroid only if the similarity value was above the threshold for the group and the similarity value to all other groups was below the threshold for the group. If the sample was above the similarity threshold for more than one group and none of the groups were the correct group, it was placed in a special, unclassifiable group called Un3 in the classification confusion matrices. If the sample was above the similarity threshold for more than one group and one of the groups was the correct group, it was placed in a special, unclassifiable group called Un4 in the classification confusion matrices.

5.3.5 Classification Results

The results of classification using methods in this research are displayed using a classification confusion matrix [46, p. 96]. The matrix shows the real group membership of the test samples in the columns and the number of samples classified to the group in the rows. For example, all of the samples that belong to group 1 are in column G01. The number of samples that were correctly classified for group 1 are in the column G01 and row G01. Any samples that

were misclassified add to the counts in the other rows. The number of correct classifications can be easily distinguished by looking at the counts on the diagonal.

Tables 5.2, 5.3, and 5.4 show classification confusion matrices for each of the scenarios above using the cosine similarity function. The groups along the left side indicate the group that the test sample was classified to and the groups along the top indicate the group that the test sample really belongs to. The number of correct classifications is obtained by calculating the sum of the counts on the diagonal. Table 5.5 shows the total number of test samples correctly classified, incorrectly classified, and unclassifiable for each of the three scenarios.

Table 5.2: Classification confusion matrix for scenario 1 from section 5.3.4 using the cosine similarity function.

		True Group Membership												
		G01	G02	G03	G04	G05	G06	G07	G08	G09	G10	G11	G12	G13
Classified Group Membership	G01	46	0	0	0	0	0	0	0	0	0	0	0	0
	G02	0	27	0	0	0	0	0	0	0	0	0	0	0
	G03	0	0	21	0	0	0	0	0	0	0	0	0	0
	G04	0	0	0	205	0	0	0	0	0	0	0	0	0
	G05	0	0	0	1	70	0	0	0	0	0	0	0	0
	G06	0	0	0	0	0	54	3	0	0	0	0	0	0
	G07	0	0	0	0	0	14	72	0	0	0	0	0	0
	G08	0	0	0	0	0	0	0	13	0	0	2	0	16
	G09	0	1	0	0	0	0	0	2	32	2	7	1	3
	G10	0	0	0	0	0	0	0	2	1	30	1	6	0
	G11	0	0	0	0	0	0	0	8	15	2	7	2	4
	G12	0	0	0	0	0	0	0	2	3	22	2	11	0
	G13	0	0	0	0	0	0	0	9	1	0	0	0	17

Table 5.3: Classification confusion matrix for scenario 2 from section 5.3.4 using the cosine similarity function.

		True Group Membership												
		G01	G02	G03	G04	G05	G06	G07	G08	G09	G10	G11	G12	G13
Classified Group Membership	G01	45	0	0	0	0	0	0	0	0	0	0	0	0
	G02	0	27	0	0	0	0	0	0	0	0	0	0	0
	G03	0	0	21	0	0	0	0	0	0	0	0	0	0
	G04	0	0	0	204	0	0	0	0	0	0	0	0	0
	G05	0	0	0	0	68	0	0	0	0	0	0	0	0
	G06	0	0	0	0	0	54	3	0	0	0	0	0	0
	G07	0	0	0	0	0	14	72	0	0	0	0	0	0
	G08	0	0	0	0	0	0	0	13	0	0	2	0	15
	G09	0	0	0	0	0	0	0	2	31	2	7	1	3
	G10	0	0	0	0	0	0	0	2	1	30	1	6	0
	G11	0	0	0	0	0	0	0	8	14	2	7	2	4
	G12	0	0	0	0	0	0	0	2	2	22	2	10	0
	G13	0	0	0	0	0	0	0	9	0	0	0	0	17
	Un2	1	1	0	2	2	0	0	0	4	0	0	1	1

Table 5.4: Classification confusion matrix for scenario 3 from section 5.3.4 using the cosine similarity function.

		True Group Membership												
		G01	G02	G03	G04	G05	G06	G07	G08	G09	G10	G11	G12	G13
Classified Group Membership	G01	44	0	0	0	0	0	0	0	0	0	0	0	0
	G02	0	28	0	0	0	0	0	0	0	0	0	0	0
	G03	0	0	20	0	0	0	0	0	0	0	0	0	0
	G04	0	0	0	199	0	0	0	0	0	0	0	0	0
	G05	0	0	0	0	67	0	0	0	0	0	0	0	0
	G06	0	0	0	0	0	32	0	0	0	0	0	0	0
	G07	0	0	0	0	0	1	2	0	0	0	0	0	0
	G08	0	0	0	0	0	0	0	0	0	0	0	0	0
	G09	0	0	0	0	0	0	0	0	0	0	0	0	0
	G10	0	0	0	0	0	0	0	0	0	0	0	0	0
	G11	0	0	0	0	0	0	0	0	0	0	0	0	0
	G12	0	0	0	0	0	0	0	0	0	0	0	0	0
	G13	0	0	0	0	0	0	0	0	0	0	0	0	0
	Un2	2	0	0	1	0	1	1	2	2	2	0	0	0
	Un3	0	0	0	0	0	0	0	3	3	0	0	1	0
	Un4	0	0	1	6	3	34	72	31	47	54	19	19	40

Table 5.5: Classification results using the cosine similarity function.

Scenario	Correctly Classified	Incorrectly Classified	Unclassifiable
1	605 (82.09%)	132 (17.91%)	0 (0%)
2	599 (81.28%)	126 (17.10%)	12 (1.63%)
3	392 (53.19%)	1 (0.14%)	344 (46.68%)

5.4 Classification using All Metrics

Classification results similar to those reported in section 5.3 were obtained for these similarity functions: Inner Product, Euclidean Distance, Mahalanobis Distance, Manhattan Distance, Average Distance, Squared Chord Distance, Canberra Distance, Coefficient of Divergence, Modified Boolean Correlation, Average Weight of Shared Terms, Overlap, Cosine Similarity, Similarity Index, and Tanimoto's. Each of these functions has been described in section 2.3. Tables 5.6, 5.7, and 5.8 show the results of classification using scenarios 1, 2, and 3 respectively using the same training/test set random permutation.

The results of performing classification using scenario 2 from section 5.3.4 with 5 repetitions are shown in table 5.10. Note that the difference in each run are the random assignment of the samples to training or test categories. Statistics from 20 repetitions are shown in table 5.9 and a box plot showing the median, $25^{th}/75^{th}$ percentile, and outliers is shown in figure 5.6.

Table 5.6: Classification of all metrics using scenario 1 from section 5.3.4 and using all 1601 frequencies.

Metric	Correctly Classified	Incorrectly Classified	Unclassifiable
Canberra	640 (87%)	97 (13%)	0 (0%)
Manhattan	624 (85%)	113 (15%)	0 (0%)
Similarity Index	622 (84%)	115 (16%)	0 (0%)
Cosine	597 (81%)	140 (19%)	0 (0%)
Euclidean	596 (81%)	141 (19%)	0 (0%)
Coe of Divergence	594 (81%)	143 (19%)	0 (0%)
Overlap	590 (80%)	147 (20%)	0 (0%)
Mahalanobis	518 (70%)	219 (30%)	0 (0%)
Squared Chord	503 (68%)	234 (32%)	0 (0%)
Inner Product	479 (65%)	258 (35%)	0 (0%)
Average	372 (50%)	365 (50%)	0 (0%)
Avg Weight	206 (28%)	531 (72%)	0 (0%)
Tanimoto's	231 (31%)	506 (69%)	0 (0%)
Mod Bool Correlat	206 (28%)	531 (72%)	0 (0%)

Table 5.7: Classification of all metrics using scenario 2 from section 5.3.4 and using all 1601 frequencies.

Metric	Correctly Classified	Incorrectly Classified	Unclassifiable
Canberra	631 (86%)	87 (12%)	19 (2%)
Manhattan	617 (84%)	107 (14%)	13 (2%)
Similarity Index	620 (84%)	107 (15%)	10 (1%)
Cosine	588 (80%)	130 (18%)	19 (2%)
Euclidean	590 (80%)	133 (18%)	14 (2%)
Coe of Divergence	588 (80%)	122 (16%)	27 (4%)
Overlap	583 (79%)	126 (17%)	28 (4%)
Mahalanobis	545 (74%)	172 (23%)	20 (3%)
Squared Chord	482 (65%)	214 (29%)	41 (6%)
Inner Product	467 (63%)	76 (10%)	194 (26%)
Average	347 (47%)	350 (48%)	40 (5%)
Avg Weight	205 (28%)	61 (8%)	471 (64%)
Tanimoto's	217 (29%)	492 (67%)	28 (4%)
Mod Bool Correlat	200 (27%)	46 (6%)	491 (67%)

Table 5.8: Classification of all metrics using scenario 3 from section 5.3.4 and using all 1601 frequencies.

Metric	Correctly Classified	Incorrectly Classified	Unclassifiable
Canberra	424 (58%)	3 (0.4%)	310 (42%)
Manhattan	378 (51%)	5 (1%)	354 (48%)
Cosine	382 (52%)	2 (0.3%)	353 (48%)
Euclidean	243 (33%)	3 (0.4%)	491 (67%)
Similarity Index	332 (45%)	4 (1%)	401 (54%)
Overlap	346 (47%)	6 (1%)	385 (52%)
Coe of Divergence	373 (51%)	2 (0.3%)	362 (49%)
Squared Chord	283 (39%)	10 (1%)	444 (60%)
Inner Product	25 (3%)	0 (0%)	712 (97%)
Mahalanobis	307 (42%)	7 (1%)	423 (57%)
Average	37 (5%)	2 (0.3%)	698 (95%)
Avg Weight	0 (0%)	0 (0%)	737 (100%)
Tanimoto's	71 (10%)	3 (0.4%)	663 (90%)
Mod Bool Correlat	0 (0%)	0 (0%)	737 (100%)

Table 5.9: Statistics from 20 repetitions of classification using all 1601 frequencies, scenario two from section 5.3.4, and different training/test permutations.

Metric	Mean	Variance	Std Deviation	Std Error	95% Confidence Interval
Canberra	624 (85%)	41.10	6.41	1.43	621.19 (84%) - 626.81 (85%)
Manhattan	608 (82%)	49.55	7.04	1.57	604.87 (82%) - 611.03 (83%)
Similarity Index	597 (81%)	74.69	8.64	1.93	593.31 (81%) - 600.89 (82%)
Cosine	593 (80%)	58.35	7.64	1.71	589.20 (80%) - 595.90 (81%)
Euclidean	592 (80%)	47.41	6.89	1.54	589.28 (80%) - 595.32 (81%)
Coe of Divergence	576 (78%)	34.15	5.84	1.31	573.39 (78%) - 578.51 (78%)
Overlap	571 (77%)	71.56	8.46	1.89	567.09 (77%) - 574.51 (78%)
Mahalanobis	551 (75%)	52.86	7.27	1.63	547.61 (74%) - 553.99 (75%)
Squared Chord	480 (65%)	101.55	10.08	2.25	475.13 (64%) - 483.97 (66%)
Inner Product	478 (65%)	104.65	10.23	2.29	473.57 (64%) - 482.53 (65%)
Average	353 (48%)	100.65	10.03	2.24	348.10 (47%) - 356.90 (48%)
Tanimoto	208 (28%)	133.29	11.55	2.58	203.19 (28%) - 213.31 (29%)
Avg Weight	204 (28%)	2.21	1.49	0.33	203.05 (28%) - 204.35 (28%)
Mod Bool Correlat	200 (27%)	5.14	2.27	0.51	198.61 (27%) - 200.59 (27%)

Figure 5.6: Box plot of 20 repetitions of classification using all 1601 frequencies, scenario two from section 5.3.4, and different training/test permutations.

The time required for caculation of the metrics using all 1601 frequencies/dimensions and scenario 2 ranged from 0.1 to 71 seconds. Figures 5.7 and 5.8 are box plots of the amount of time required for calculation of the metrics over 20 repetitions using different training/test permuations each time. Table 5.11 is a summary of other statistics on the same repetitions.

Table 5.10: Number of samples correctly classified during multiple repetitions of the classification methods using scenario 2 from section 5.3.4 and using all 1601 frequencies.

Metric	Rep 1	Rep 2	Rep 3	Rep 4	Rep 5
Canberra	633 (86%)	623 (85%)	615 (83%)	624 (85%)	609 (83%)
Manhattan	630 (85%)	611 (83%)	603 (82%)	606 (82%)	612 (83%)
Cosine	595 (81%)	600 (81%)	599 (81%)	595 (81%)	596 (81%)
Euclidean	598 (81%)	602 (82%)	586 (80%)	590 (80%)	593 (80%)
Similarity Index	612 (83%)	586 (80%)	584 (79%)	594 (81%)	589 (80%)
Overlap	586 (80%)	575 (78%)	568 (77%)	575 (78%)	568 (77%)
Coe of Divergence	582 (79%)	576 (78%)	575 (78%)	575 (78%)	563 (76%)
Mahalanobis	538 (73%)	558 (76%)	553 (75%)	544 (74%)	558 (76%)
Squared Chord	488 (66%)	479 (65%)	475 (64%)	475 (64%)	484 (66%)
Inner Product	477 (65%)	479 (65%)	476 (65%)	480 (65%)	470 (64%)
Average	361 (49%)	357 (48%)	348 (47%)	352 (48%)	349 (47%)
Avg Weight	205 (28%)	203 (28%)	203 (28%)	205 (28%)	204 (28%)
Tanimoto's	214 (29%)	214 (29%)	193 (26%)	190 (26%)	209 (28%)
Mod Bool Correlat	195 (26%)	200 (27%)	199 (27%)	199 (27%)	198 (27%)

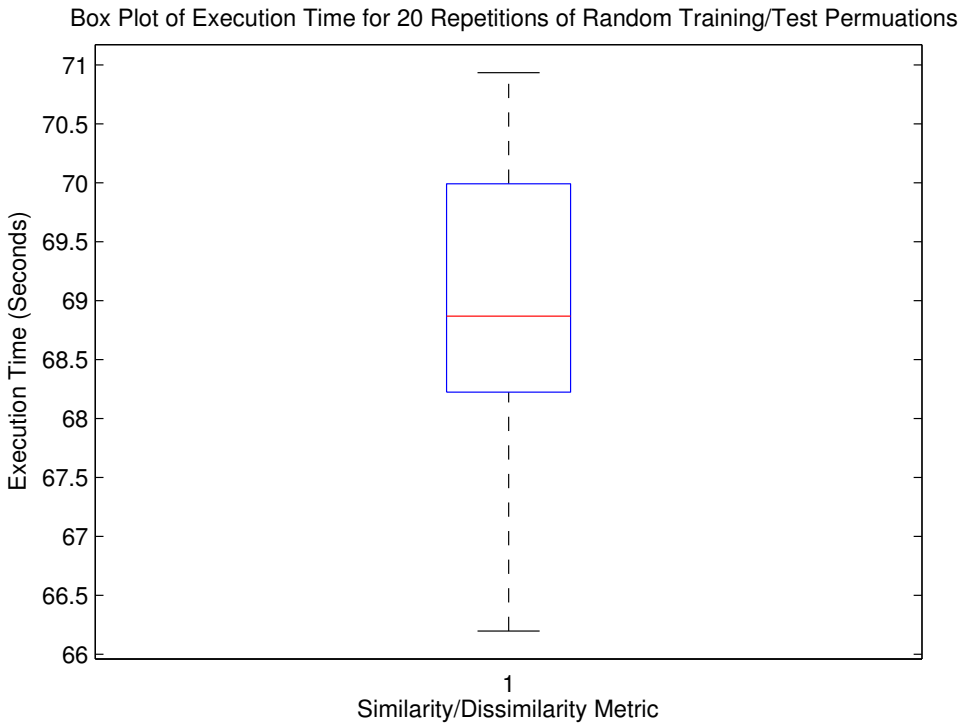


Figure 5.7: Box plot of metric 1 (Mahalanobis Distance) from timing of the metrics using 20 repetitions, 1601 frequencies, and scenario 2 from section 5.3.4.

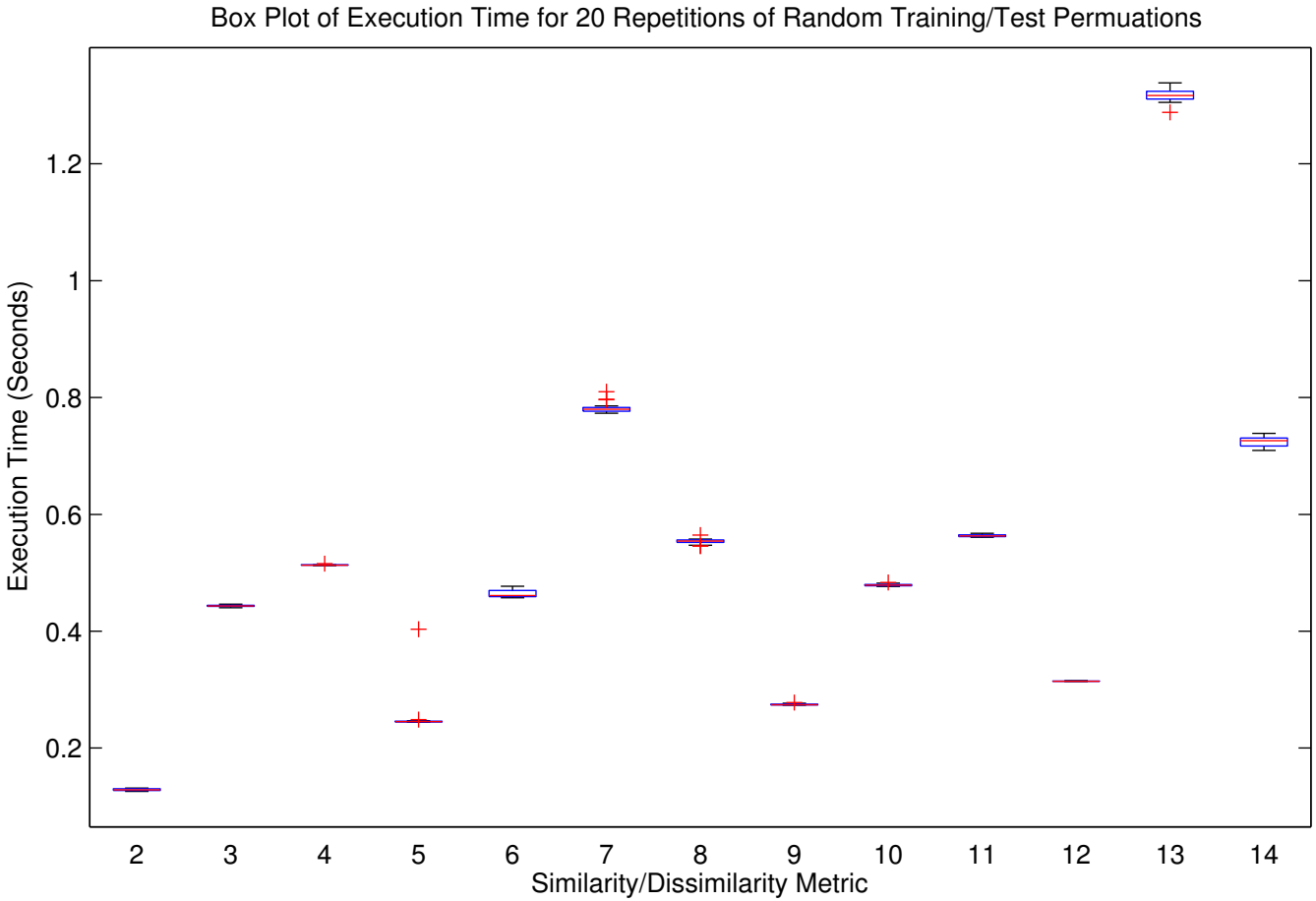


Figure 5.8: Box plot of metrics 2-14 (See table 5.12 for metric names) from timing of the metrics using 20 repetitions, 1601 frequencies, and scenario 2 from section 5.3.4.

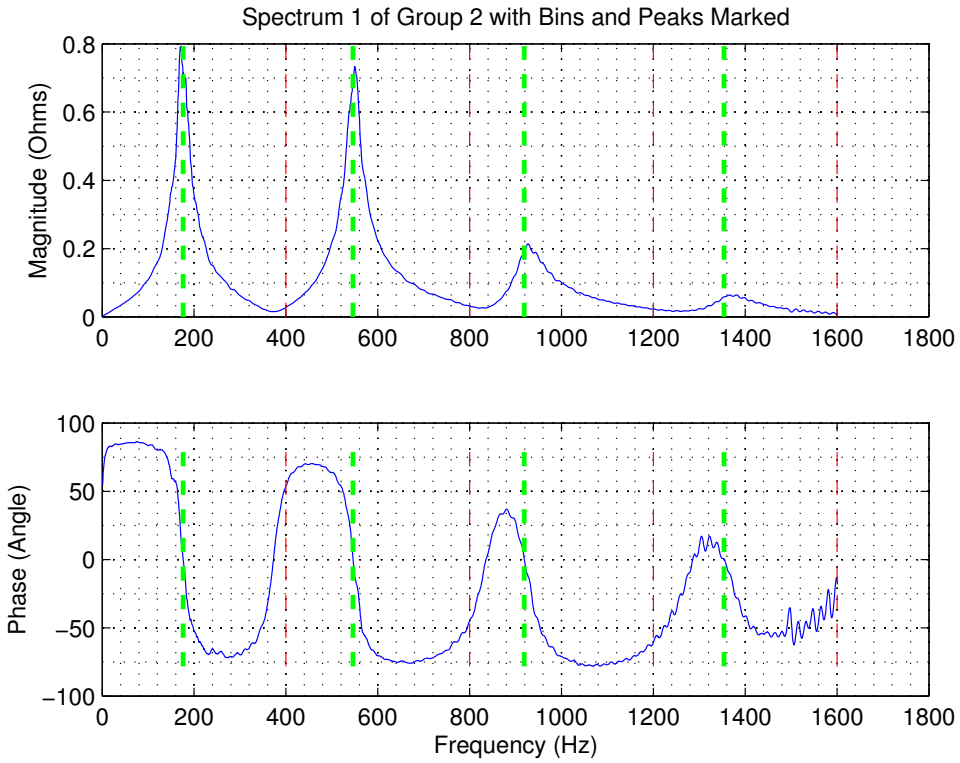


Figure 5.9: Spectrum 1 from group 1 with bins overlaid in red and peaks marked in green.

5.5 Dimension Reduction Using Peak Binning

The methods for these results created N evenly sized bins from the original 1601 frequencies and recorded the average values of all peaks located in the bin. N has a value between 1 and 1601. Figure 5.9 illustrates spectrum sample 1 of group 02 being divided into 4 bins of 400, 400, 400, 400, and 401 frequencies with red lines.

The algorithm for determination of a peak was unique to this data. For this data, the frequencies of magnitude peaks was obtained by looking at zero crossings for the spectrum phase. Figure 5.9 shows the zero crossings for sample 1 of group 02 in green.

To determine the most effective bin demarcations, all three scenarios from section 5.3 were used to classify the same training/test group with bin counts 1-1601. Figures 5.10,

Table 5.11: Box plot from timing of the metrics using 20 repetitions, 1601 frequencies, and scenario 2 from section 5.3.4.

Metric	Mean	Var	Std Dev	Std Error	95% Conf Interval
Mahalanobis	68.93	1.79	1.34	0.30	68.35-69.52
Similarity Index	1.32	0.00	0.01	0.00	1.31-1.32
Canberra	0.78	0.00	0.01	0.00	0.78-0.79
Tanimoto	0.72	0.00	0.01	0.00	0.72-0.73
Overlap	0.56	0.00	0.00	0.00	0.56-0.56
Coe of Divergence	0.55	0.00	0.00	0.00-0.55	0.56
Manhattan	0.51	0.00	0.00	0.00	0.51-0.51
Avg Weight	0.48	0.00	0.00	0.00	0.48-0.48
Squared Chord	0.46	0.00	0.01	0.00	0.46-0.47
Euclidean	0.44	0.00	0.00	0.00	0.44-0.44
Cosine	0.31	0.00	0.00	0.00	0.31-0.31
Mod Bool Correlat	0.27	0.00	0.00	0.00	0.27-0.28
Average	0.25	0.00	0.03	0.01	0.24-0.27
Inner Product	0.13	0.00	0.00	0.00	0.13-0.13

5.13, and 5.16 show the number of correctly classified samples for each of the bin counts using scenarios 1, 2, and 3 from section 5.3.4. Figures 5.11, 5.14, and 5.17 show the same graphs with a average smoothing filter applied to allow more readability. Scenarios 1 and 2 show drastic changes in the first 100 bin sizes and is shown in figures 5.12 and 5.15. Table 5.13 shows the bin size that gives the highest number of correct classifications for each metric.

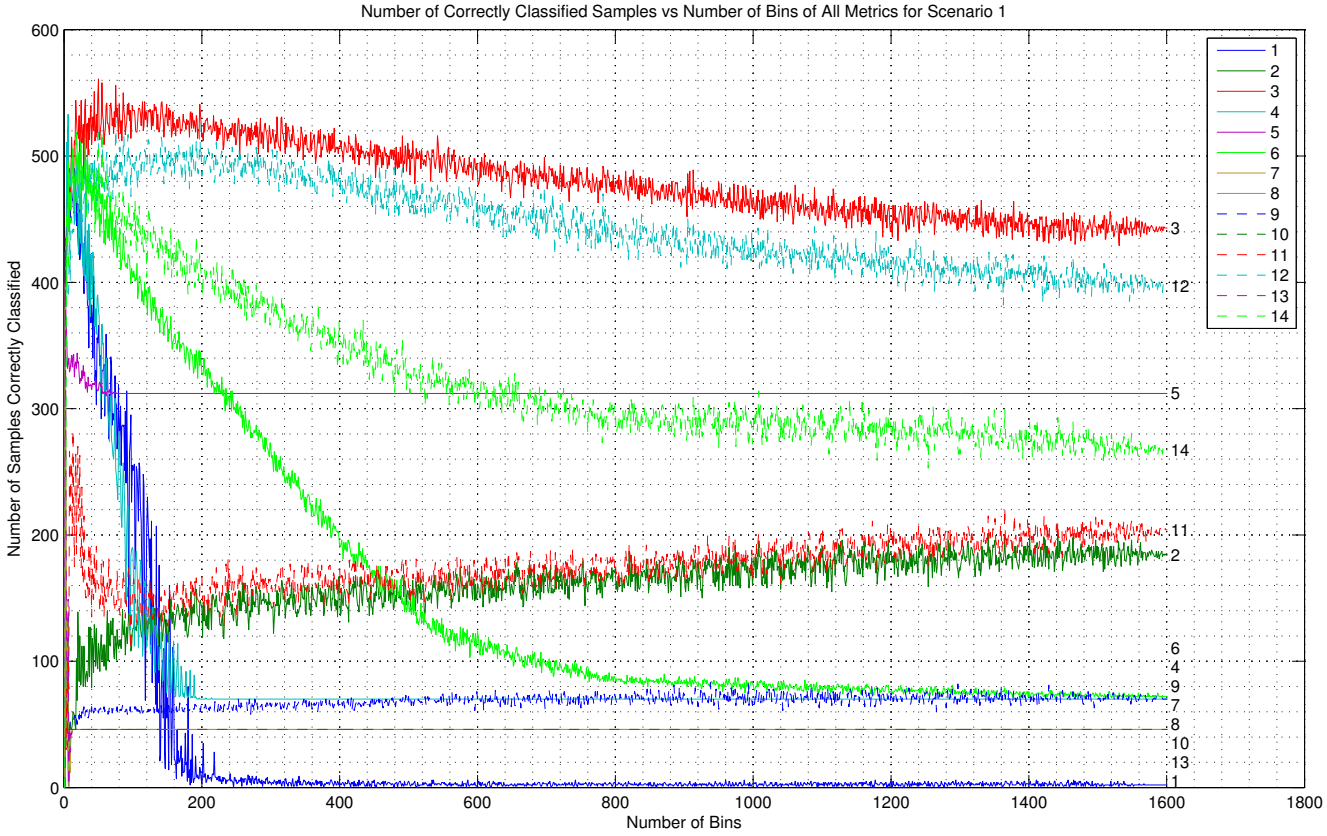


Figure 5.10: Number of correctly classified samples vs number of bins used for classification using scenario one from section 5.3.4. The line number to metric mapping can be found in table 5.12

Table 5.12: Mapping for line numbers to comparison metrics in figures.

Line Number	Metric
1	Mahalanobis
2	Inner Product
3	Euclidean
4	Manhattan
5	Average
6	Squared Chord
7	Canberra
8	Coefficient of Divergence
9	Modified Boolean Correlation
10	Average Weight of Shared Terms
11	Overlap
12	Cosine
13	Similarity Index
14	Tanimoto

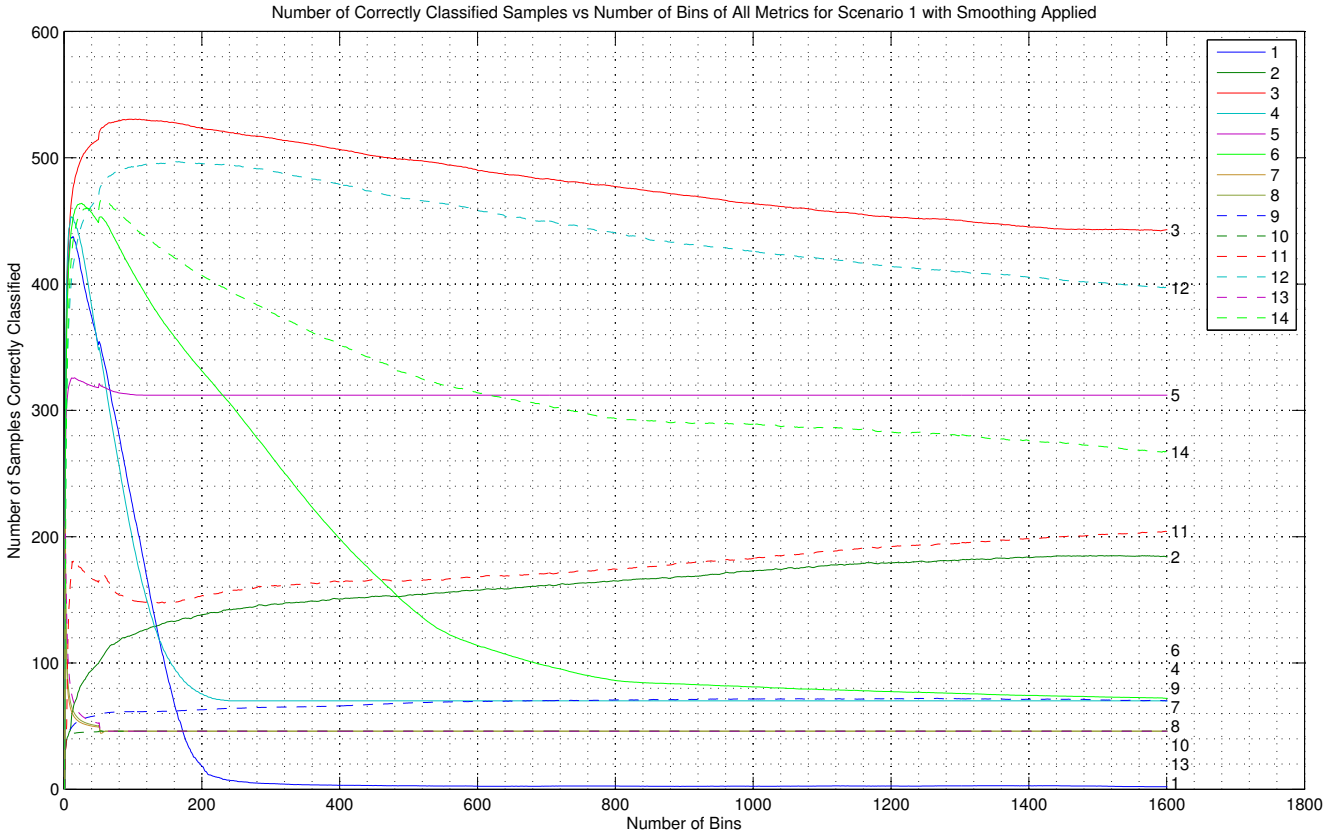


Figure 5.11: Number of correctly classified samples vs number of bins used for classification using scenario one from section 5.3.4 with a moving average of 100 applied. The line number to metric mapping can be found in table 5.12

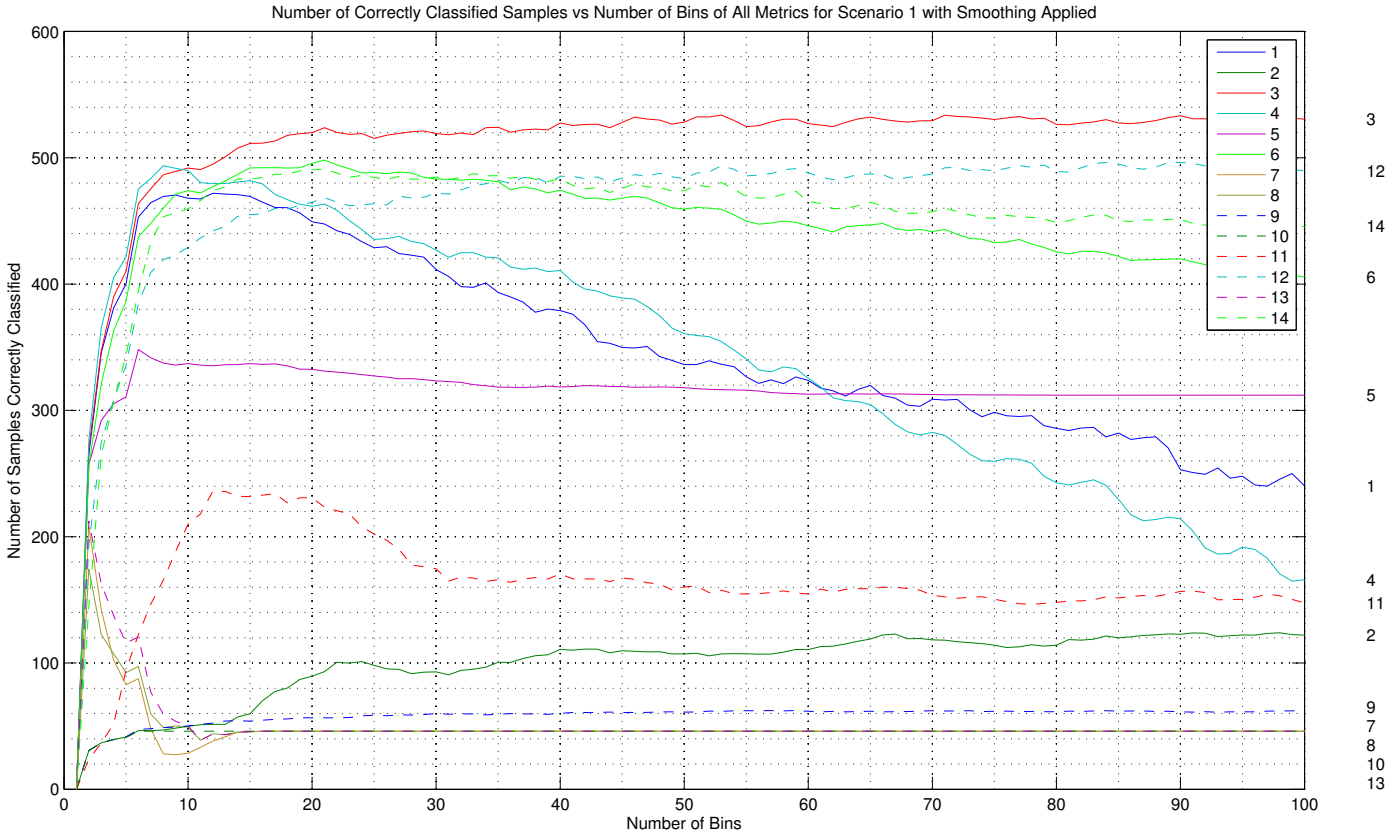


Figure 5.12: Number of correctly classified samples vs number of bins used for classification using scenario one from section 5.3.4 zoomed to show the first 100 bin counts with a moving average of 10 applied. The line number to metric mapping can be found in table 5.12

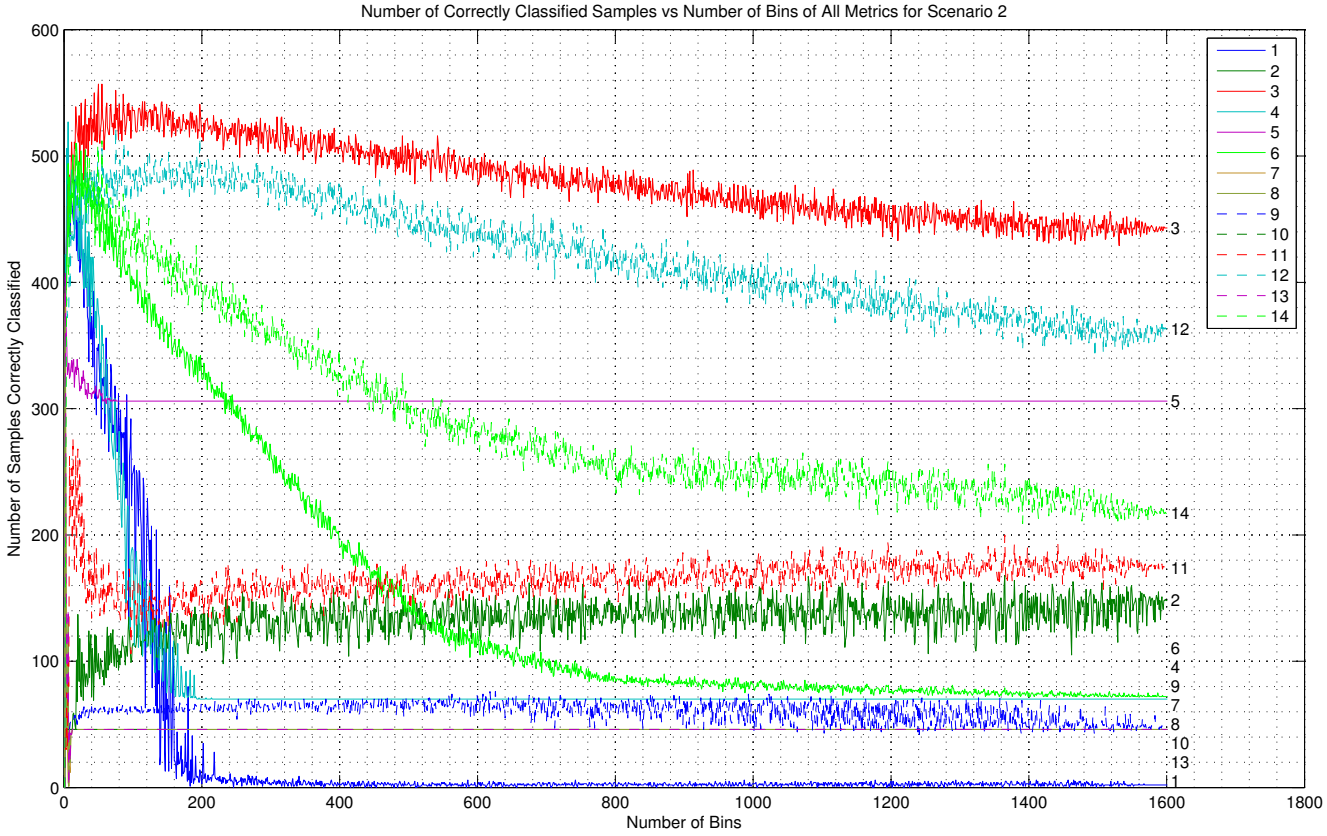


Figure 5.13: Number of correctly classified samples vs number of bins used for classification using scenario two from section 5.3.4. The line number to metric mapping can be found in table 5.12

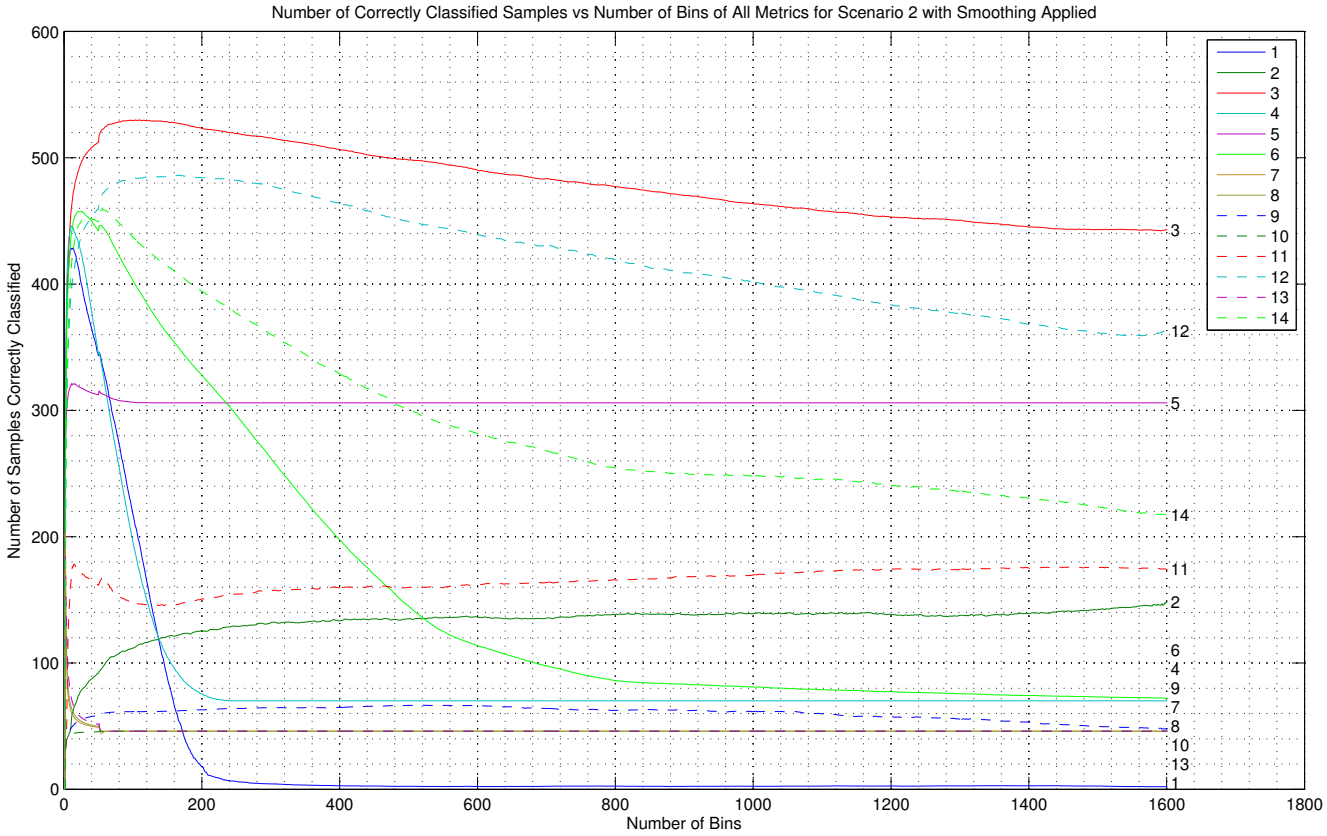


Figure 5.14: Number of correctly classified samples vs number of bins used for classification using scenario two from section 5.3.4 with a moving average of 100 applied. The line number to metric mapping can be found in table 5.12

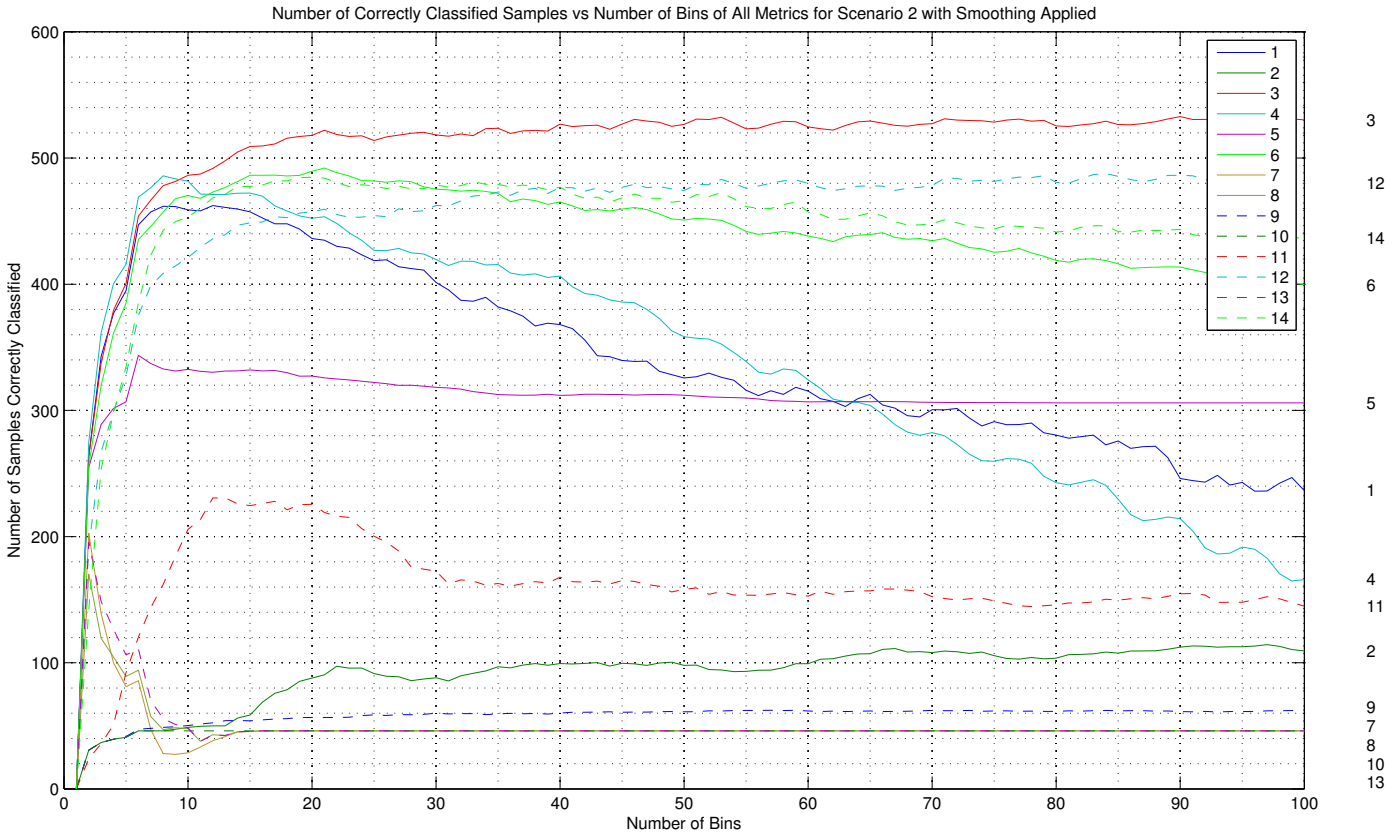


Figure 5.15: Number of correctly classified samples vs number of bins used for classification using scenario two from section 5.3.4 zoomed to show the first 100 bin counts with a moving average of 10 applied. The line number to metric mapping can be found in table 5.12

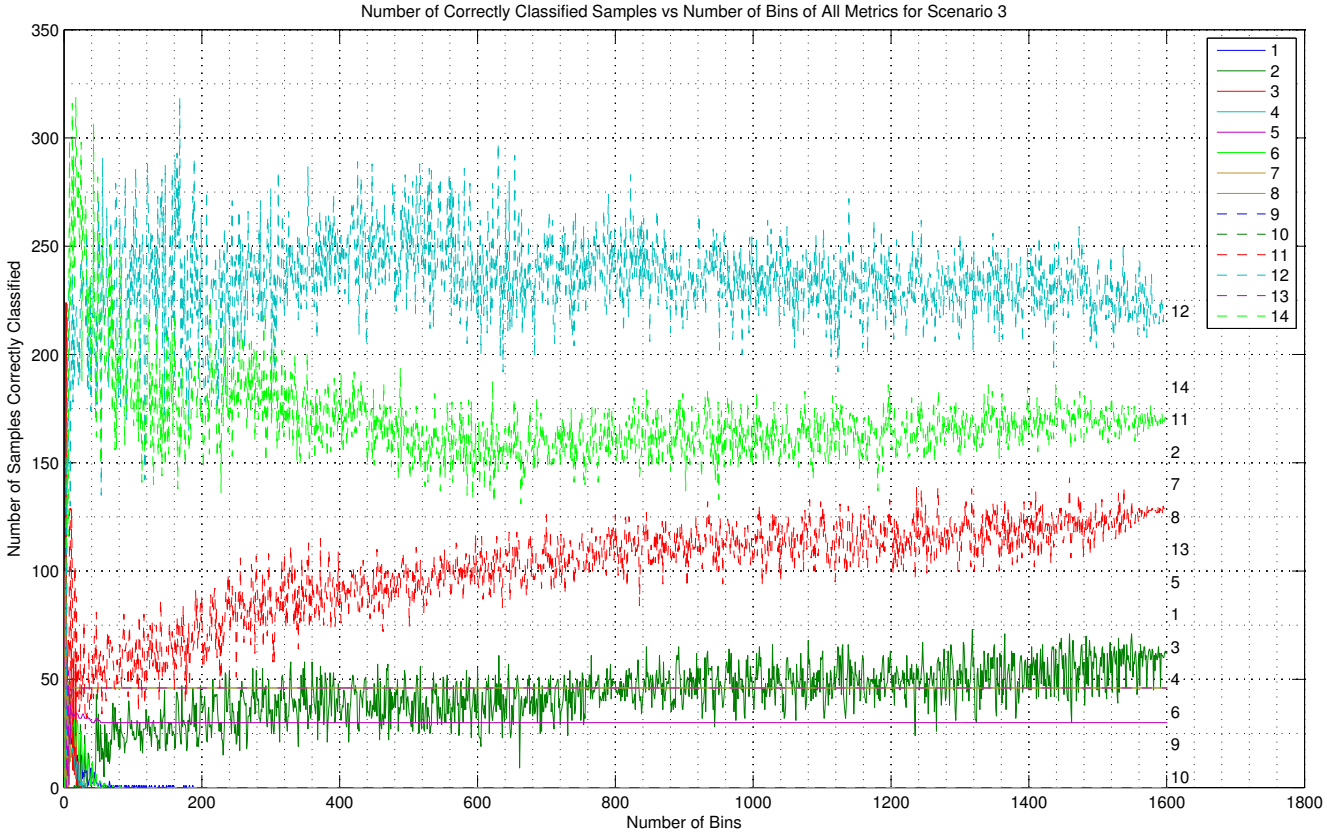


Figure 5.16: Number of correctly classified samples vs number of bins used for classification using scenario three from section 5.3.4. The line number to metric mapping can be found in table 5.12

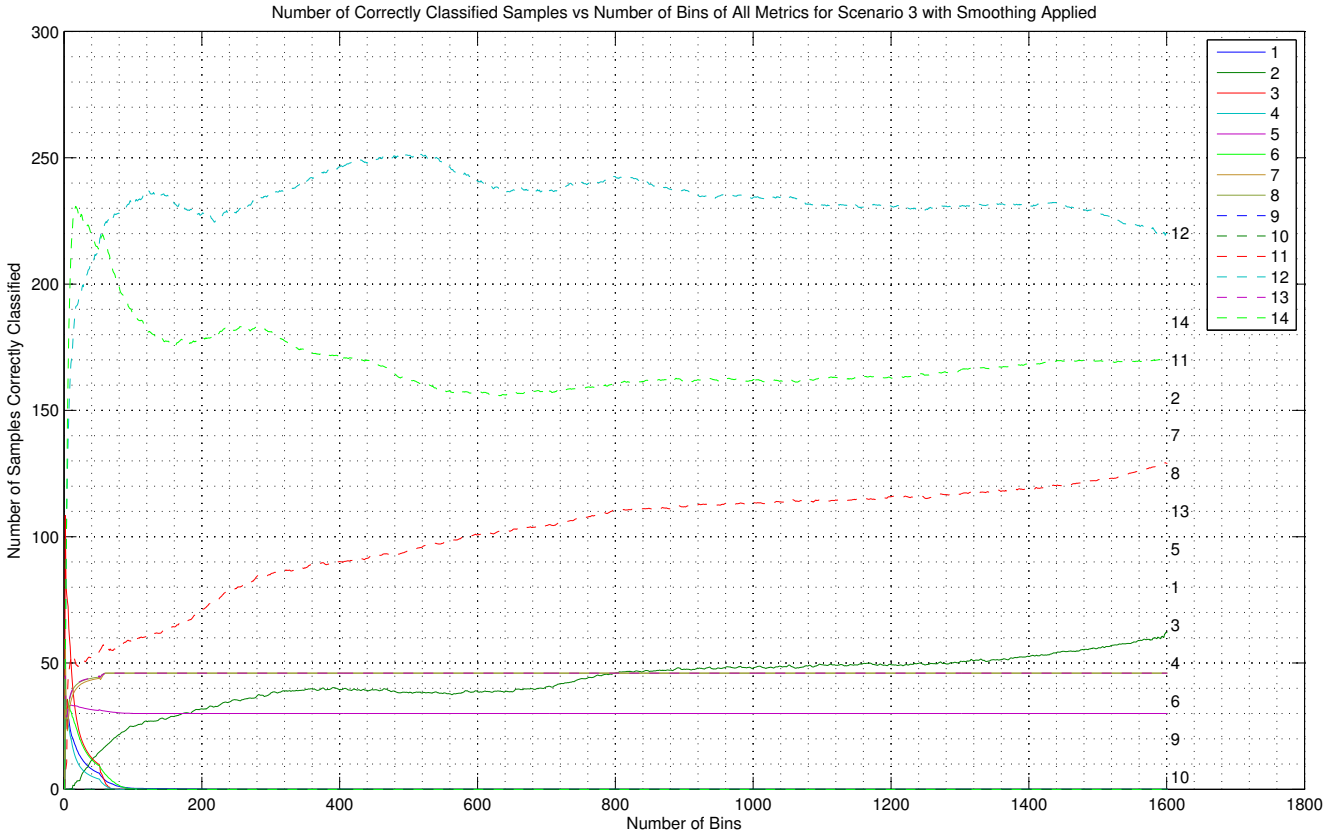


Figure 5.17: Number of correctly classified samples vs number of bins used for classification using scenario three from section 5.3.4 with a moving average of 100 applied. The line number to metric mapping can be found in table 5.12

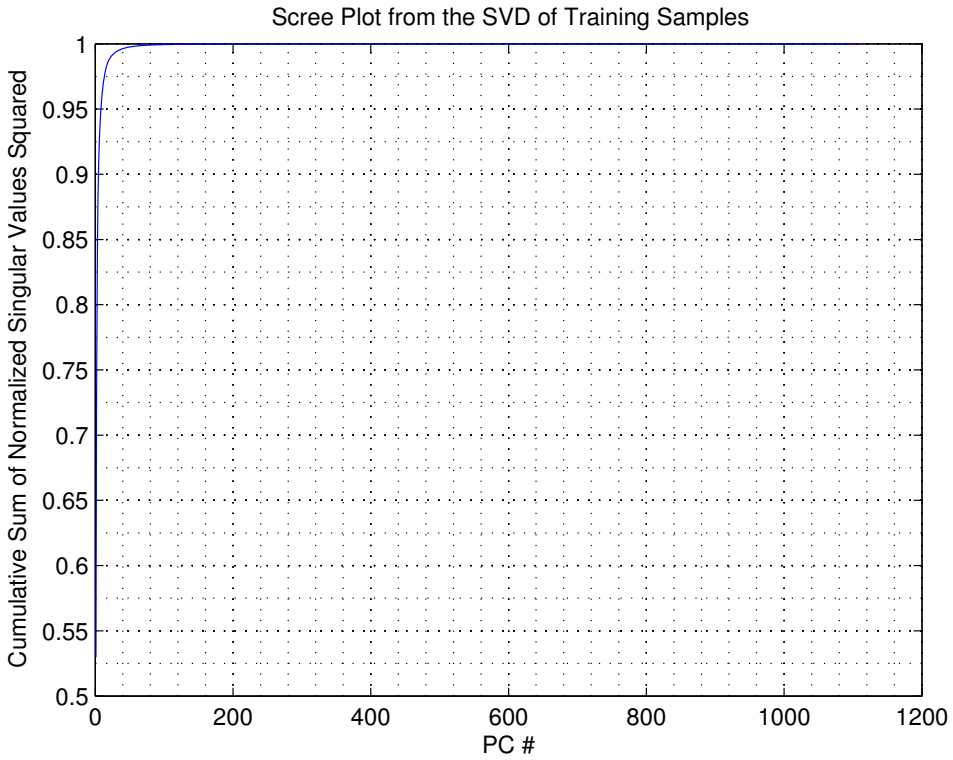


Figure 5.18: Scree plot from SVD of training samples.

5.6 Dimension Reduction Using SVD

The results of using SVD to reduce the number of dimensions for classification are detailed here. First, all of the training data samples were combined into one matrix where the rows are samples and the columns are complex impedance values as a function of frequency values. The mean value of each column of the training samples was calculated, stored, and subtracted from each training sample's value at the corresponding frequency. The SVD function in MATLAB was used to decompose the mean-centered training data matrix, X, into $U\Sigma V'$. To begin analyzing the results of SVD, the scree plot is shown in figure 5.18. The scree plot shows the sum of the ratios of the squares of the singular values to the sum of the squares of all the singular values. The first 4 loadings vectors are shown in figure 5.19.

Table 5.13: Optimal number of dimensions to keep from binning for classification using each comparison metric. Change is with respect to results from multiple runs using all 1601 frequencies.

Change	Metric	Optimal # of Dimensions	# of Correct Classifications
↑ **4**	Euclidean	50	558 (76%)
	Manhattan	6	525 (71%)
↑ **9**	Tanimoto	17	511 (69%)
↑ **5**	Squared Chord	18	511 (69%)
↓ **1**	Cosine	75	508 (69%)
↑ **2**	Mahalanobis	6	484 (66%)
↓ **6**	Canberra	2	425 (58%)
↓ **5**	Similarity Index	2	413 (56%)
↑ **2**	Average	2	397 (54%)
↓ **4**	Coe of Divergence	2	372 (50%)
↓ **4**	Overlap	13	276 (37%)
↓ **2**	Inner Product	548	181 (25%)
↑ **1**	Mod Bool Correlat	440	79 (11%)
↓ **1**	Avg Weight	2	46 (6%)

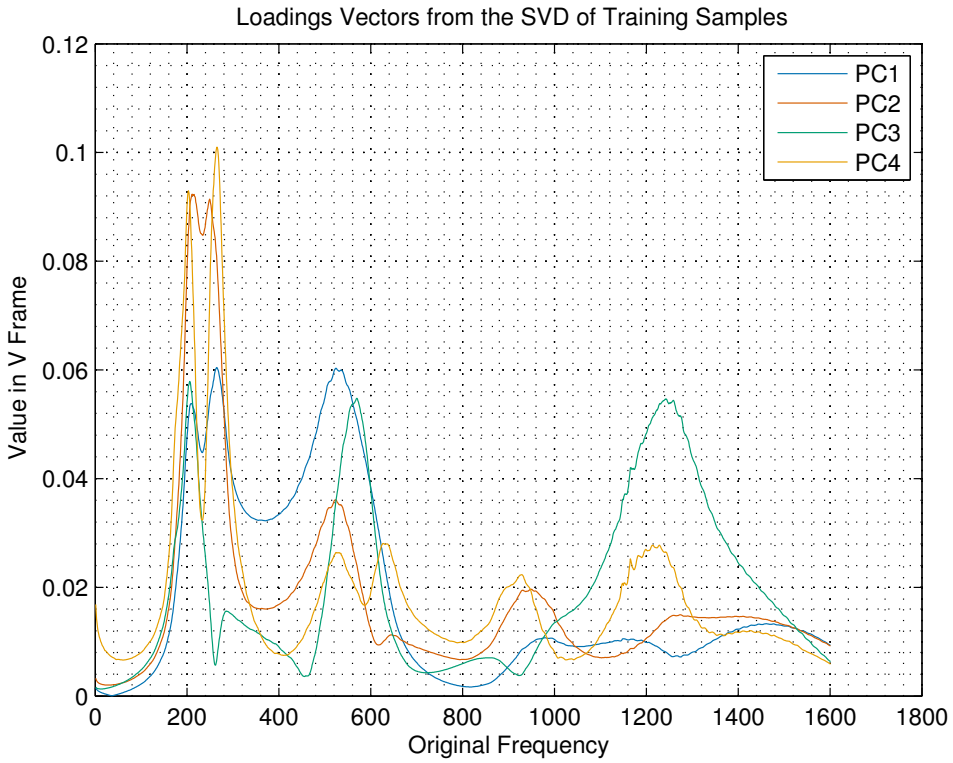


Figure 5.19: The first 4 loadings vectors from the SVD of training samples.

Figure 5.20 shows the scores of the samples in a group projected on the first two principal component vectors, with the scores for each group shown in a difference color. Figure 5.21 shows the same scores but with the first three principal components.

Dimension reduction using SVD was performed by using only the first N columns of the scores matrix, which reduces the data to the projection of the data points in 1601 dimensions down to the N-dimension subspace spanned by V. N represents the number of dimensions to use. Each column mean of the training data was subtracted from the corresponding test sample data set, followed by post-multiplying by the V_r matrix from the traning data matrix to obtain the scores for the test samples. Classification was achieved by representing each sample point with only their scores in the first N principal components and is shown in figure 5.22 for each comparison metric. Some of the curves change direction sharply around

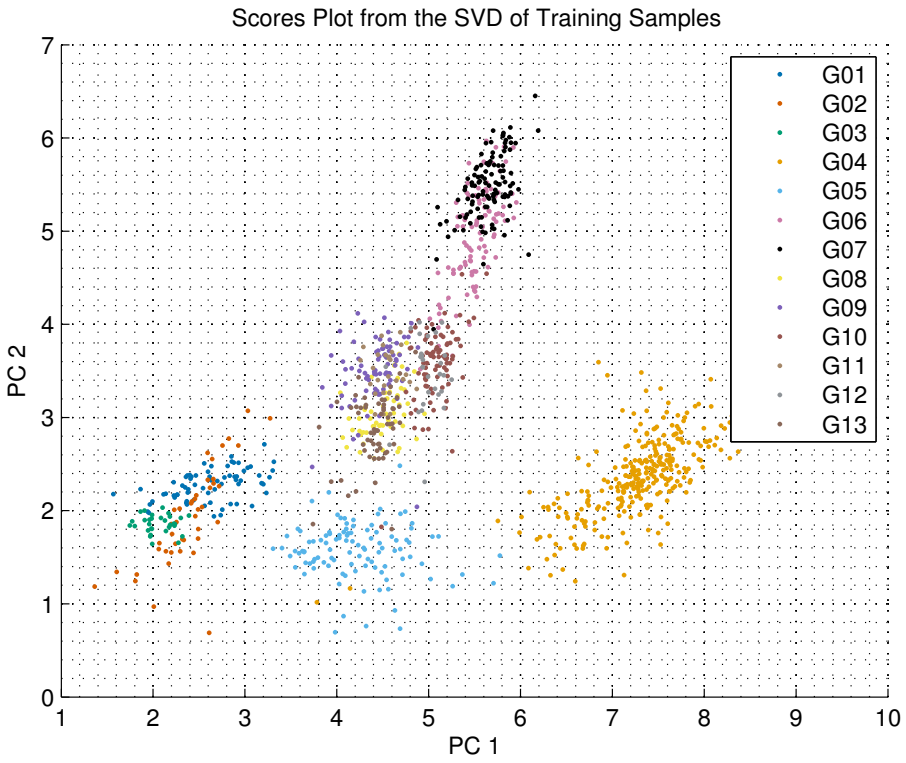


Figure 5.20: Plot of the first 2 columns of the scores matrix from the SVD of training samples.

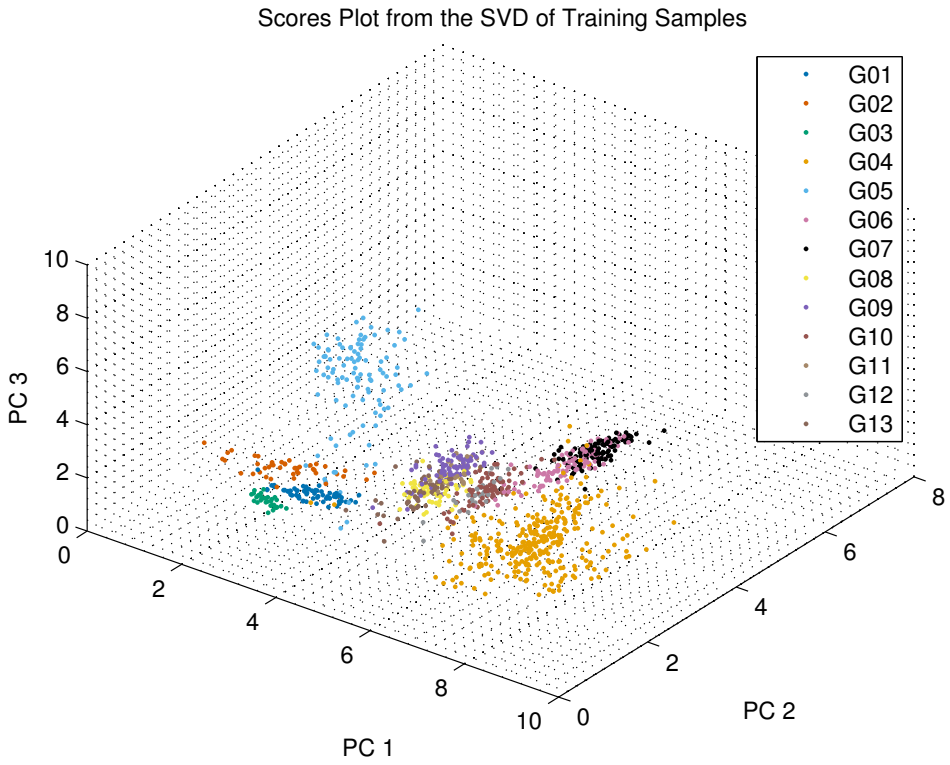


Figure 5.21: Plot similar to figure 5.20 but using the first 3 columns of the scores matrix from the SVD of training samples.

the 50-100 dimensions used and is shown in figure 5.23. The mapping of line numbers to comparison metric is in table 5.12.

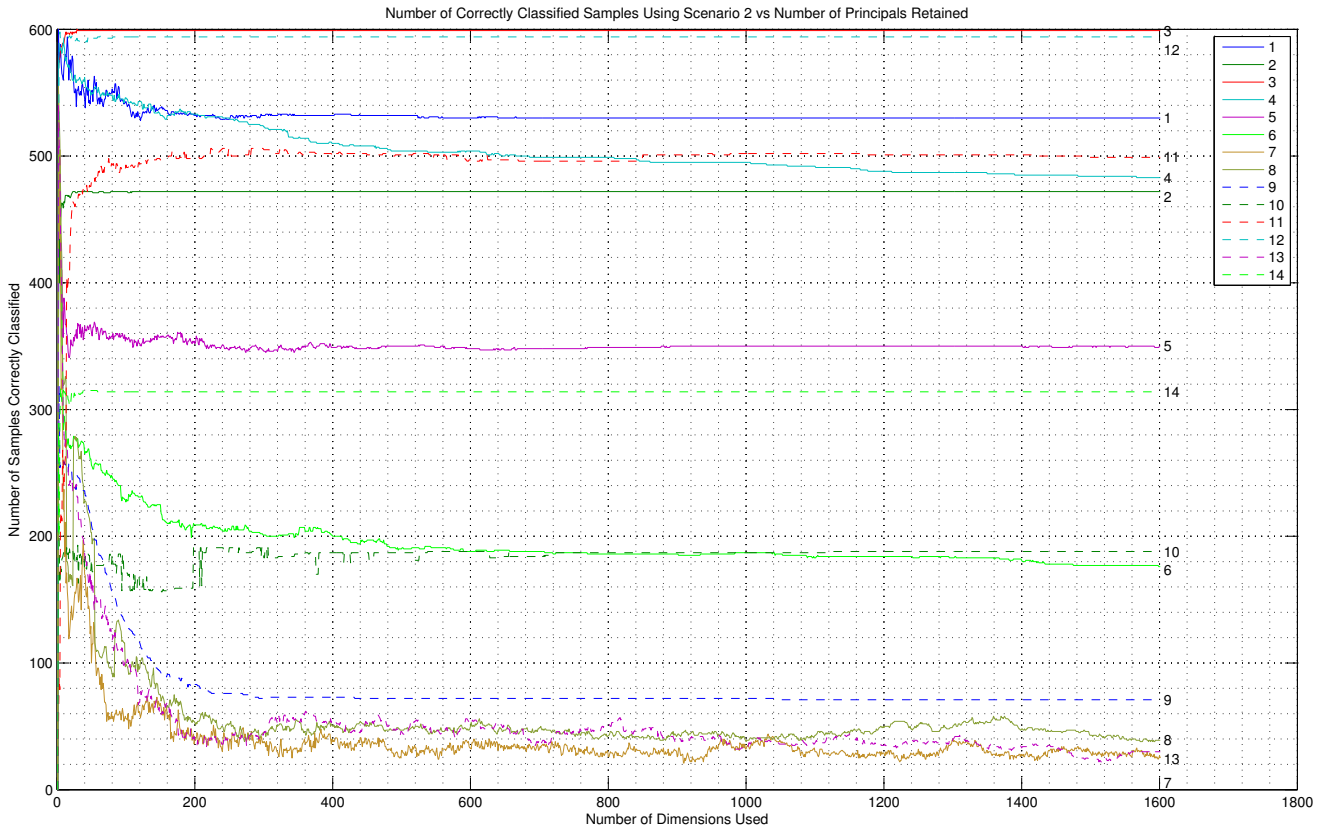


Figure 5.22: Number of correctly classified samples using scenario two from section 5.3.4 vs number of columns of the scores matrix from SVD used for classification. The line number to metric mapping can be found in table 5.12

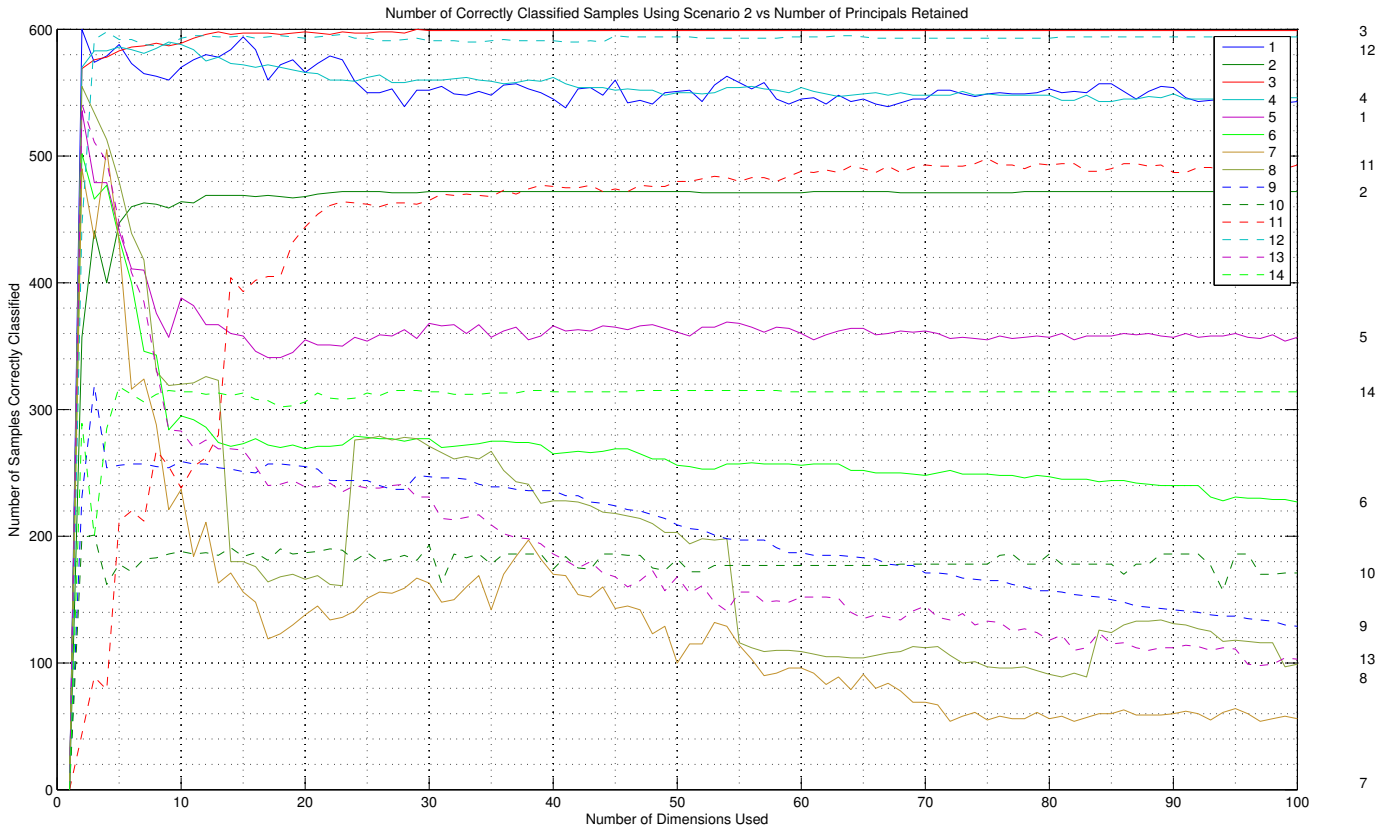


Figure 5.23: Number of correctly classified samples using scenario two from section 5.3.4 vs number of columns of the scores matrix from SVD used for classification. Only the first 100 dimensions used are shown. The line number to metric mapping can be found in table 5.12

From figure 5.22, it is shown that different metrics have different optimum number of dimensions to use. Table 5.14 shows the minimum dimension with the largest number of correctly classified samples.

Table 5.14: Optimal number of dimensions to keep from SVD for classification using scenario two from section 5.3.4 and using each comparison metric.

Change	Metric	Optimal # of Dimensions	# of Correct Classifications
↑ **7**	Mahalanobis	2	600 (80%)
↑ **3**	Euclidean	29	600 (81%)
↑ **1**	Cosine	4	598 (80%)
↓ **2**	Manhattan	9	590 (80%)
↑ **1**	Coe of Divergence	2	555 (44%)
↓ **3**	Similarity Index	2	541 (46%)
↑ **4**	Average	2	536 (76%)
↓ **1**	Overlap	223	506 (64%)
↓ **8**	Canberra	4	505 (43%)
↓ **1**	Squared Chord	2	502 (72%)
↓ **1**	Inner Product	23	472 (69%)
↑ **2**	Mod Bool Correlat	3	319 (35%)
↓ **1**	Tanimoto's	5	318 (42%)
↓ **1**	Avg Weight	3	201 (28%)

Chapter 6

Discussions

6.1 Model Analysis

Determination of an expanded circuit impedance equation and transfer function did not assist in improving the effectiveness of classification of the impedance data. The results do show that it is possible to estimate a transfer function that contains many of the same features of the measured data.

6.1.1 Bode Plot of Data vs Estimated Transfer Function Bode Plot

The bode plot of an estimated transfer function, calculated from the measured data, in figure 5.2 shows some similarity to the bode plot of the measured data in figure 5.2. The phase in the lower frequencies is much lower and takes a different approach in the estimated transfer function bode plot. The corner frequencies occur at the same frequencies in both graphs and the magnitude follows the same general shape.

6.2 Similarity/Dissimilarity Metric Performance

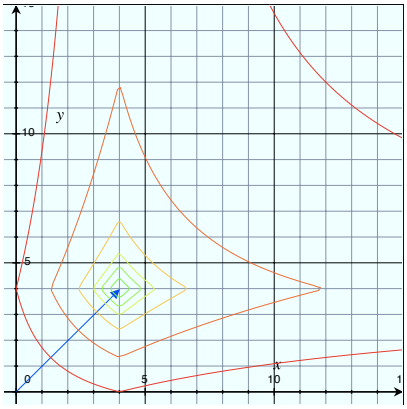
The results of performance of the similarity/dissimilarity metrics varied from very good to very poor. After classification, the results show that some of the metrics are not fit for this type of data. When using all 1601 frequencies for classification, Canberra, Manhattan, Similarity Index, Cosine, Euclidean, Overlap, and Coefficient of Divergence did very well (see section 6.2.1 for analysis) with Canberra and Manhattan consistently with the most samples correctly classified (see section 6.2.2 for analysis). Average Weight, Tanimoto's, and Modified Boolean Correlation did a very poor job of classification (see section 6.2.3 for analysis). The Mahalanobis distance did not classify as well as expected and was not a top performer (see section 6.2.4 for analysis).

Repetition of classification using different random permutations of training/test samples showed that most metrics had similar classification results across repetitions, with the exception of the Mahalanobis distance (see section 6.2.5 for analysis).

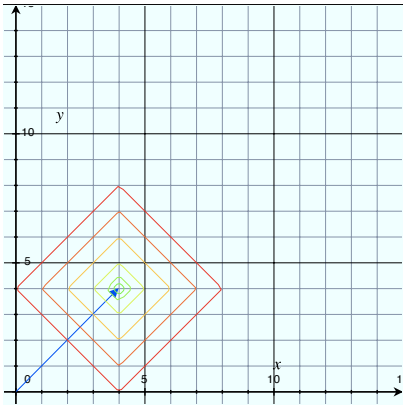
The amount of time required for calculation of most of the metrics was very similar, with the Mahalanobis distance being the only exception and taking much longer for execution (see section 6.2.6 for analysis).

6.2.1 Best Classification Metrics

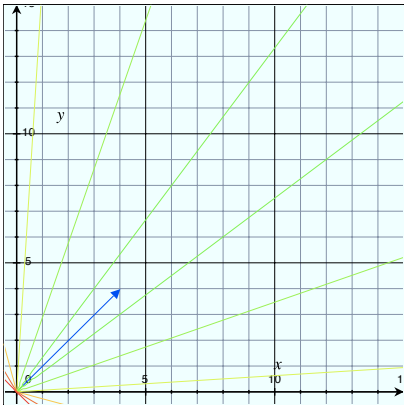
Table 5.10 shows the metrics Canberra, Manhattan, Cosine, Euclidean, Similarity Index, Overlap, and Coefficient of Divergence consistently classifying 76-86% of the test samples into the correct groups. Figure 6.1 shows contour graphs of the metrics from section 2.3. These metrics all share the property that as the vectors get further apart in the NorthWest or SouthEast direction, the value of similarity decreases. This group also includes all of the metrics that were bounded with a minimum and maximum similarity: Cosine, Normalized Euclidean, and Coefficient of Divergence.



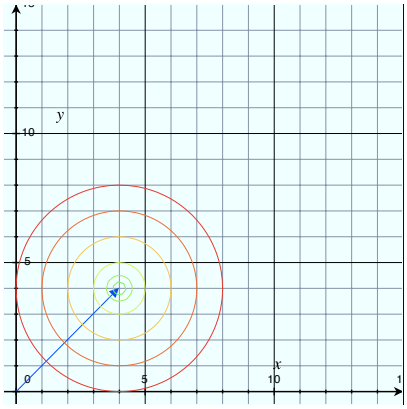
(a) Canberra



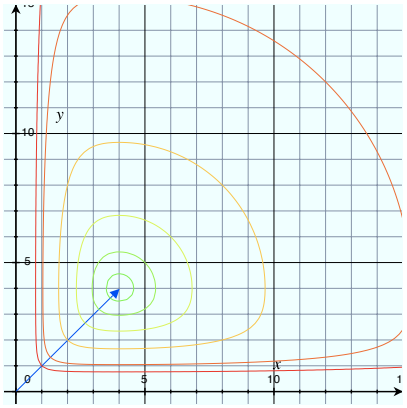
(b) Manhattan



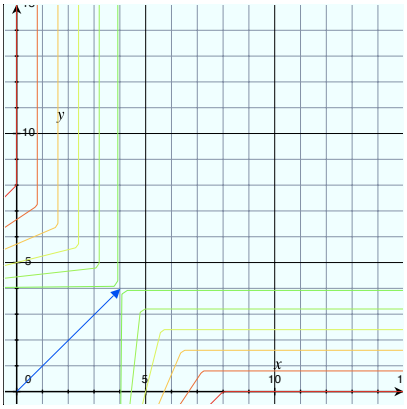
(c) Cosine



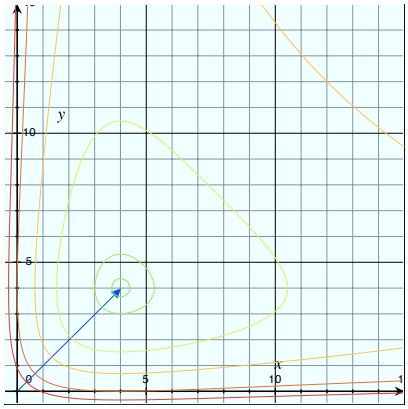
(d) Euclidean



(e) Similarity Index



(f) Overlap



(g) Coefficient of Divergence

Figure 6.1: Contour graphs of the top classification metrics using 1601 frequencies.

6.2.2 Canberra and Manhattan Results

Tables 5.10, 5.6, 5.7, and 5.8 all consistently show the Canberra and Manhattan distances classifying the most correct samples in all three scenarios using all 1601 frequencies in the normal frame. Figure 6.1 shows the contour graphs of both metrics. The contour graphs show that both metrics have a similar diamond, or four-sided star, shape. The Canberra contours look almost as if the Manhattan contours were stretched out parallel to the axis.

These contours show that a sample can have a high similarity value to another sample if at least one of the dimensions is very similar. If one of the dimensions is similar, flexibility is given on the other dimension to be larger for the Canberra distance. In multiple dimensions with the data used in this research, all samples in the same group have most of their dimensions very similar with only a few values at the peaks showing the most variability.

6.2.3 Poor Classification Metrics

Classification using the Average Weight, Tanimoto's, and Modified Boolean Correlation metrics yielded results that were less than desirable for a classification system. The root causes were different for each of the metrics.

Details of the average weight metric can be found in section 2.3.10 and show that the distance is the average of all of the positive values in the two vectors. When working with complex numbers, the definition of a positive number is difficult to define. For the purposes of this research, all of our values were defined to be positive, even though some of the imaginary values were negative. Ignoring the problem with determining if a number is positive, this metric still has other problems. If one of the vectors has very large values, the similarity value is very large, even if they are very dissimilar. This metric would need to be modified to use a range around the average of the centroid.

Details of Tanimoto's metric can be found in section 2.3.14. The exact cause of Tanimoto's poor performance has not been determined; however, it may be due to the non-linear changes in similarity as the distance from the centroid increases.

Details of the Modified Boolean Correlation metric can be found in section 2.3.9, and the equations are intended for Boolean values of zero or one in each dimension of the vector. This requirement is evident in the classification results using the metric.

6.2.4 Mahalanobis Distance Underperformance

Before results were obtained, the Mahalanobis distance was expected to perform rather well in the classification of the samples. Instead, it performed worse than many of the metrics. The cause of the problems was most likely the large number of vector values that are close to zero in all of the samples. Analysis of the dimension reduction results show that the Mahalanobis distance did much better when only a few dimensions with high variability were used.

6.2.5 Mahalanobis Distance Variability Between Reps

The Mahalanobis distance requires the inverse of the covariance matrix formed from the training samples. The covariance matrix is ill-conditioned in most cases causing small changes in the covariance matrix to have a large impact on the inverse of the matrix. This ill-conditioned property causes some small changes in the training samples to have a big impact on the classification. This problem was addressed by calculating the pseudo-inverse using SVD and reducing rank by setting the singular values of less than 1/1000th of the maximum singular value to zero.

6.2.6 Mahalanobis Distance Computation Time

The most computationally intensive part of the Mahalanobis distance is the inverse of the covariance matrix. The inverse calculation can be very costly and is heavily dependent on the number of dimensions used and the implementation algorithm. Time was saved by storing the inverse and only recalculating the inverse when a new set of training samples exist. In a production implementation, the recalculation would be set to be conducted on a routine basis when utilization is low.

6.3 Unclassifiable Determination Scenarios

Three scenarios for determination if a sample is unclassifiable were researched. The results of the scenarios are shown in tables 5.6, 5.7, and 5.8. In scenario 1, all test samples were classified to the most similar group centroid (see section 6.3.1 for analysis). In scenario 2 and 3, a threshold was determined for each group for each metric that minimizes the classification error using the training data. Scenario 2 places a test sample in the unclassifiable group if the similarity value to the most similar group is not above the threshold for that group (see section 6.3.2 for analysis). Scenario 3 expands scenario 2 by also classifying a sample as unclassifiable if it has similarity values above the threshold for more than one group (see section 6.3.3 for analysis).

6.3.1 Classification using Scenario 1

Table 5.6 shows the results of all metrics using scenario 1 for classification and table 5.2 shows the classification confusion matrix using scenario 1 with the cosine similarity metric. Scenario 1 gives the largest percentage of correctly classified test samples, but also gives the highest percentage of incorrectly classified test samples for most of the metrics. The classification confusion matrix shows that the classification algorithm has a difficult time

distinguishing between groups 6 and 7 and also between groups 8-13 because the spectra are too similar.

6.3.2 Classification using Scenario 2

Scenario 2 builds on scenario 1 by eliminating the test samples that do not exceed the threshold for their classification group. The number of incorrectly classified test samples decreased for most metrics, but the number of correctly classified also decreased. Comparing the classification confusion matrices of scenario 1 and 2, the incorrectly classified samples of groups 2 and 4 were moved to unclassifiable, but some of the correctly classified samples in groups 1, 4, and 5 were also moved to unclassifiable. The classification of groups 6-13 were mostly unchanged.

This scenario did not have as much of an effect on this data set because the groups were well defined with most of the samples matching the same general shape of their centroid. If samples that were improperly measured or incorrectly classified were included in the test samples, they would show up using this scenario.

6.3.3 Classification using Scenario 3

Scenario 3 had a drastic affect on the results of classification with all of the metrics. Table 5.6 shows the results of classification using scenario 3 with each of the metrics. In all metrics, the number of incorrectly classified became almost zero. However, the number of correctly classified samples also dropped significantly.

Table 5.4 shows the classification confusion matrix using scenario 3 and the cosine similarity function. Comparing the matrix to the scenario 1 classification confusion matrix in table 5.2, it can be seen that this scenario only affects the groups that are very similar to each other (Groups 6-13).

Table 5.4 also shows that when the sample matches multiple groups, one of the groups is most often the correct group. The classification system could give as a result a subset of the groups that the sample may belong to.

This scenario could be very useful in circumstances where the number of incorrectly classified samples is more important than the number of correctly classified samples. A good example is the medical field where it would more appropriate to determine that a sample is unclassifiable, than to incorrectly classify the sample.

6.4 Dimension Reduction Using Binning

The results of using binning for dimension reduction have shown that binning is not an effective method of reducing the number of dimensions in the spectral data. A comparison of the optimal number of bins vs not using binning shows most metrics performing worse after binning (see section 6.4.1 for analysis). The threshold scenarios have the same effect on the binned data as they do on the non-reduced, non-binned data (see section 6.4.2 for analysis).

6.4.1 Binning vs Non-Binning Results

Table 5.13 shows the bin size with the highest number of correct classifications. These values are less than the number of correct classifications obtained from using all 1601 frequencies. When using 1601 bins, the only difference between the data and non-reduced data is that only the peak frequencies contain values. The fact that this scenario has less samples correctly classified shows that frequencies without peaks contain information vital to correct classification.

6.4.2 Binning Effect on Threshold Scenarios

Figures 5.10, 5.13, and 5.16 show the results of every bin size using scenarios 1, 2, and 3 for determination if a sample is unclassifiable. These results match the same respective change in number of correctly classified samples as the comparison using all 1601 frequencies. Scenario 1 and 2 are very similar and scenario 3 has fewer correctly classified samples.

6.5 Dimension Reduction Using SVD

Dimension reduction using SVD appears to be a promising method of data reduction. The overall groupings of good/ok/poor metrics did not change but some metrics did perform better than others after reduction using SVD (see section 6.5.1 for analysis). The number of dimensions to keep after transformation with SVD depends on the metric but is always much less than all 1601 dimensions (see section 6.5.2 for analysis).

6.5.1 Comparison With and Without Reduction

Table 6.1 shows the ranking of the classification performance of all metrics using all 1601 frequencies and the ranking of metrics from their most effective dimension of reduction using SVD. Some of the metrics did significantly better at their optimal dimension reduction and some did worse.

The metrics that performed better using dimension reduction with SVD have some properties in common. Most of the well-performing metrics have elliptical contours of equivalent similarity distance.

6.5.2 Optimal Dimension for Classification

Table 5.14 shows the optimal dimension to use for classification of SVD transformed data. However, analysis of the table by itself isn't enough to determine the most effective number

Table 6.1: Ranking of similarity/dissimilarity metrics

Rank	Full 1601 Dimensions	Reduction by SVD Optimal Dimension
1	Canberra	Euclidean
2	Manhattan	Tanimoto's
3	Cosine	Inner Product
4	Euclidean	Mahalanobis
5	Similarity Index	Modified Boolean Correlation
6	Overlap	Overlap
7	Coefficient of Divergence	Manhattan
8	Squared Chord	Canberra
9	Inner Product	Similarity Index
10	Mahalanobis	Cosine
11	Average	Average
12	Average Weight	Squared Chord
13	Tanimoto's	Average Weight
14	Modified Boolean Correlation	Coefficient of Divergence

of dimensions to keep. Figure 5.22 shows number of correctly classified samples for all dimensions. For some of the metrics, they stabilize at dimensions higher than that of the highest correctness. For example, Euclidean and Cosine would be much more effective at 25 dimensions to work with the variability in future samples that is not encompassed in this research.

Chapter 7

Recommendations

Analysis of data for this research has shown that certain metric and data dimension reduction methods will result in better classification under specific situations. Several choices must be made to optimize the classification system for accuracy, complexity, and computational time requirements. The following sections outline the benefits for each of the options in the classification system.

7.1 Choice of Similarity/Dissimilarity Metric

The choice of comparison metric will depend on if dimension reduction with SVD is used and the desired speed of classification.

When using all dimensions of the data without any dimension reduction or rotation, tables 5.6, 5.7, and 5.8 show that regardless of the scenario, Canberra distance is the metric to use. However, if speed is a concern, the Manhattan distance has comparable performance results and requires only half the calculations of the Canberra distance. Table 2.1 shows the equations required for Canberra and Manhattan respectively.

When using dimension reduction with SVD, table 5.14 shows that Euclidean distance provides the best results with the least number of dimensions. The Euclidean distance requires relatively short computation time in comparison to the other metrics.

7.2 Choice of Unclassifiable Scenario

The three scenarios for unclassifiable determination of a sample have utility in applications of different requirements. Scenario one would be used when there is a requirement to always classify an unknown sample to a group. Scenario two would be used when the system may have unknown samples that do not belong to any of the groups currently trained in the system. Scenario three would be used when incorrectly classified samples could have a big impact. A field that might use scenario three is medical.

7.3 Classification Steps

The choices from section 7.1 and 7.2 can be combined to form the algorithm for classification. A flow chart for a classification system using the decision process from this research is shown in figure 7.1. Pseudocode for a classification system based on the results of this research and the flow chart is also included below.

```
Sample = Read_Sample();
IF Sample is a Reference Sample {
  Store_Spectrum(Sample);
  Queue System to recalculate training data;
 } ELSE Sample is an Unknown Sample to be classified {
  Get stored group centroids
  IF Dimension Reduction is being used {
```

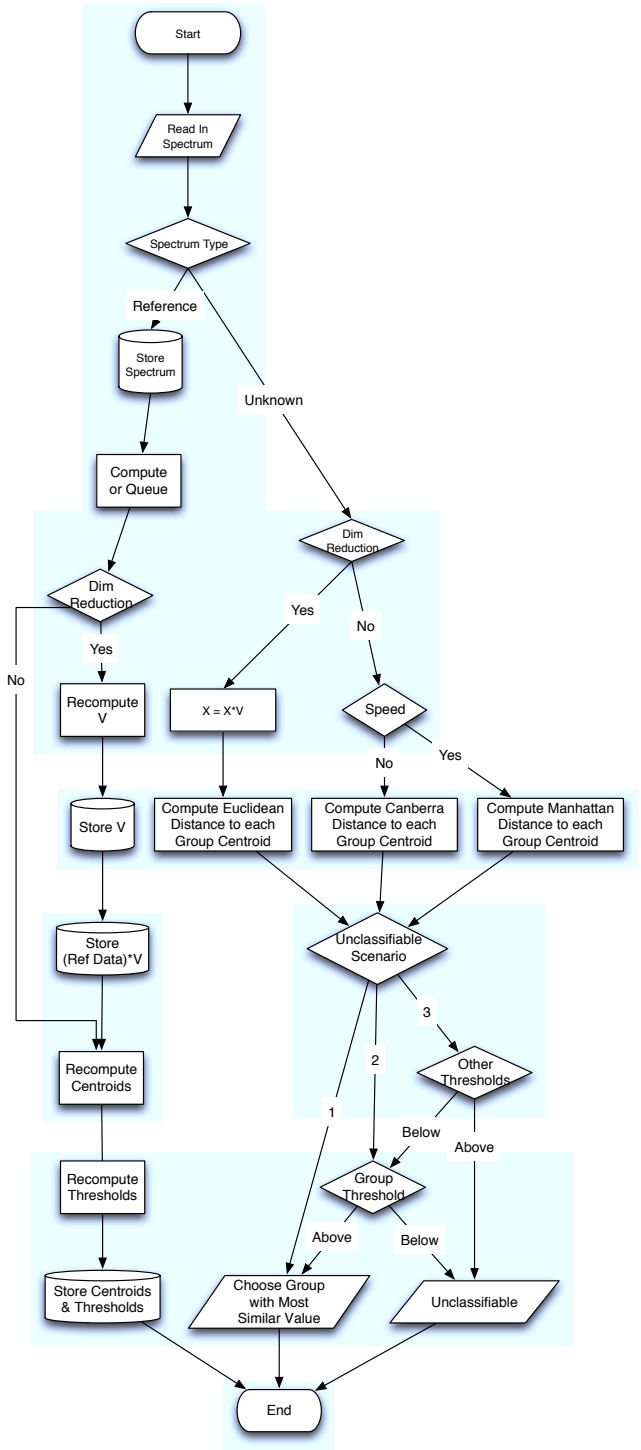


Figure 7.1: Flow chart for a classification system based on this research.

```
  Get stored V from SVD
  Sample = Sample * V
  Compute Euclidean distance between Sample and group centroids
 } ELSE Dimension Reduction is not being used {
  IF Speed is a concern {
   Compute Manhattan distance between Sample and group centroids
  } ELSE Accuracy is more important {
   Compute Canberra distance between Sample and group centroids
  }
 }
IF Unclassifiable Scenario is 1 {
  OUT = Most similar group centroid
  END
 } ELSE Scenario is 2 or 3 {
  IF Scenario is 3 {
    IF Similarity value is above threshold for more than 1 group {
      OUT = Unclassifiable
      END
     }
   }
  IF Similarity value is below the threshold for most similar group {
    OUT = Unclassifiable
    END
   } ELSE {
    OUT = Most similar group centroid
    END
   }
```

```
   }
 }

FUNCTION Recalculate_Training_Data {
  All_Reference_Samples = Stored_Reference_Samples combined with new Sample
  IF Dimension Reduction is being used {
    All_Reference_Samples is Number_of_Samples X Number_of_Frequencies
    [U S V] = SVD(All_Reference_Samples)
    Store V matrix
    Store All_Reference_Samples*V
    All_Reference_Samples = All_Reference_Samples*V
   }
  Compute centroids of groups in All_Reference_Samples
  Store centroids
 }
```

Chapter 8

Future Work

The results of this research have shown several experiments that can be expanded into seperate research projects. Many of the comparison metrics that did very poorly are not properly formulated to handle complex values. Research can be conducted to modify these equations to find an equivalent complex valued equation that provides more effective classification of the data. The classification system can be expanded to handle determination of a new group and effectively regroup the samples using unsupervised learning. Finally, the entire system can be combined into a system with a GUI interface and efficient processing as a production system.

Bibliography

[1] G. P. Kurpis and C. J. Booth, *The New IEEE Standard Dictionary of Electrical and Electronics Terms*. New York, NY: The Institute of Electrical and Electronics Engineers, Inc., 5th ed., 1993. 3

[2] A. E. Kennelly, "Impedance," *American Institute of Electrical Engineers Transactions*, vol. 10, pp. 175–232, 1893. 3

[3] J. J. Beard, "Phasor." http://commons.wikimedia.org/wiki/File:Phasor.svg, May 2006. 5

[4] Mets501, "Polar to cartesian." http://commons.wikimedia.org/wiki/File:Polar_to_cartesian.svg, June 2007. 5

[5] G. S. Ohm, *The Galvanic Circuit Investigated Mathematically*. 1827. 5

[6] C. K. Alexander and M. Sadiku, *Fundamentals of Electric Circuits with CD-ROM*. McGraw-Hill Science/Engineering/Math, 2 ed., Jan. 2003. 6

[7] K. C. A. Smith and R. E. Alley, "Multiple resonance," in *Electrical Circuits: An Introduction*, pp. 169–172, Cambridge University Press, Jan. 1992. 8

[8] A. Technologies, "Agilent PNA-X measurement receiver N5264A data sheet and technical specifications," tech. rep., 2008. 8

[9] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, vol. 2, pp. 559–572, 1901. 9

[10] J. E. Jackson, *A User's Guide to Principal Components (Wiley Series in Probability and Statistics)*. Wiley-Interscience, Sept. 2003. 9

[11] I. T. Joliffe, *Principal Component Analysis (Springer Series in Statistics)*. Springer, 2nd ed., Dec. 2010. 9

[12] G. Dunteman, *Principal Components Analysis (Quantitative Applications in the Social Sciences)*. Sage Publications, Inc, May 1989. 9

[13] E. Puryear, D. Angulo, A. Schilling, K. Drew, and G. von Laszewski, "Comparing mass spectra," Mar. 2006. 11, 12, 34

[14] M. McGill, "An evaluation of factors affecting document ranking by information retrieval systems," tech. rep., Syracuse Univ., N.Y. School of Information Studies, Oct. 1979. 11, 12, 23, 36

[15] W. P. Jones and G. W. Furnas, "Pictures of relevance: A geometric analysis of similarity measures," *Journal of the American Society for Information Science*, vol. 38, no. 6, pp. 420–442, 1987. 12

[16] M. R. Leek, M. F. Dorman, and Q. Summerfield, "Minimum spectral contrast for vowel identification by normal- hearing and hearing-impaired listeners," *Journal of the Acoustical Society of America*, vol. 81, no. 1, pp. 148–54, 1987. 12

[17] B. Logan and A. Salomon, "A music similarity function based on signal analysis," in *ICME 2001*, (Tokyo, Japan), pp. 745–748, Aug. 2001. 12, 35

[18] S. E. Stein and D. R. Scott, "Optimization and testing of mass spectral library search algorithms for compound identification," *Journal of The American Society for Mass Spectrometry*, vol. 5, pp. 859–866, 1994. 12

[19] Z. B. Alfassi, "On the comparison of different tests for identification of a compound from its mass spectrum," *Journal of The American Society for Mass Spectrometry*, vol. 14, pp. 262–264, 2003. 12, 30

[20] T. Noreault, M. McGill, and M. B. Koll, "A performance evaluation of similarity measures, document term weighting schemes and representations in a boolean

environment," in *Proceedings of the 3rd annual ACM conference on Research and development in information retrieval*, (Cambridge, England), pp. 57–76, June 1980. 12, 29

[21] P. C. Mahalanobis, "On the generalised distance in statistics," *Proceedings of the National Institute of Science, India*, vol. 2, pp. 49–55, Apr. 1936. 15

[22] D. J. Birdwell, T. wei Wang, D. J. Icove, S. P. Horn, and M. Rader, "METHOD AND APPARATUS FOR PREDICTING OBJECT PROPERTIES AND EVENTS USING SIMILARITY-BASED INFORMATION RETRIEVAL AND MODELING," Dec. 2010. 17

[23] MathWorks, "R2009b MathWorks documention." gmdistribution.mahal. 17

[24] D. G. Gavin, W. W. Oswald, E. R. Wahl, and J. W. Williams, "A statistical approach to evaluating distance metrics and analog assignments for pollen records," *Quaternary Research*, vol. 60, pp. 356–367, Nov. 2003. 20

[25] J. T. Overpeck, T. Webb, and I. C. Prentice, "Quantitative interpretation of fossil pollen spectra: Dissimilarity coefficients and the method of modern analogs," *Quaternary Research*, vol. 23, pp. 87–108, Jan. 1985. 20

[26] G. N. Lance and W. T. Williams, "Computer programs for hierarchical polythetic classification ("similarity analyses")," *Computer Journal*, vol. 9, pp. 60–64, 1966. 22

[27] P. H. A. Sneath and R. R. Sokal, *Numerical Taxonomy: The Principles and Practice of Numerical Classification*. W H Freeman & Co (Sd), June 1973. 23

[28] W. K. H. Sager and P. C. Lockemann, "Classification of ranking algorithms," *International Forum on Information and Documentation*, vol. 1, no. 1, pp. 41–46, 1976. 24

[29] K. Reitsma and J. Sagalyn, "Correlation measures," *Information Storage and Retrieval*, vol. 13, p. Ch. IV, 1967. 25

[30] W. S. Torgerson, *Theory and methods of scaling*. R.E. Krieger Pub. Co., 1985. 29

[31] G. Strang, *Linear Algebra and Its Applications, 4th Edition*. Brooks/Cole INDIA, 4th indian ed., 2005. 29

[32] J. O. Lay, M. L. Gross, J. J. Zwinselman, and N. M. M. Nibbering, "A field ionization and collisionally activated dissociation/charge stripping study of some [C9H10]+ ions," *Organic Mass Spectrometry*, vol. 18, no. 1, pp. 16–21, 1983. 29

[33] K. X. Wan, I. Vidavsky, and M. L. Gross, "Comparing similar spectra: From similarity index to spectral contrast angle," *Journal of The American Society for Mass Spectrometry*, vol. 13, pp. 85–88, 2002. 29, 30

[34] T. T. Tanimoto, "IBM internal report," tech. rep., Nov. 1957. 30

[35] D. J. Rogers and T. T. Tanimoto, "A computer program for classifying plants," *Science*, vol. 132, no. 3434, pp. 1115–1118, 1960. 30

[36] P. Jaccard, "Distribution de la flore alpine dans le bassin des dranses et dans quelques rgions voisines," *Bulletin de la Socit Vaudoise des Sciences Naturelles*, vol. 37, pp. 241–272, 1901. 30

[37] K. Scharnehorst, "Angles in complex vector spaces," *Acta Applicandae Mathematicae*, vol. 69, pp. 95–103, 2001. 31

[38] R. J. Mathews and J. D. Morrison, "Comparative study of methods of Computer-Matching mass spectra," *Australian Journal of Chemistry*, vol. 27, pp. 2167–73, 1974. 34

[39] G. T. Rasmussen and T. L. Isenhour, "The evaluation of mass spectral search algorithms," *Journal of Chemical Information and Modeling*, vol. 19, pp. 179–186, Aug. 1979. 35

[40] J. julien Aucouturier and F. Pachet, "Music similarity measures: What's the use ?," 2002. 35

[41] B. Blad and B. Baldetorp, "Impedance spectra of tumour tissue in comparison with normal tissue; a possible clinical application for electrical impedance tomography," *Physiological Measurement*, pp. A105–15, Nov. 1996. 35

[42] W. L. Martnez and ngel R. Martnez, *Exploratory data analysis with MATLAB*. CRC Press, 2005. 36

[43] R. R. Sokal and P. H. A. Sneath, *Principles of numerical taxonomy*. W. H. Freeman, 1963. 36

[44] F. van der Meer, "The effectiveness of spectral similarity measures for the analysis of hyperspectral imagery," *International Journal of Applied Earth Observation and Geoinformation*, vol. 8, pp. 3–17, Jan. 2006. 36

[45] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley, 2001. 56

[46] G. Shmueli, N. R. Patel, and P. C. Bruce, *Data Mining for Business Intelligence: Concepts, Techniques, and Applications in Microsoft Office Excel with XLMiner*. John Wiley and Sons, Oct. 2010. 69

Appendix

Appendix A

Source Code

All source code is written for execution in MATLAB® version 7.11.0.584 (R2010b) 64-bit.

A.1 Dimension Reduction

A.1.1 Peak Binning

```
% This M-File contains the commands for dimension reduction using binning
% techniques.
% Requires the following files: data_groups.mat, dispDebug.m, getPeaks.m

% SETTINGS
NUMBER_OF_BINS = 1601;
% Calculate total not matching the group centroid
CALC_TOT_NOT_MATCH_CENTROID = 1;

if exist('data_groups', 'var')==0
    load('data_groups');
end
```

```
13
14 % Create our cell array that will hold the binned values
15 data_groups_bin = cell(size(data_groups));
16
17 bins = 0:size(data_groups{1},1)/NUMBER_OF_BINS:size(data_groups{1},1);
18 if exist('num_dif_from_centroid','var')==0
19     num_dif_from_centroid = zeros(1601,13);
20 end
21 binned_centroid = zeros(NUMBER_OF_BINS,13);
22
23 % Reduce the dimension of each data group
24 for k=1:length(data_groups)
25     dispDebug(['Currently Binning Group ' num2str(k) ' Into ' ...
26         num2str(NUMBER_OF_BINS) ' Bins'])
27     % Create our centroid reduced
28     centroid = mean(data_groups{k},2);
29     cur_peaks = getPeaks(centroid);
30     for j=1:size(binned_centroid,1)
31         bin_min = bins(j);
32         if j<length(bins)-1
33             bin_max = bins(j+1);
34         else
35             bin_max = inf;
36         end
37         peaks_in_bin = find(cur_peaks≥bin_min & cur_peaks<bin_max);
38         if isempty(peaks_in_bin)
39             binned_centroid(j,k) = 0;
40         else
41             binned_centroid(j,k) = mean(centroid(peaks_in_bin));
42         end
43     end
44     % Preallocate the matrix that will hold our bin values
```

```
45      data_bin = zeros(NUMBER_OF_BINS,size(data_groups{k},2));
46      % For every sample
47      for i=1:size(data_bin,2)
48          dif_from_centroid = 0;
49          cur_peaks = getPeaks(data_groups{k}(:,i));
50          for j=1:size(data_bin,1)
51              bin_min = bins(j);
52              if j<length(bins)-1
53                  bin_max = bins(j+1);
54              else
55                  bin_max = inf;
56              end
57              peaks_in_bin = find(cur_peaks≥bin_min & cur_peaks<bin_max);
58              if isempty(peaks_in_bin)
59                  data_bin(j,i) = 0;
60              else
61                  data_bin(j,i) = mean(data_groups{k}(peaks_in_bin,i));
62              end
63              if (data_bin(j,i)≠0 && binned_centroid(j,k)==0) || ...
64                      (data_bin(j,i)==0 && binned_centroid(j,k)≠0)
65                  dif_from_centroid=1;
66              end
67          end
68          if dif_from_centroid==1
69              num_dif_from_centroid(NUMBER_OF_BINS,k) = ...
70                  num_dif_from_centroid(NUMBER_OF_BINS,k)+1;
71          end
72      end
73      data_groups_bin(k) = {data_bin};
74  end
75  % Clean up the variables that won't be used again
76  clear i j k data_bin cur_peaks bin_min bin_max peaks_in_bin;
```

```
function output = getPeaks( spectrum )
output = [];
for i=2:length(spectrum)
    %if( phase(spectrum(i-1))*phase(spectrum(i))<0 )
    if( phase(spectrum(i-1))>0 && phase(spectrum(i))<0 )
        if( isempty(output) || i-1>output(length(output))+20 )
            output = [output,i-1];
        end
    end
end
```

A.1.2 SVD

```
function [scree_fig, loadings_fig, scores2_fig, scores3_fig] = ...
    createSVDPlots( data_groups )
% Creates scree, loadings, and scores plots of the data in cells of
% data_groups combined and mean centered.

% REQUIRES: Variables U, S, V from SVD

% Mean center the data
X = cat(1,data_groups{:});
X_c = X-ones(size(X,1),1)*mean(X);
[U, S, V] = svd(X_c);

% OPTIONS
SHOW_SCREE_PLOT = 1;
SHOW_V_PLOT = 1;
SHOW_U_PLOT = 1;
```

```
17 SHOW_3D_U_PLOT = 1;
18 DEBUG = 1;
19
20 dispDebug('Creating SVD Plots', DEBUG);
21
22 % Show the scree plot
23 if SHOW_SCREE_PLOT==1
24     scree_fig = figure();
25     set(scree_fig, 'color', 'white');
26     plot(cumsum(diag(S).^2)./sum(diag(S).^2), 'b')
27     grid minor
28     xlabel('PC #');
29     ylabel('Cumulative Sum of Normalized Singular Values Squared');
30     title('Scree Plot from the SVD of Training Samples')
31     clear scree_fig;
32 end
33
34 colors = [0 114 178; 213 94 0; 0 158 115; 230 159 0; 86 180 233; ...
35     204 121 167; 0 0 0; 240 228 66; 125 97 186; 151 84 79; 165 136 105; ...
36     143 148 152; 137 104 89];
37
38 % Show the V plot with the first n PCs
39 if SHOW_V_PLOT==1
40     loadings_fig = figure();
41     set(loadings_fig, 'color', 'white');
42     num_v_to_plot = 4;
43     legend_labels = char(zeros(num_v_to_plot,3));
44     for i=1:num_v_to_plot
45         plot(abs(V(:,i)),'Color',colors(i,:)./255)
46         hold on
47         legend_labels(i,:) = ['PC' num2str(i)];
48     end
```

```
49      grid minor
50      legend(legend_labels);
51      xlabel('Original Frequency');
52      ylabel('Value in V Frame');
53      title('Loadings Vectors from the SVD of Training Samples')
54      clear v_fig num_v_to_plot legend_labels i;
55  end
56
57  % Get the group counts for proper labeling of U legends
58  if SHOW_U_PLOT==1 || SHOW_3D_U_PLOT==1
59      group_sizes = zeros(1,length(data_groups));
60      for i=1:length(data_groups)
61          group_sizes(i) = size(data_groups{i},1);
62      end
63      clear i;
64  end
65
66  % Show the U plot with the first 2 PCs
67  if SHOW_U_PLOT==1
68      scores2_fig = figure();
69      set(scores2_fig, 'color', 'white');
70      current_u_pointer = 0;
71      hold on
72      US = U*S;
73      legend_labels = char(zeros(length(data_groups),3));
74      for i=1:length(data_groups)
75          plot(abs(US(current_u_pointer+1:current_u_pointer+group_sizes(i)...
76              ,1)),abs(US(current_u_pointer+1:current_u_pointer+ ...
77              group_sizes(i),2)),'Color',colors(i,:)./255,'Marker','.', ...
78              'LineStyle','none')
79          hold on
80          current_u_pointer = current_u_pointer+group_sizes(i);
```

```
81          legend_labels(i,:) = ['G' num2str(i,'%02d')];
82      end
83      grid minor
84      legend(legend_labels);
85      xlabel('PC 1');
86      ylabel('PC 2');
87      title('Scores Plot from the SVD of Training Samples')
88      clear u_fig current_u_pointer US legend_labels i;
89  end
90
91  % Show the U plot with the first 3 PCs
92  if SHOW_3D_U_PLOT==1
93      scores3_fig = figure();
94      set(scores3_fig, 'color', 'white');
95      current_u_pointer = 0;
96      hold on
97      US = U*S;
98      legend_labels = char(zeros(length(data_groups),3));
99      for i=1:length(data_groups)
100         plot3(abs(US(current_u_pointer+1:current_u_pointer+group_sizes( ...
101             i),1)),abs(US(current_u_pointer+1:current_u_pointer+ ...
102             group_sizes(i),2)),abs(US(current_u_pointer+1: ...
103             current_u_pointer+group_sizes(i),3)),'Color',colors(i,: ...
104             )./255,'Marker','.','LineStyle','none')
105         hold on
106         current_u_pointer = current_u_pointer+group_sizes(i);
107         legend_labels(i,:) = ['G' num2str(i,'%02d')];
108     end
109     grid minor
110     legend(legend_labels);
111     xlabel('PC 1');
112     ylabel('PC 2');
```

```
113     zlabel('PC 3');
114     title('Scores Plot from the SVD of Training Samples')
115     clear u3_fig current_u_pointer US legend_labels i;
116 end
117
118 clear group_sizes colors;
119 % Clear the options
120 clear SHOW_SCREE_PLOT SHOW_V_PLOT SHOW_U_PLOT SHOW_3D_U_PLOT DEBUG;
```

```
1  function [reduced_data_groups, V_from_SVD] = ...
2      ReduceDataWithSVD(data_groups,num_of_dim,V_from_SVD)
3
4  if exist('V_from_SVD','var')==0
5      % Compute the SVD of all of the data.
6      dispDebug('Calculating SVD of Data');
7      % Mean center the data
8      X = cat(1,data_groups{:});
9      X_c = X-ones(size(X,1),1)*mean(X);
10     [~, ~, V_from_SVD] = svd(X_c);
11 end
12
13 reduced_data_groups = cell(1,length(data_groups));
14 dispDebug(['Reducing Data to First ' num2str(num_of_dim) ...
15     ' Principal Components']);
16 for i=1:length(reduced_data_groups)
17     reduced_data_groups{i} = data_groups{i}*V_from_SVD;
18     reduced_data_groups{i} = reduced_data_groups{i}(:,1:num_of_dim);
19 end
```

A.2 Classification Routine

```
function classif_matrix = Classification(data_groups,random_perm)

% This M-File contains the steps required for classification of the
% impedance spectra following the steps outlined in the thesis proposal for
% Carl Sapp's electrical impedance classification

% SETTINGS - These parameters can be changed to modify features of the
% classification.
% TRAINING_PERCENT - The percentage of each group that will be used as the
% training set. Valid range is 0-1. 1-TRAINING_PERCENT is the percentage
% that will be used for the test set.
TRAINING_PERCENT = .6;
% SIM_MEASURES - Function handles to similarity functions. The functions
% must operate the same way as the input to the function pdist.
SIM_MEASURES = {@complexMahalanobis, @complexInnerProduct, ...
    @complexDistNormalized, @complexManhattan, @complexAverage, ...
    @complexSquaredChordDist, @complexCanberra, ...
    @complexCoefficientOfDivergence, @complexModBoolCorrelation, ...
    @complexAvgWeightOfSharedTerms, @complexOverlap, @complexRSquared, ...
    @complexSimilarityIndex, @complexTanimoto};
% SIM_OR_DIS - Specifies if the above similarity functions are similarity
% or disimilarity function. The size of SIM_OR_DIS must be the same as
% SIM_MEASURES. A value of 1 indicates a higher value is more similar. A
% value of 0 indicates that a higher value is less similar.
SIM_OR_DIS = [0,1,1,0,0,0,0,0,1,1,1,1,0,1];
% NUM_PERMUTATIONS - The number of times to repeat the classification
% procedure.
NUM_PERMUTATIONS = 1;
% CLASSIFY_IF_BELOW_THRESH - Indicates if a sample should be classified
```

```
% into a group if its below the threshold for the most similar group. A
% value of one assigns the sample to the most similar category. A value of
% zero classifies the sample in group n+1 where n is the number of
% categories.
CLASSIFY_IF_BELOW_THRESH = 0;
% CLASSIFY_IF_SIM_TO_MULT - Indicates if a sample should be classified into
% a group if its above the threshold for multiple groups. A value of one
% assigns the sample to the most similar category. A value of zero
% classifies the sample in group n+2 where n is the number of categories.
CLASSIFY_IF_SIM_TO_MULT = 1;
% DEBUG - Specifies whether to display debug ouput. 1 to display, 0 to not.
DEBUG = 1;
SAVE_FIGS = 0;
USE_EXISTING_TRAIN_TEST_PERM = 1;

% Load the data for the classification if its not loaded
if exist('data_groups', 'var')==0
    load('data_groups_cell.mat')
end

% Initialize storage for the classification confusion matrices
classif_matrix = zeros(length(data_groups)+3,length(data_groups), ...
    NUM_PERMUTATIONS,length(SIM_MEASURES));
% Save our error rates for reporting - 101 per metric per group per run per
% metric
error_rates = zeros(101,length(data_groups),NUM_PERMUTATIONS, ...
    length(SIM_MEASURES));
timings = zeros(length(SIM_MEASURES),NUM_PERMUTATIONS);
for i=1:NUM_PERMUTATIONS
    dispDebug(['Executing Run ' num2str(i)], DEBUG);
    % Separate each group into their training and test groups
    dispDebug('Creating Training and Test Groups', DEBUG);
```

```
62      data_groups_training = cell(1,length(data_groups));
63      data_groups_test = cell(1,length(data_groups));
64      if USE_EXISTING_TRAIN_TEST_PERM==0 || exist('random_perm','var')==0 ...
65              || length(random_perm)<length(data_groups)
66          random_perm = cell(1,length(data_groups));
67      end
68      for g=1:size(data_groups,2)
69          % Shuffle the samples in the group
70          if USE_EXISTING_TRAIN_TEST_PERM==0 || ...
71                  exist('random_perm','var')==0 || isempty(random_perm{g})
72              dispDebug(['Creating Random Permutation for Training/Test ' ...
73                  'for Group ' num2str(g)], DEBUG);
74              random_perm{g} = randperm(size(data_groups{g},2));
75          end
76          % Assign 60% of the samples to training, 40% to test
77          num_training = round(size(data_groups{g},1)*TRAINING_PERCENT);
78          data_groups_training(g) = {data_groups{g}( ...
79              random_perm{g}(1:num_training),:)};
80          data_groups_test(g) = {data_groups{g}(random_perm{g}( ...
81              num_training+1:size(data_groups{g},1)),:)};
82      end
83      clear num_training g
84
85      % Determine the centroids of each training group
86      dispDebug('Calculating Group Centroids', DEBUG);
87      training_centroids = zeros(length(data_groups_training), ...
88          size(data_groups_training{1},2));
89      for g=1:length(data_groups_training)
90          training_centroids(g,:) = mean(data_groups_training{g});
91      end
92      clear g
93
```

```
94      if SAVE_FIGS==1
95          % Create the folder to save everything into
96          c = clock;
97          fig_folder = [ 'test' num2str(c(1)) num2str(c(2),'%02d') ...
98              num2str(c(3),'%02d') num2str(c(4),'%02d') ...
99              num2str(c(5),'%02d') num2str(c(6),'%02.0f') ];
100         mkdir(fig_folder);
101         fig_folder = [fig_folder '/'];
102         clear c;
103     end
104
105     % If we will be checking the complex Mahalanobis distance, lets precompute
106     % the inverse of the covariance of the groups to speed things up
107     for s=1:length(SIM_MEASURES)
108         if isequal(SIM_MEASURES{s},@complexMahalanobis) && ...
109                 ( USE_EXISTING_TRAIN_TEST_PERM==0 || ...
110                 exist('data_groups_training_cov_inv','var')==0 || ...
111                 size(data_groups_training_cov_inv{1},1) ≠ ...
112                 size(data_groups_training{1},2) )
113             dispDebug('Preparing Inverse Covariance Matrices for the ' ...
114                 'Mahalanobis Distance', DEBUG);
115             % Precompute the inverse of the covariance of each group
116             data_groups_training_cov_inv = cell(size(data_groups));
117             for c=1:length(data_groups_training_cov_inv)
118                 dispDebug(['Preparing Inverse Covariance Matrix for ' ...
119                     'Group ' num2str(c)], DEBUG);
120                 data_groups_training_cov_inv{c} = ...
121                     pinv_cond(cov(data_groups_training{c}),1000);
122             end
123         end
124     end
125     clear s c
```

```
126
127     % BEGIN Repeat for each similarity measure
128     if SAVE_FIGS==1
129         if NUM_PERMUTATIONS>1
130             run_folder = ['Run' num2str(i)];
131             mkdir([fig_folder run_folder]);
132             run_folder = [run_folder '/'];
133         else
134             run_folder = '';
135         end
136     end
137     for s=1:length(SIM_MEASURES)
138         % Determine our thresholds for each group
139         f = functions(SIM_MEASURES{s});
140         dispDebug(['Testing Similarity Function ' f.function], DEBUG);
141         if SAVE_FIGS==1
142             sim_fun_folder = f.function;
143             mkdir([fig_folder run_folder sim_fun_folder]);
144             sim_fun_folder = [sim_fun_folder '/'];
145         end
146         thresholds = inf(1,length(data_groups_training));
147         for g=1:length(data_groups_training)
148             % Mahalanobis Distance is different because we compare a sample
149             % to a group and have to calculate it differently
150             if isequal(SIM_MEASURES{s},@complexMahalanobis)
151                 % in_group_sim consists of distances to the group
152                 in_group_sim = zeros(size(data_groups_training{g},1),1);
153                 for t=1:size(data_groups_training{g},1)
154                     in_group_sim(t) = SIM_MEASURES{s}( ...
155                         data_groups_training{g}(t,:), ...
156                         data_groups_training{g}, ...
157                         data_groups_training_cov_inv{g});
```

```
158                 end
159                 non_group_train = cat(1,data_groups_training{[1:g-1 ...
160                     g+1:length(data_groups_training)]});
161                 out_group_sim = zeros(size(non_group_train,1),1);
162                 for t=1:size(non_group_train,1)
163                     out_group_sim(t) = SIM_MEASURES{s}( ...
164                         non_group_train(t,:), data_groups_training{g}, ...
165                         data_groups_training_cov_inv{g});
166                 end
167                 clear t non_group_train;
168             else
169                 in_group_sim = SIM_MEASURES{s}(training_centroids(g,:), ...
170                     data_groups_training{g});
171                 out_group_sim = SIM_MEASURES{s}(training_centroids(g,:),...
172                     cat(1,data_groups_training{[1:g-1 g+1:length( ...
173                     data_groups_training)]}));
174             end
175             edges = min([in_group_sim; out_group_sim]):(max( ...
176                 [in_group_sim; out_group_sim])-min([in_group_sim; ...
177                 out_group_sim]))/100:max([in_group_sim; out_group_sim]);
178             if SAVE_FIGS==1
179                 % Graph the histograms
180                 members = histc(in_group_sim,edges);
181                 nonmembers = histc(out_group_sim,edges);
182                 % Save the histogram as a PNG image
183                 fig = figure();
184                 subplot(2,1,1)
185                 bar(edges,[members./sum(members).*100, ...
186                     nonmembers./sum(nonmembers).*100]);
187                 hold on
188                 legend(['G' num2str(g,'%02d') ' Members'], ...
189                     ['G' num2str(g,'%02d') ' Nongroup Members']);
```

```
190                 xlabel('Similarity Value');
191                 ylabel('% of Nongroup or Group Samples');
192                 f = functions(SIM_MEASURES{s});
193                 title(['Similarity of All Samples to Group ' ...
194                     num2str(g,'%02d') ' Centroid using ' f.function]);
195                 clear f;
196                 grid minor
197             end
198
199             % Calculate our error rate at each edge
200             min_error_rate = inf;
201             for e=1:length(edges)
202                 % Error is defined as number of non-group members greater than
203                 % or equal to the threshold + number of member less than the
204                 % threshold
205                 if SIM_OR_DIS(s)==1
206                     error_rates(e,g,i,s) = length(find( ...
207                         in_group_sim<edges(e)))/length(in_group_sim)+ ...
208                         length(find(out_group_sim≥edges(e)))/length( ...
209                         out_group_sim);
210                 else
211                     error_rates(e,g,i,s) = length(find( ...
212                         in_group_sim≥edges(e)))/length(in_group_sim)+ ...
213                         length(find(out_group_sim<edges(e)))/length( ...
214                         out_group_sim);
215                 end
216                 if error_rates(e,g,i,s)<min_error_rate
217                     min_error_rate = error_rates(e,g,i,s);
218                     thresholds(g) = edges(e);
219                 end
220             end
221             if SAVE_FIGS==1
```

```
222                 % Plot the threshold value
223                 plot([thresholds(g); thresholds(g)], [min([members; ...
224                     nonmembers]); max([members./sum(members); ...
225                     nonmembers./sum(nonmembers)])*100], 'Color', ...
226                     [.93 .53 .18], 'LineWidth', 2)
227                 subplot(2,1,2)
228                 plot([thresholds(g); thresholds(g)], [min([members; ...
229                     nonmembers]); max(error_rates(:,g,i,s)./2.*100)], ...
230                     'Color', [.93 .53 .18], 'LineWidth', 2)
231                 hold on
232                 % Plot the error rates
233                 plot(edges,error_rates(:,g,i,s)./2.*100)
234                 grid minor
235                 xlabel('Similarity Value')
236                 ylabel('Error Rate %')
237                 hold off
238                 saveas(fig, [fig_folder run_folder sim_fun_folder 'G' ...
239                     num2str(g,'%02d') 'ThreshBarGraph.fig']);
240                 close(fig);
241             end
242         end
243
244         % Classify each of the test samples to a training group or no group
245         % Do each group individually
246         f = functions(SIM_MEASURES{s});
247         dispDebug(['Performing Classification with ' f.function], DEBUG);
248         tic;
249         for g=1:length(data_groups_test)
250             % Classify each sample in the group
251             for sample=1:size(data_groups_test{g},1)
252                 % Calculate the similarity between the sample and each
253                 %  centroid
```

```
254                 if isequal(SIM_MEASURES{s},@complexMahalanobis)
255                     % If we are using the mahalanobis distance, we have to
256                     % compare to each group instead of centroids.
257                     dist_to_centroids = zeros(length(data_groups_test),1);
258                     for tg=1:length(data_groups_test)
259                         dist_to_centroids(tg) = SIM_MEASURES{s}( ...
260                             data_groups_test{g}(sample,:), ...
261                             data_groups_training{tg}, ...
262                             data_groups_training_cov_inv{tg});
263                     end
264                 else
265                     dist_to_centroids = SIM_MEASURES{s}( ...
266                         data_groups_test{g}(sample,:),training_centroids);
267                 end
268                 if SIM_OR_DIS(s)==1
269                     [classif_distance, classif_category] = ...
270                         max(dist_to_centroids);
271                 else
272                     [classif_distance, classif_category] = ...
273                         min(dist_to_centroids);
274                 end
275                 % Calculate number of categories we are higher than
276                 % threshold
277                 if SIM_OR_DIS(s)==1
278                     [¬, higher_than_thresh] = max([thresholds; ...
279                         dist_to_centroids']);
280                 else
281                     [¬, higher_than_thresh] = min([thresholds; ...
282                         dist_to_centroids']);
283                 end
284                 % If the distance is less less similar than the threshold
285                 % for that category, we categorize it to group 14 (unknown)
```

```
286                 if ((SIM_OR_DIS(s)==1 && classif_distance<thresholds( ...
287                         classif_category)) || (SIM_OR_DIS(s)==0 && ...
288                         classif_distance>thresholds(classif_category))) ...
289                         && CLASSIFY_IF_BELOW_THRESH==0
290                     classif_matrix(length(data_groups)+1,g,i,s) = ...
291                         classif_matrix(length(data_groups)+1,g,i,s)+1;
292                 else
293                     if CLASSIFY_IF_SIM_TO_MULT==0 && length(find( ...
294                             higher_than_thresh==2))>1
295                         if isempty(find( find(higher_than_thresh==2)==g ...
296                                 ))==0
297                             classif_matrix(length(data_groups)+3,g,1,s) ...
298                                 = classif_matrix(length(data_groups ...
299                                 )+3,g,1,s)+1;
300                         else
301                             classif_matrix(length(data_groups)+2,g,1,s) ...
302                                 = classif_matrix(length(data_groups ...
303                                 )+2,g,1,s)+1;
304                         end
305                     else
306                         classif_matrix(classif_category,g,i,s) = ...
307                             classif_matrix(classif_category,g,i,s)+1;
308                     end
309                 end
310             end
311         end
312         timings(s,i) = toc;
313         num_correct = sum(diag(classif_matrix(:,:,i,s)));
314         num_incorrect = sum(sum(classif_matrix(:,:,i,s)))-num_correct;
315         f = functions(SIM_MEASURES{s});
316         dispDebug([f.function ' - Correct: ' num2str(num_correct) ' (' ...
317             num2str(num_correct/(num_correct+num_incorrect)*100) ...
```

```
318             '%) Incorrect: ' num2str(num_incorrect) ' (' ...
319             num2str(num_incorrect/(num_correct+num_incorrect)*100) '%)']);
320     end % END Repeat for each similarity measure
321     clear s g bin_size e error_rate min_error_rate i num_correct ...
322         num_incorrect f;
323     if NUM_PERMUTATIONS>1
324         clear random_perm;
325     end
326 end
327 % Clear the settings
328 clear TRAINING_PERCENT SIM_MEASURES SIM_OR_DIS NUM_PERMUTATIONS;
```

A.3 Similarity/Dissimilarity Metrics

```
1  % This function is set up for use in the pdist function as a metric. takes
2  % as arguments a 1-by-p vector a, corresponding to a single row of X, and
3  % an m-by-p matrix b, corresponding to multiple rows of X. distfun must
4  % accept a matrix b with an arbitrary number of rows. distfun must return
5  % an m-by-1 vector of distances result, whose kth element is the distance
6  % between a and b(k,:).
7  function result = complexAverage(a,b)
8
9  % Make sure our inputs are rows of values
10 if size(a,1)>1
11     if size(a,2)>1
12         error('The first argument must be a vector');
13     end
14     a = a.';
15 end
```

```
16 % Make sure we have samples as rows and features as columns
17 if size(b,2) ≠ size(a,2)
18     % b should be #ofSample X #ofFeatures. Width of b should match
19     % width of a
20     if size(b,1) ≠ size(a,2)
21         error(['One of the sides of the b matrix must be the same ' ...
22             'length as a']);
23     end
24     b = b.';
25 end
26
27 result = abs(sum(ones(size(b,1),1)*a-b,2)./size(b,2));
```

```
1  % This function is set up for use in the pdist function as a metric. takes
2  % as arguments a 1-by-p vector a, corresponding to a single row of X, and
3  % an m-by-p matrix b, corresponding to multiple rows of X. distfun must
4  % accept a matrix b with an arbitrary number of rows. distfun must return
5  % an m-by-1 vector of distances result, whose kth element is the distance
6  % between a and b(k,:).
7  function result = complexAvgWeightOfSharedTerms(a,b)
8
9  % Make sure our inputs are rows of values
10 if size(a,1)>1
11     if size(a,2)>1
12         error('The first argument must be a vector');
13     end
14     a = a.';
15 end
16 % Make sure we have samples as rows and features as columns
17 if size(b,2) ≠ size(a,2)
18     % b should be #ofSample X #ofFeatures. Width of b should match width
```

```
19      % of a
20      if size(b,1) ≠ size(a,2)
21          error(['One of the sides of the b matrix must be the same ' ...
22              'length as a']);
23      end
24      b = b.';
25  end
26
27
28
29  % The inner product of two complex valued vectors is the inner product with
30  % the second vector being the complex conjugate.  Performing a transpose of
31  % b causes the conjugate transpose so we are fine
32  % We have to use the magnitude of the complex values for this metric
33  % because negative real or imaginary parts messes things up and makes the
34  % calculation incorrect.
35  a = abs(a);
36  b = abs(b);
37  result = abs((sum(((ones(size(b,1),1)*a+b)),2))./(2*size(b,2)));
```

```
1   % This function is set up for use in the pdist function as a metric. takes
2   % as arguments a 1-by-p vector a, corresponding to a single row of X, and
3   % an m-by-p matrix b, corresponding to multiple rows of X. distfun must
4   % accept a matrix b with an arbitrary number of rows. distfun must return
5   % an m-by-1 vector of distances result, whose kth element is the distance
6   % between a and b(k,:).
7   function result = complexCanberra(a,b)
8
9   % Make sure our inputs are rows of values
10  if size(a,1)>1
11      if size(a,2)>1
```

```
        error('The first argument must be a vector');
    end
    a = a.';
end
% Make sure we have samples as rows and features as columns
if size(b,2) ≠ size(a,2)
    % b should be #ofSample X #ofFeatures. Width of b should match width
    % of a
    if size(b,1) ≠ size(a,2)
        error(['One of the sides of the b matrix must be the same ' ...
            'length as a']);
    end
    b = b.';
end

result = abs( sum(abs(ones(size(b,1),1)*a-b)./(ones(size(b,1),1)*a+b),2) );
```

```
% This function is set up for use in the pdist function as a metric. takes
% as arguments a 1-by-p vector a, corresponding to a single row of X, and
% an m-by-p matrix b, corresponding to multiple rows of X. distfun must
% accept a matrix b with an arbitrary number of rows. distfun must return
% an m-by-1 vector of distances result, whose kth element is the distance
% between a and b(k,:).
function result = complexCoefficientOfDivergence(a,b)

% Make sure our inputs are rows of values
if size(a,1)>1
    if size(a,2)>1
        error('The first argument must be a vector');
    end
    a = a.';
```

```
end
% Make sure we have samples as rows and features as columns
if size(b,2) ≠ size(a,2)
    % b should be #ofSample X #ofFeatures. Width of b should match width
    % of a
    if size(b,1) ≠ size(a,2)
        error(['One of the sides of the b matrix must be the same ' ...
            'length as a']);
    end
    b = b.';
end



% The inner product of two complex valued vectors is the inner product with
% the second vector being the complex conjugate.  Performing a transpose of
% b causes the conjugate transpose so we are fine

result = abs(sum(((ones(size(b,1),1)*a-b)./abs(ones(size(b,1),1)*a+b) ...
    ).^2,2)./size(b,2));
```

```
% This function is set up for use in the pdist function as a metric. takes
% as arguments a 1-by-p vector a, corresponding to a single row of X, and
% an m-by-p matrix b, corresponding to multiple rows of X. distfun must
% accept a matrix b with an arbitrary number of rows. distfun must return
% an m-by-1 vector of distances result, whose kth element is the distance
% between a and b(k,:).
function result = complexDistNormalized(a,b)

% Make sure our inputs are rows of values
if size(a,1)>1
```

```
11      if size(a,2)>1
12          error('The first argument must be a vector');
13      end
14      a = a';
15  end
16  % Make sure we have samples as rows and features as columns
17  if size(b,2) ≠ size(a,2)
18      % b should be #ofSample X #ofFeatures. Width of b should match width
19      % of a
20      if size(b,1) ≠ size(a,2)
21          error(['One of the sides of the b matrix must be the same ' ...
22              'length as a']);
23      end
24      b = b.';
25  end
26
27  % Preallocate the result matrix for speed
28  result = zeros(size(b,1),1);
29  lengthA = sum(real(a)*real(a)'+imag(a)*imag(a)')^(1/2);
30  for j=1:size(b,1)
31      % Subtract A from B
32      b(j,:) = b(j,:)-a;
33
34      lengthB = sum(real(b(j,:))*real(b(j,:))'+imag(b(j,:))*imag( ...
35          b(j,:))')^(1/2);
36
37      result(j,1) = 1/(1+lengthB^2/lengthA^2);
38  end
39
40  end
```

```
% This function is set up for use in the pdist function as a metric. takes
% as arguments a 1-by-p vector a, corresponding to a single row of X, and
% an m-by-p matrix b, corresponding to multiple rows of X. distfun must
% accept a matrix b with an arbitrary number of rows. distfun must return
% an m-by-1 vector of distances result, whose kth element is the distance
% between a and b(k,:).
function result = complexInnerProduct(a,b)

% Make sure our inputs are rows of values
if size(a,1)>1
    if size(a,2)>1
        error('The first argument must be a vector');
    end
    a = a';
end
% Make sure we have samples as rows and features as columns
if size(b,2) ≠ size(a,2)
    % b should be #ofSample X #ofFeatures. Width of b should match width
    % of a
    if size(b,1) ≠ size(a,2)
        error(['One of the sides of the b matrix must be the same ' ...
            'length as a']);
    end
    b = b.';
end

result = abs(a*b')';
```

```
% Calculates the Mahalanobis distance between the vector a and the group b.
% The input a can be a row or column vector.  The input matrix b is a group
```

```
% of samples with the same dimension as a.  The matrix b needs to
% have one of the sides be the same size as the column vector.
function result = complexMahalanobis(a,b,b_cov_inv)

% Make sure our inputs are rows of values
% a needs to be 1 X #ofFeatures
if size(a,1)>1
    if size(a,2)>1
        error('The first argument must be a vector');
    end
    disp('Transposing a');
    a = a.';
end

% b needs to be #ofSamples X #ofFeatures
if size(b,2) ~= size(a,2)
    % b should be #ofSample X #ofFeatures. Width of b should match width
    % of a
    if size(b,1) ~= size(a,2)
        error(['One of the sides of the b matrix must be the same ' ...
            'length as a']);
    end
    disp('Transposing b');
    b = b.';
end

% Check if the inverse of the covariance of b was provided to speed things
% up.
if exist('b_cov_inv','var')
    result = abs(sqrt((a-mean(b))*b_cov_inv*(a-mean(b)).'));
else
    % The size of sigma is #ofFeatures X #ofFeatures
```

```
35      sigma = cov(b);
36      warning off all
37      result = abs((a-mean(b))*pinv_cond(sigma,1000)*(a-mean(b)).');
38      warning on all
39  end
```

```
1  % This function is set up for use in the pdist function as a metric. takes
2  % as arguments a 1-by-p vector a, corresponding to a single row of X, and
3  % an m-by-p matrix b, corresponding to multiple rows of X. distfun must
4  % accept a matrix b with an arbitrary number of rows. distfun must return
5  % an m-by-1 vector of distances result, whose kth element is the distance
6  % between a and b(k,:).
7  function result = complexManhattan(a,b)
8
9  % Make sure our inputs are rows of values
10 if size(a,1)>1
11     if size(a,2)>1
12         error('The first argument must be a vector');
13     end
14     a = a.';
15 end
16 % Make sure we have samples as rows and features as columns
17 if size(b,2) ≠ size(a,2)
18     % b should be #ofSample X #ofFeatures. Width of b should match width
19     % of a
20     if size(b,1) ≠ size(a,2)
21         error(['One of the sides of the b matrix must be the same ' ...
22             'length as a']);
23     end
24     b = b.';
25 end
```

```
26
27 result = sum(abs(ones(size(b,1),1)*a-b),2);
```

```
1  % This function is set up for use in the pdist function as a metric. takes
2  % as arguments a 1-by-p vector a, corresponding to a single row of X, and
3  % an m-by-p matrix b, corresponding to multiple rows of X. distfun must
4  % accept a matrix b with an arbitrary number of rows. distfun must return
5  % an m-by-1 vector of distances result, whose kth element is the distance
6  % between a and b(k,:).
7  function result = complexModBoolCorrelation(a,b)
8
9  % Make sure our inputs are rows of values
10 if size(a,1)>1
11     if size(a,2)>1
12         error('The first argument must be a vector');
13     end
14     a = a.';
15 end
16 % Make sure we have samples as rows and features as columns
17 if size(b,2) ≠ size(a,2)
18     % b should be #ofSample X #ofFeatures. Width of b should match width
19     % of a
20     if size(b,1) ≠ size(a,2)
21         error(['One of the sides of the b matrix must be the same ' ...
22             'length as a']);
23     end
24     b = b.';
25 end
26
27 result = abs((sum(((ones(size(b,1),1)*a.*b)),2)+size(b,2))./size(b,2));
```

```
% This function is set up for use in the pdist function as a metric. takes
% as arguments a 1-by-p vector a, corresponding to a single row of X, and
% an m-by-p matrix b, corresponding to multiple rows of X. distfun must
% accept a matrix b with an arbitrary number of rows. distfun must return
% an m-by-1 vector of distances result, whose kth element is the distance
% between a and b(k,:).
function result = complexOverlap(a,b)

% Make sure our inputs are rows of values
if size(a,1)>1
    if size(a,2)>1
        error('The first argument must be a vector');
    end
    a = a.';
end
% Make sure we have samples as rows and features as columns
if size(b,2) ≠ size(a,2)
    % b should be #ofSample X #ofFeatures. Width of b should match width
    % of a
    if size(b,1) ≠ size(a,2)
        error(['One of the sides of the b matrix must be the same ' ...
            'length as a']);
    end
    b = b.';
end

a = abs(a);
b = abs(b);
numerator = sum(min(ones(size(b,1),1)*a,b),2);
denominator = min(sum(ones(size(b,1),1)*a,2),sum(b,2));
denominator(denominator==0)=0.0001;
```

```
result = abs( numerator./denominator );
```

```
% This function is set up for use in the pdist function as a metric. takes
% as arguments a 1-by-p vector a, corresponding to a single row of X, and
% an m-by-p matrix b, corresponding to multiple rows of X. distfun must
% accept a matrix b with an arbitrary number of rows. distfun must return
% an m-by-1 vector of distances result, whose kth element is the distance
% between a and b(k,:).
%
% [       a        ]   |        b        |   |   |
%                      |        b        |   |   |
%                      |        b        | = | r |
%                      |        b        |   |   |
%                      |        b        |   |   |
function result = complexRSquared(a,b)

% Make sure our inputs are rows of values
% If the first column of the the 1st input is larger than one value, its
% not a horizontal vector
if size(a,1)>1
    if size(a,2)>1
        % If the 1st column is larger than 1 and the 1st row is larger than
        % one, we have a rectangle that can't be a vector
        error('The first argument must be a vector');
    else
        % We have a column vector, we can easily transpose it to a row
        % vector
        a = a.';
    end
end
% Make sure we have samples as rows and features as columns
```

```
30 if size(b,2) ≠ size(a,2)
31     % b should be #ofSample X #ofFeatures. Width of b should match width
32     % of a
33     if size(b,1) ≠ size(a,2)
34         error(['One of the sides of the b matrix must be the same ' ...
35             'length as a']);
36     end
37     b = b.';
38 end
39
40 numerator = a*b';
41 denominator = (diag(b*b')'.^(1/2).*diag(a*a')'.^(1/2));
42 denominator(denominator==0)=0.0001;
43 result = abs(numerator./denominator)';
44
45 end
```

```
1  % This function is set up for use in the pdist function as a metric. takes
2  % as arguments a 1-by-p vector a, corresponding to a single row of X, and
3  % an m-by-p matrix b, corresponding to multiple rows of X. distfun must
4  % accept a matrix b with an arbitrary number of rows. distfun must return
5  % an m-by-1 vector of distances result, whose kth element is the distance
6  % between a and b(k,:).
7  function result = complexSimilarityIndex(a,b)
8
9  % Make sure our inputs are rows of values
10 if size(a,1)>1
11     if size(a,2)>1
12         error('The first argument must be a vector');
13     end
14     a = a.';
```

```
end
% Make sure we have samples as rows and features as columns
if size(b,2) ≠ size(a,2)
    % b should be #ofSample X #ofFeatures. Width of b should match width
    % of a
    if size(b,1) ≠ size(a,2)
        error(['One of the sides of the b matrix must be the same ' ...
            'length as a']);
    end
    b = b.';
end

% We will use the value from Wan et al. 2002 to take into account 0
% minimums
result = abs( sqrt(sum(((abs(ones(size(b,1),1)*a-b)./(ones(size(b,1),1 ...
    )*a+b)).*100).^2,2)./size(b,2)) );
```

```
% This function is set up for use in the pdist function as a metric. takes
% as arguments a 1-by-p vector a, corresponding to a single row of X, and
% an m-by-p matrix b, corresponding to multiple rows of X. distfun must
% accept a matrix b with an arbitrary number of rows. distfun must return
% an m-by-1 vector of distances result, whose kth element is the distance
% between a and b(k,:).
function result = complexSquaredChordDist(a,b)

% Make sure our inputs are rows of values
if size(a,1)>1
    if size(a,2)>1
        error('The first argument must be a vector');
    end
    a = a.';
```

```
15 end
16 % Make sure we have samples as rows and features as columns
17 if size(b,2) ≠ size(a,2)
18     % b should be #ofSample X #ofFeatures. Width of b should match width
19     % of a
20     if size(b,1) ≠ size(a,2)
21         error(['One of the sides of the b matrix must be the same ' ...
22             'length as a']);
23     end
24     b = b.';
25 end
26
27 result = abs(sum((ones(size(b,1),1)*sqrt(a)-sqrt(b)).^2,2));
```

```
1  % This function is set up for use in the pdist function as a metric. takes
2  % as arguments a 1-by-p vector a, corresponding to a single row of X, and
3  % an m-by-p matrix b, corresponding to multiple rows of X. distfun must
4  % accept a matrix b with an arbitrary number of rows. distfun must return
5  % an m-by-1 vector of distances result, whose kth element is the distance
6  % between a and b(k,:).
7  function result = complexTanimoto(a,b)
8
9  % Make sure our inputs are rows of values
10 if size(a,1)>1
11     if size(a,2)>1
12         error('The first argument must be a vector');
13     end
14     a = a.';
15 end
16 % Make sure we have samples as rows and features as columns
17 if size(b,2) ≠ size(a,2)
```

```
    % b should be #ofSample X #ofFeatures. Width of b should match width
    % of a
    if size(b,1) ≠ size(a,2)
        error(['One of the sides of the b matrix must be the same ' ...
            'length as a']);
    end
    b = b.';
end

result = abs( sum(ones(size(b,1),1)*a.*b,2)./(sum((ones(size(b,1),1)*a ...
    ).^2,2)+sum(b.^2,2)-sum(ones(size(b,1),1)*a.*b,2)) );
```

```
function x = pinv_cond(x, condition_number)
% This function takes the square matrix x and calculates the pseudo inverse
% using SVD but replaces all diagonals in S of USV' with zero if they are
% less than max(diag(S))/condition_number.

[U S V] = svd(x);
diag_S = diag(S);
if exist('condition_number','var')
    diag_S(diag_S<diag_S(1)/condition_number) = 0;
    disp(['Nonzero entries: ' num2str(length(find(diag_S)))])
end
diag_S(diag_S>0) = 1./diag_S(diag_S>0);
x = V*diag(diag_S)*U';
```

Vita

Carl Gordon Sapp was born in the 1980s in Florida. He is the son of William and Delarie. He grew up in Roanoke, Virginia and attended Northside High School. After graduation in 2002, Carl entered the College of Engineering at Virginia Polytechnic Institute and State University (Virginia Tech). Carl received a Bachelor of Science degree in Computer Engineering and a minor in Computer Science in May 2007.

After graduation, Carl worked at Virginia Bioinformatics Institute, GE Energy, and his own information technology consultation business. In 2009, Carl enrolled in graduate study at the University of Tennessee in Knoxville, Tennessee. Carl graduated with a Masters of Science degree in Electrical Engineering in May 2011. He has accepted employment as a Component Design Engineer with Intel Corporation in Columbia, South Carolina.