12-2014

Statistical Methods for Signal Processing with Application to Automatic Accent Recognition

Zichen Ma

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation


This Thesis is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.
Rochester Institute of Technology

Master’s Thesis

Statistical Methods for Signal Processing with Application to Automatic Accent Recognition

Author: Zichen Ma
Supervisor: Dr. Ernest Fokoué

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Applied Statistics

in the

The John D. Hromi Center for Quality and Applied Statistics
The Kate Gleason College of Engineering

December 2014
Committee Approval

The undersigned have examined thesis titled “Statistical Methods for Signal Processing with Application to Automatic Accent Recognition” by Zichen MA, a candidate for the degree of Master of Science in Applied Statistics, and hereby approve that it is worthy of acceptance:

Dr. Ernest FOKOUÉ
Thesis advisor, associate professor, Center for Quality and Applied Statistics

Dr. Joseph VOELKEL
Committee member, professor, Center for Quality and Applied Statistics

Dr. Steven LALONDE
Committee member, associate professor, Center for Quality and Applied Statistics
Declaration of Authorship

I, Zichen Ma, declare that this thesis titled, “Statistical Methods for Signal Processing with Application to Automatic Accent Recognition” and the work presented in it are my own. I confirm that:

■ This work was done wholly or mainly while in candidature for a research degree at this University.

■ Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.

■ Where I have consulted the published work of others, this is always clearly attributed.

■ Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.

■ I have acknowledged all main sources of help.

■ Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ____________________________________________________________

Date: ______________________________________________________________
“We balance probabilities and choose the most likely. It is the scientific use of the imagination.”

Sir Arthur Conan Doyle, *The Hound of Baskervilles*
The problem of classification of people based on their phonetic features of accents is posted. This thesis intends to construct an automatic accent recognition machine that can accomplish this classification task with a decent accuracy. The machine consists of two crucial steps, feature extraction and pattern recognition. In the thesis, we review and explore multiple techniques of both steps in great detail. Specifically, in terms of feature extraction, we explore the techniques of principal component analysis and cepstral analysis, and in terms of pattern recognition, we explore the algorithms of discriminant function, support vector machine, and k-nearest neighbors. Since signal data usually exhibit the feature of High Dimension Low Sample Size, it is crucial in the automatic accent recognition task to reduce the dimensionality.

Two studies are constructed in which speech signals are collected and a binary classification of American English accent and non-American English accent is performed. In the first study, a total of 330 speech signals, without the disturbance of noise, of an average dimensionality of 44,050 are classified into two categories. In the time domain, the dimensionality is reduced to 250 using principal component analysis. Although the in-sample prediction shows an optimistic accuracy of over 90%, the out-of-sample prediction accuracy using cross-validation is as low as 60%. Alternatively, a feature extraction technique in the frequency domain, cepstral analysis, is implemented instead of principal component analysis, by which a special type of feature called mel-frequency cepstral coefficients is extracted and the dimensionality is reduced to some values between 12 and 39. The out-of-sample prediction accuracy can be as high as around 95%. Although cepstral analysis demonstrates itself as a powerful tool in accent recognition, through a second study we further show that it may quickly fail when there is evident amount of noise in the signal. The prediction performance is reduced to 80% or lower, depending on the amplitude of the noise and the length of the signals.
Acknowledgements

I am especially grateful to my thesis advisor Dr. Ernest Fokoué for his teaching of all the concepts and notions that are extremely useful in the completion of this research work. I enjoyed every single conversation with him and have learned so much through these conversations. I will remember for my life his enthusiasm, patience, and kindheartedness in sharing his beautiful knowledge and experience with me.

I wish to thank Dr. Steven LaLonde and Professor Joseph Voelkel for their kindly acceptance of being in my thesis committee and their precious advises to my research.

I wish to sincerely thank the John D. Hromi Center for Quality and Applied Statistics for accepting me as a student in the master’s program and paying part of my tuition. I would also thank every single faculty and staff member in the department for their teach and help in the past two years.

I would like to thank Mr. Kyle Sauln and Ms. Sara Smacher for their help in the voice collection and all the participants for their generous participation.

I would give my deepest thanks to my parents, who are always supporting me in accomplishing this master’s degree overseas in the United States, which is an opportunity that they never dreamed of for themselves.

Last, this thesis is typed and edited in \LaTeX.
# Contents

Committee Approval i

Declaration of Authorship ii

Abstract iv

Acknowledgements v

Contents vi

List of Figures ix

List of Tables x

1 Introduction 1

1.1 Motivation and problems ........................................... 1
1.2 Background: Signal processing .................................... 2
  1.2.1 Signal and signal processing ................................. 2
  1.2.2 Speech processing and its subfields ......................... 3
  1.2.3 Continuous-time and discrete-time signals .................. 4
  1.2.4 Signal as big data of HDLSS variety ....................... 4
  1.2.5 Graphical representations of signal ........................... 5
1.3 Background: Statistical learning .................................. 5
  1.3.1 Learning problems ............................................. 5
  1.3.2 Supervised learning ............................................ 6
  1.3.3 Learning problems in the HDLSS context .................... 7
1.4 Aim of the thesis .................................................. 7
1.5 Thesis organization ................................................ 8

2 Techniques in Feature Extraction 10

2.1 Principal component analysis ...................................... 11
  2.1.1 Classical PCA .................................................. 11
  2.1.2 PCA on dual space ............................................. 12
2.2 Cepstral analysis .................................................. 13
## Contents

2.2.1 Pre-emphasis and finite impulse response ........................................ 14  
2.2.2 Window function and short time Fourier transform .............................. 15  
2.2.3 Filter banks and mel-scale .............................................................. 17  
2.2.4 Computation of MFCCs ................................................................ 20

3 Statistical Classification ................................................................. 21
  3.1 Binary classifier .............................................................................. 22  
    3.1.1 Discriminant function ................................................................. 22  
    3.1.2 Support vector machine ............................................................... 24  
    3.1.3 K-nearest neighbors ..................................................................... 26  
  3.2 Cross-validation ............................................................................. 28  
    3.2.1 K-fold cross-validation and confusion matrix ............................... 29  
    3.2.2 Hold-out method and stratified sampling .................................... 30

4 Implementation of Automatic Accent Recognition - Study and Results .... 32
  4.1 The First Study ................................................................................. 32  
  4.2 Analysis on Time Domain ................................................................. 33  
    4.2.1 Dimension Reduction .................................................................... 35  
    4.2.2 Comparison of classifiers .............................................................. 36  
    4.2.3 How many PC’s to keep? ................................................................. 37  
  4.3 Analysis on frequency domain ......................................................... 38  
    4.3.1 Predictive performance ................................................................. 38  
    4.3.2 Computational performance ......................................................... 41  
  4.4 Further Discussion ........................................................................... 44

5 Implementation of Automatic Accent Recognition - Effect of Noise ....... 45
  5.1 Continuation of the first study ........................................................... 45  
    5.1.1 Perturbing signal with noise ......................................................... 45  
    5.1.2 Predictive performance on noisy data ......................................... 46  
  5.2 The second study ............................................................................... 47  
    5.2.1 Description of the study ............................................................... 47  
    5.2.2 Analysis on frequency domain .................................................... 49  
  5.3 Further Discussion ........................................................................... 50

6 Conclusion ......................................................................................... 52  
  6.1 Performance of accent recognition machine .................................... 52  
    6.1.1 Time or frequency? ......................................................................... 52  
    6.1.2 Which classifiers to use? ............................................................... 52  
    6.1.3 When does it fail? ......................................................................... 53  
  6.2 Future work ..................................................................................... 53

A Selected R Code ............................................................................... 55  
  A.1 R Code in Performing Analysis on Time Domain .............................. 55  
  A.2 R Code in Performing Analysis on Frequency Domain ..................... 57
Bibliography 59
# List of Figures

1.1 Block diagram of the recognition task .......................... 1  
1.2 Waveform and spectrogram ........................................ 6  

2.1 Pre-emphasis ......................................................... 15  
2.2 Fourier decomposition ............................................... 16  
2.3 Generalized Hamming window ...................................... 17  
2.4 Effect of Hamming window ........................................... 18  
2.5 Relation between mel-scale and hertz-scale ......................... 19  
2.6 Filter bank .......................................................... 19  

3.1 Linear optimal hyperplane in SVM ................................. 24  
3.2 Nonlinear optimal hyperplane in SVM ............................... 25  
3.3 Discrepancy between training error and test error .................. 28  

4.1 Comparison of waveforms ............................................ 34  
4.2 Histogram of length of amplitude vectors ............................ 34  
4.3 Plot of the eigenvalues ............................................... 35  
4.4 Test error on time domain ............................................. 37  
4.5 Percent variation vs. mean prediction accuracy ....................... 38  
4.6 Comparison of spectrograms ......................................... 39  
4.7 Comparison of prediction accuracy on frequency domain ............ 40  
4.8 Relative prediction accuracy on frequency domain .................. 41  
4.9 Computational time for feature extraction ........................... 42  
4.10 Comparison of computational time of classifiers ...................... 42  
4.11 Relative computational time ......................................... 43  

5.1 Noise with different levels of amplitude ......................... 46  
5.2 Mean prediction accuracy with noisy data .......................... 48  
5.3 A speech signal in the second study ............................... 49
List of Tables

3.1 Confusion matrix ................................................. 29
4.1 Summary of study ................................................. 33
4.2 Number of Principal Components .............................. 36
4.3 Training performance on time domain ....................... 36
4.4 Predictive performance on frequency domain ............. 40
5.1 Predictive performance with noisy data .................... 47
5.2 Predictive performance in the second study ............... 50
For my father
Chapter 1

Introduction

1.1 Motivation and problems

Our work is mainly motivated by a question in the field of signal processing: *can people be statistically grouped into different classes by their accent features of some spoken language?* In such questions, several points should be drawn attention to. First, we are solely interested in the features at the stage of phonetics, not involving any semantics analysis. In linguistics, phonetics is the study of the sounds of the speech whereas semantics involves the study of the meaning of the speech ([Hayes, 2010], [Johnson, 2003]). That is to say, we are to perform the classification based only on the sounds, not taking care of what the speech exactly is. Second, such classification should be performed through statistical methods. Though other methods can be used to accomplish the classification, mainly in linguistics ([Biadsy, 2011]), our work focuses on the statistical methods. Third, a basis of “some spoken language” should be chosen upon which different accents can be compared to each other. In this research, the basis is English, or more specifically, American English.

Following the logical idea of this question, it is of interest to analyze the statistical classification of people according to their accents of speaking English. Figure 1.1 gives an abstract block diagram of such automatic accent recognition. The recognition task is achieved through a set, or rather, a chain of machines. Starting from some raw input speeches, some feature extraction techniques are used to extract the useful phonetic

![Figure 1.1: A block diagram of the automatic accent recognition task](image-url)
features in the speech, and then the features are sent to some pattern recognition machines, that is, some classification techniques, to accomplish the classification. At last, the outputs are the classes, usually represented by a set of integer indices.

Thus, the general question posted in the beginning boils down to the following specific questions:

- How are we to perform the feature extraction?
- How are we to perform the pattern recognition?
- If there are multiple techniques that can be chosen at each stage, is there a best one? If so, which combination of techniques should be preferred?
- Is the predictive performance of such combination of techniques in feature extraction and pattern recognition consistently stable?

Combining the above specific questions together, the goal of our research is indeed: **Given a set of speech signals in English, construct an automatic accent machine that classifies speakers into different classes of accents.** For instance, in the sense of binary classification, we can group the speakers into two categories that indicate if they have American accent.

### 1.2 Background: Signal processing

#### 1.2.1 Signal and signal processing

In [Priemer, 1991], a signal is defined as “a function that conveys information about the behavior or attributes of some phenomenon.” Such information may include but not limit to audio, video, speech, image, conversation, music. Physically, any vibration over time or over space would create signals. If the quantity exhibits vibration over time, it generates any kinds of patterned audio signals, while a vibration over space would create image signals.

Signal processing is an area of study that deals with the analysis of signals. In [Oppenheim and Schafer, 1975], the authors dated the start of signal processing back to as early as 17th century, while they also claimed that the modern signal processing started from 1940s.

Below are some subfields in signal processing:
• **Audio processing**: The study of electrical signals that represent sound.

• **Image processing**: The study of signal processing in which the input is image.

• **Video processing**: The study of video signals, especially movies.

• **Array processing**: The study of signals from arrays of sensors.

In this thesis, the word “signal” only means the vibration over time or simply audio signal. Also, although noise is often defined as random disturbance of signal, in this thesis we sometimes use the word “signal” as the mixture of pure signal and noise for simplicity.

### 1.2.2 Speech processing and its subfields

As a subfield of signal processing, speech signal processing, or speech processing for short, deals with speech signals and its process. That is, the input signals are human speech. Speech processing has risen as an important area in the sense that it is closely related to natural language processing and artificial intelligence in general ([Boves and de Vethd, 1996], [Liu and Hansen, 2010], [Hardcastle and Laver, 1999]). Though all intend to perform the analyses based on speech signals, its subjects usually exhibit some substantial differences. Below is a list of some of the subfields in speech processing ([Nerbonne, 2003]).

• **Speech recognition**: The study that deals with the translation between signal and text.

• **Speaker recognition**: The study that deals with the identification of specific speakers based on speech signals.

• **Dialect recognition**: The study of recognizing the dialect of the object of a speech signal.

• **Accent recognition**: The study of recognizing the accent of the object of a speech signal.

Specifically, although there is a connection between dialect recognition and accent recognition, these two terms should not be used interchangeably ([Torres-Carrasquillo et al., 2004]). Dialect recognition performs the classification of speech signals based on the dialect being used, which is the different expressions being used by different groups of people that intend to convey the same idea, whereas accent recognition intends to accomplish the same task solely based on the accent, which is merely the difference of
pronunciation. Thus, dialect recognition involves analysis in both phonetics and semantics, but usually accent recognition only involves phonetics. For instance, the sentences “He has British accent” and “He speaks British dialect” convey slightly different meanings. In terms of analysis, accent recognition is often text-independent while dialect recognition is usually text-dependent.

1.2.3 Continuous-time and discrete-time signals

Signals can be categorized in various ways ([Huang et al., 2001]). In terms of time, signals can be categorized into:

- **Continuous-time signal**: any real-valued function that is defined at a continuous time $t$ in a finite or infinite interval.

- **Discrete-time signal**: a function that maps from integers, representing discrete time, to any real numbers. In this sense, a signal is simply a time series.

Intuitively, continuous-time signals are real signals that come from speech, telephone, or other sources. Usually, discrete-time signals are obtained through sampling from continuous-time signals using digital devices. Most of the time the signals being used for analysis are discrete-time signals or rather digital signals.

1.2.4 Signal as big data of HDLSS variety

As stated above, the signals in research analyses are usually discrete-time signals and are collected from sampling. It is of great importance to recognize that signal becomes “big data” due to this sampling procedure. A sampling rate is defined as the number of samples being taken from the signal per unit time, usually in seconds ([Shannon, 1949]). That is, how frequent it is to draw a sample from the continuous signal. To preserve most of the information from the signal, it is crucial to keep this frequency as a relatively large number. For instance, if we are to sample a signal with a sampling rate of 8,000 Hz, that is, to draw 8,000 samples from the signal per second, a 10-second digital signal would contain 80,000 elements in it. For most recording devices, a sampling rate of 44,100 Hz or 48,000 Hz is considered as moderate.

In terms of data matrix, if we arrange these elements from one single piece of signal of 1 second with a sampling rate of 44,100 Hz as a row vector, we need to collect 44,100 pieces of signals or more fitting into the matrix to make it at least a square matrix. And yet with limited resource, one can hardly this huge task of data collection. Thus, a lot of
researches in signal processing exhibit the feature of High Dimension Low Sample Size (HDLSS).

Formally, an $n \times p$ data matrix $\mathbf{X}$ exhibits the form below.

$$
\mathbf{X} = \begin{pmatrix}
    x_{11} & x_{12} & \cdots & x_{1p} \\
    x_{21} & x_{22} & \cdots & x_{2p} \\
    \vdots & \vdots & \ddots & \vdots \\
    x_{n1} & x_{n2} & \cdots & x_{np}
\end{pmatrix}.
$$

It is said to be HDLSS if $n \ll p$.

### 1.2.5 Graphical representations of signal

So far, a question should be asked naturally: What exactly is sampled from the signal? Or, in terms of time series, what indeed does the time series consist of? Usually on time domain, the signal is a time series of amplitude, the instant loudness of the signal at the time being collected. It is always helpful to present the signals graphically. In general, there are two types of graphical representations that are commonly used:

- **Waveform**: a time series plot of the amplitude of a signal.
- **Spectrogram**: a visual representation of the frequency of a signal. That is, how much of a signal at each time period lies on each of the frequency bands.

As a crucial term, “frequency” shall be introduced in Chapter 2. Figure 1.2 gives a comparison of waveform and spectrogram of a male voice speaking the word “approximation”. Notice that the waveform is rather simple, and the spectrogram is much more complex and interesting, in which the X-axis represents time, Y-axis represents frequency, and the shade on the graph represents how much energy there is at each period of time and each band of frequency.

### 1.3 Background: Statistical learning

#### 1.3.1 Learning problems

It is rather difficult to define what a learning problem is, since statistical learning or rather machine learning is a very broad concept and contains various branches of knowledge. In [Vapnik, 1995], the author addressed that statistical learning problem is “so
Chapter 1. Introduction

1.3.2 Supervised learning

In general, learning problems can be categorized into two classes, supervised learning and unsupervised learning, based on the availability of the knowledge of the target ([Vapnik, 1995]). These two types of problems can be defined as:

- *Supervised learning*: the type of learning problems that seeks to find the pattern between the independent variables (or features) and the known dependent variable (or target) $y$.

- *Unsupervised learning*: the type of learning problems in which there is no already known target.

Although the difference between the two types of problems only lies on whether or not the target is known, the techniques being used to solve the problems are very different. For instance, as a simple example, regression analysis is considered as a supervised
learning simply because there is a known target that is being regressed on. Without this crucial piece of knowledge, such analysis can never be established on (especially when the target is continuous.)

In detail, the solution of a supervised problem consists of three parts ([Vapnik, 1995]):

- Some data that consist of feature vectors $x$.
- A target $y$ that contains the output of every feature vector $x$.
- A learning machine that contains a set of functions which ultimately achieves the mapping from input space to output space.

Specifically, if the target $y$ is discrete and contains the indices that indicate the distinct class to which the feature vectors $x$ belong, such supervised learning problems are called classification. In such case, the learning machine is usually called a classifier.

1.3.3 Learning problems in the HDLSS context

In the context of data that exhibit HDLSS feature, learning problems are sometimes difficult to deal with, mainly because such problems are not well-posed. According to [Hadamard, 1923], a well-posed problem should have the following properties:

- A solution exists.
- The solution is unique.
- The solution’s behavior changes continuously with the initial conditions.

The violation of any one of the above properties would cause the problem to deteriorate to an ill-posed problem. Thus, an important task in this research is to seek some methods through which the ill-posed problems can be transformed into well-posed problems.

1.4 Aim of the thesis

Our research intends to give insights in the following:

- Dimension reduction: We show that dimension reduction can be performed in different ways through feature extraction in terms of signal processing. And in this case it is feasible to transform the ill-posed problem to well-posed. Also, the
domain-specific technique of feature extraction performs better than the general technique since it reduces the dimensionality of the data and in the mean time extracts more meaningful features.

- Classification: We demonstrate the difference of classifiers in performing the classification task through some procedures of model validation.

- Robustness of the performance: We also examine if the predictive performance is affected significantly by noise. That is, when there is evident amount of noise, would the accent recognition machine still perform relatively convincing prediction results or degrade quickly?

1.5 Thesis organization

In order to thoroughly convey the idea and results of our research, the thesis consists of 6 chapters, with this chapter being the first and presenting an introduction to the topic and a brief background of the context of the research.

Chapter 2 and 3 can be considered as Part One, in which some theory and facts are reviewed. Chapter 2 reviews techniques in feature extraction. We start with the general technique of principal component analysis and elaborate the situation in which the data matrix exhibit high dimension low sample size feature so that the classical principal analysis is infeasible and an alternative should be preferred. Then we move on to a domain-specific technique of feature extraction, namely, the cepstral analysis. We thoroughly work through the steps following which the computation of a useful type of feature, the mel-frequency cepstral coefficients, is performed.

Chapter 3 explores some basic concepts in statistical classification. We first review three types of binary classifiers, through which the features are able to be classified into distinct classes. Although the techniques are aiming at the same goal of accomplishing the classification, we intend to illustrate the differences of these techniques.

Chapter 4 and 5 can be considered as Part Two, in which the experiments and results that represents the performance of the implementation of automatic accent recognition are presented. Chapter 4 starts with the description of the first experiment, in which the pure signals were collected, and continues to the analysis in time domain and frequency domain based on this dataset.

Chapter 5 intends to examine the effect of noise in automatic accent recognition. It consists of two parts. The first part is a continuation of Chapter 4, in which the pure signals are artificially perturbed with noise and the analysis on frequency domain is
performed based on the contaminated sounds. The second part describes the second experiment being conducted in a real-life surroundings in which the signals are with noise from the beginning. The analysis on frequency domain is performed as well.

At last, Chapter 6 concludes the research by highlighting both the powerfulness and the weakness of the techniques being used in the work. And then, it gives some substantial topics that should be considered in the future work.
Chapter 2

Techniques in Feature Extraction

This chapter reviews two alternative methods in feature extraction, principal component analysis (PCA) and cepstral analysis. PCA can be performed in the time domain, while cepstral analysis is naturally implemented on frequency domain. Both techniques have the advantage of reducing dimensionality together with feature extraction.

Section 2.1 reviews the technique of principal component analysis. Though normally considered as a method of remedying multicollinearity, and thus of reducing dimensionality, PCA essentially achieves the goal of feature extraction via orthogonal transform. Typically, the principal components can be obtained through eigenvalue decomposition of $X^TX$ and orthogonal transformation, where $X$ is the data matrix, but this is not achievable when the data exhibits structure of high dimension and low sample size, as is usually encountered in signal processing. In the latter case, PCA has to be performed in terms of the dual space $XX^T$.

Section 2.2 reviews the method of cepstral analysis, a powerful tool for feature extraction, specifically designed for signal processing. Starting from some pre-processing work on time domain, the signal is transformed onto frequency domain. And the goal is to extract a type of feature called mel-frequency cepstral coefficients (MFCCs) via nonlinear mapping and discrete cosine transform. The MFCC vectors are orthogonal to each other, and by controlling how many vectors to preserve, we can control the dimensionality of the input.
2.1 Principal component analysis

2.1.1 Classical PCA

As one of the most important applications in linear algebra and possibly one of the most influential multivariate statistical learning techniques, principal component analysis can be dated back to [Pearson, 1901], or, as its modern formalization, to [Hotelling, 1933], who also created the term principal component. Its goal is to answer the following question: For a dataset that contains possibly inter-correlated variables, can we obtain another dataset that is the linear combination of the original basis and that re-expresses the data optimally? ([Abdi and Williams, 2010]) Given a large dataset that contains numerous variables, the problem of multi-collinearity, that is, variables are heavily correlated to each other, is almost always inevitable. Thus it is not meaningful to preserve the original dimensionality. Principal component analysis helps to reduce the high dimensionality and to reveal the feature of the data.

Specifically, let $X_{n \times p}$ be a given dataset that contains $n$ observations and $p$ variables and $n > p$. Each of the column vector $x_i$ in the matrix

$$X_{n \times p} = [x_1, x_2, \ldots, x_p]$$

is a random variable with mean $E(X)$ and finite variance. Thus the covariance matrix $\Sigma$ can be written as

$$\Sigma = E \left[ (X - E[X])^T (X - E[X]) \right].$$

If we assume that $x_i$’s are centered at 0, the equation above can be simplified to

$$\Sigma = E \left[ X^T X \right].$$

And the corresponding sample covariance matrix is

$$S = \frac{1}{n-1} \begin{pmatrix}
    x_1^T x_1 & x_1^T x_2 & \cdots & x_1^T x_p \\
    x_2^T x_1 & x_2^T x_2 & \cdots & x_2^T x_p \\
    \vdots & \vdots & \ddots & \vdots \\
    x_p^T x_1 & x_p^T x_2 & \cdots & x_p^T x_p
\end{pmatrix}.$$

Notice that in the above symmetric positive definite matrix, we have the variance of the variables on the diagonal entries and covariance on the off-diagonal entries. Since we would like to reduce as much the multicollinearity as possible, the goal is to find a set of linear transformations of $X$ that minimizes the covariance, or the off-diagonal entries, while preserving as much the variation as possible. The optimal solution for this
problem is to re-construct a dataset $Z$ of which the covariance matrix is in fact diagonal, that is, the covariance between any two variables is exactly 0. It can be shown that this optimal solution is indeed the eigenvalue decomposition of the matrix $X^T X$ ([Johnson and Wichern, 2007]).

Based on the well-known theorem from linear algebra that any square symmetric matrix can be orthogonally (orthonormally) diagonalized, we have

$$X^T X = U \Lambda U^T \quad (2.1)$$

where $U$ is a $p \times p$ orthogonal matrix of which the column vectors are the eigenvectors and $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_p)$ is the diagonal matrix of which the diagonal entries are the eigenvalues.

Given the eigen-pairs $[(\lambda_1, u_1), (\lambda_2, u_2), \ldots, (\lambda_p, u_p)]$, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$, each element $z_{ij}$ in the $j$th principal component $Z_j$ can be written as a linear combination of the column vectors in the original data matrix.

$$z_{ij} = x_i^T u_j, \quad i = 1, 2, \ldots, n, j = 1, 2, \ldots, p. \quad (2.2)$$

An important fact of this decomposition is that the sum of the eigenvalues equates to the trace of the matrix being decomposed, that is,

$$\sum_{i=1}^{p} \lambda_i = \text{trace}(X^T X)$$

and notice again that the trace is simply the sum of variance of the variables, that is, the total variance. Thus, based on the magnitude of the eigenvalues, it is often not necessary to preserve all $p$ principal components. If, given a positive number $q < p$, the last $(p - q)$ eigenvalues are relatively small and we can arbitrarily conclude that those variables essentially contribute very little to the total randomness of the data, it is natural that we only preserve the first $q$ principal components, or the important information, and thus achieve the task of dimension reduction ([Johnson and Wichern, 2007]).

### 2.1.2 PCA on dual space

In the above subsection, we assume that the number of observations is greater than the number of variables in the data matrix $X$, so that the matrix $X^T X$ is nonsingular, or equivalently of full rank. However, in the HDLSS case, in which the number of
observations is smaller than the number of variables, $X^TX$ is not of full rank any more, or, in some extreme cases when the ratio between $n$ and $p$ is extremely small, the matrix cannot even be estimated computationally. Thus, the regular PCA cannot be performed in such situation. Fortunately, it has been shown that PCA is achievable in terms of the dual space $XX^T$ ([Shen et al., 2013], [Yata and Aoshima, 2010]).

We start from the fact that the matrices $X^TX$ and $XX^T$ share the same non-zero eigenvalues ([Strang, 2009]). Notice that in Equation 2.1, the matrix $U$ is orthonormal, that is, $U^{-1}U = U^TU = I$, where $I$ is the identity matrix. Thus, we have

$$X^TXU = U\Lambda U^TU = U\Lambda.$$  \hspace{1cm} (2.3)

Then by pre-multiplying both sides of Equation 2.3 by $X$ and using Equation 2.2, we have

$$XX^TXU = XUA$$

and

$$XX^TZ = Z\Lambda.$$  \hspace{1cm} (2.4)

Equation 2.4 indicates that instead of dealing with matrix $X^TX$, we can perform eigenvalue decomposition on the dual matrix $XX^T$ and obtain the component scores directly from matrix $V$ ([Jung and Marron, 2009], [Jung, 2011]).

2.2 Cepstral analysis

As is shown by Huang et al. ([Huang et al., 2001]), the main idea of the so-called cepstral analysis is to transform the signal from time domain to frequency domain and
to map the transformed signal in hertz onto Mel-scale due to the fact that 1 kHz is a threshold of humans’ hearing ability. Human ears are less sensitive to sound with frequency above that threshold. Cepstral analysis is considered a powerful algorithm in fields such as speaker recognition ([Tivari, 2010], [Hasan et al., 2004], [Hanani, 2012], [Chen and Luo, 2009]) and musical instrument analysis ([Logan, 2000], [Sturm et al., 2010]). The calculation of MFCCs includes the following steps:

- Pre-emphasis filtering
- Take the absolute value of the short time Fourier transformation using windowing
- Warp to auditory frequency scale (Mel-scale)
- Take the discrete cosine transformation of the log-auditory-spectrum
- Return the first \( q \) MFCCs

### 2.2.1 Pre-emphasis and finite impulse response

Let \( x[n] \) be a discrete signal, where \( n \) is the index. Before the signal \( x[n] \) is transformed to frequency domain or is passed to any formal analysis, it is usually necessary to consider some preliminary analysis, especially when the signal contains an evident background noise. Finite impulse response (FIR) filter is one of these techniques. Usually, a finite impulse response filter is defined as

\[
    s[n] = b_0 x[n] + b_1 x[n - 1] + \cdots + b_N x[n - N]
\]

\[
    = \sum_{i=0}^{N} b_i \cdot x[n - i],
\]

where \( x[n] \) is the input signal and \( s[n] \) is the output. Notice that FIR filter can be considered as a smoothing technique over a finite time period \( N \) given some conditions on the coefficients \( b_i \). Specifically, the first-order FIR filter

\[
    s[n] = x[n] + \alpha x[n - 1]
\]

(2.5)

is called pre-emphasis and can be determined as a high-pass filter or a low-pass filter based on the value of \( \alpha \). When \( \alpha > 0 \), it is a low-pass filter, which means that low frequency is emphasized while high frequency is de-emphasized. It is a high-pass filter when \( \alpha < 0 \) and the high frequency is emphasized in this situation.

In the pre-emphasis step, \( \alpha \) is usually controlled between -1 and -0.95, which results in a stable high-pass filter that has a strong emphasis on high frequency bands. Pre-emphasis
is important as a preliminary process due to the fact that the signal-to-noise ratio (SNR) is lower at low frequency. By emphasizing the high frequency band and de-emphasizing the low frequency band, the pre-emphasis filter is in fact boosting the signal. Figure 2.1 shows the effect of pre-emphasis. Notice that the voice portion of the signal is boosted while the non-voice portion, or the transition part between voice portions, is damped.

![Figure 2.1: The effect of pre-emphasis with α = −0.95](image)

2.2.2 Window function and short time Fourier transform

Named after the French mathematician Joseph Fourier (1822), Fourier transform has been playing an important role in signal processing in the sense that it is the crucial link between time domain and frequency domain. Given a periodical but not necessarily sinusoidal signal function over time, such as a piece of music or speech, the goal is to represent this signal by some pure sinusoidal functions, that is, sine and cosine waves.

Specifically, given a signal function $f(x)$, the Fourier transform can be written as

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-2\pi j \xi x} dx,$$

for any real number $\xi$, where $j$ is the complex unit. When the independent variable $x$ represents time with unit of seconds, the transform variable $\xi$ represents frequency in hertz. The transform utilizes the Euler’s formula

$$e^{ix} = \cos x + i \sin x.$$
and thus we have a integration of sine and cosine waves with different frequencies. Figure 2.2 gives an intuitive idea of Fourier transform. The waveform in panel A is called a square wave. Using Fourier transform, it can be decomposed into the sum of different sinusoidal waves. From panel B to panel D, we add two, three, and four sine waves together, and the approximation is more similar to the true square wave.

An important assumption of Fourier transform is that it assumes that the signal is stationary, but this assumption can hardly hold when the signal is relatively long. Although the amplitude over time is exactly centered at 0 due to the sampling procedure, its variance may vary a lot at different time. This flaw can be remedied by implementing short-time Fourier transform, which states that a signal over a very short time is nearly stationary. Mathematically, it is given by

$$X_a[k] = \sum_{n=0}^{L_w-1} s[n] \cdot w_a[n] \cdot e^{-j2\pi kn/L_w} = \sum_{n=0}^{L_w-1} s[n] \cdot w_a[n] \cdot e^{-j\omega k}, \quad 0 \leq k < L_w \quad (2.6)$$

where $w_a[n]$ is the window function being utilized as a method to truncate a long signal into short pieces, and $L_w$ is the length of the window. By definition, a window function is a mathematical function that is everywhere zero except for a chosen interval. Though there are numerous types of window functions, such as the rectangular window and the triangular window, it is of importance to see that a specific type of window function, the generalized Hamming window, is very prevalent in practice. The form of generalized
Hamming window is given by

\[ w[n] = (1 - \alpha) - \alpha \cos \left( \frac{2\pi n}{L_w - 1} \right), \quad 0 \leq n < L_w \]  

(2.7)

and when \( \alpha = 0.54 \), it is the traditional Hamming window, named after Richard Hamming, and when \( \alpha = 0.5 \), it is a Hanning window, named after Julius von Hann. Figure 2.3 generalized Hamming window with different values of \( \alpha \). Notice that all the functions approach 1 in the center while die down toward 0 at the end. If we are to break the signal into short pieces, an overlap between pieces is necessary and this die-down pattern is crucial to keep the consistency of the signal. Figure 2.4 illustrates the effect of a traditional Hamming window. The ends of the windowed signal are approaching 0 by multiplying the window function and \( s[n] \) together. Usually the window length is chosen to be around 40 milliseconds, with an overlap of around 10 milliseconds between the consecutive windows to keep the continuity of the whole signal.

### 2.2.3 Filter banks and mel-scale

Now we introduce the concept of filter banks. A filter bank is a set of \( m \) filters that is implemented to a signal in order to modify and re-group it into some components or features. In practice, it has been shown that the triangular filter is very useful. A
triangular filter of length $L_f$ can be written as

$$M_m[k] = 1 - \left| \frac{k - \frac{L_f - 1}{2}}{\frac{L_f - 1}{2}} \right|, \quad 0 \leq k < L_f \quad (2.8)$$

We can assign the filter bank linearly to a signal on frequency domain, that is, to arrange the filters in a way such that every filter covers a frequency band with the same length. Yet this method is not efficient enough on a hertz scale since the hearing ability of human is more sensitive to the frequency band from 20 to 1000 hertz. Thus it is less efficient to assign the same length to a filter at higher frequency as to a filter at lower frequency. Fortunately, the mel-scale, named by Stevens, Volkman, and Newman (1937), compensates this problem through a nonlinear mapping between hertz-scale and mel-scale using logarithm. This transform can be done by

$$mel = \begin{cases} f & f \leq 1000 \\ 2595 \log_{10} \left( 1 + \frac{f}{700} \right) & f > 1000 \end{cases} \quad (2.9)$$

where $f$ denotes the frequency with hertz scale. And Figure 2.5 illustrates this nonlinear relationship. Now we can group the signal using the trick of mel-scale by assigning the filters linearly on mel-scale. The filters at higher frequency would automatically cover a wider range after being transformed back to hertz-scale. Figure 2.6 shows a filter bank of 20 filters on frequency domain of hertz scale with sampling rate of 8000 Hz. Notice
Figure 2.5: The transform between mel-scale and hertz-scale is a nonlinear mapping.

Figure 2.6: A filter bank of 20 filters over 8000 Hz.
again that as the frequency increases, the filters would cover a wider band. And again a short length of overlap is necessary to keep the continuity.

### 2.2.4 Computation of MFCCs

Now we can perform the final computation of MFCCs with the help of filter banks and mel-scale. First, we can compute the log-energy for each filter, which is written as

\[
S[m] = \ln \left[ \sum_{k=0}^{N-1} |x_a[k]|^2 M_m[k] \right], 0 < m \leq M
\]  

(2.10)

Notice that within the summation sign, we multiply the energy and the filter together to perform the grouping of feature.

Finally, the mel-frequency cepstral coefficients are given by the discrete cosine transform of the above output:

\[
c[q] = \sum_{m=0}^{M-1} \left[ S[m] \cos \left( \frac{\pi q(m - \frac{1}{2})}{M} \right) \right], 0 < m \leq M
\]  

(2.11)

Some points should be drawn attention to. First, we can arbitrarily control the number \( q \) to decide how many coefficients to preserve. In practice, this number is usually between 12 and 40. Second, for a complete signal, the output of MFCCs is an \( N_w \times q \) matrix where \( N_w \) is the number of windows and \( q \) is the number of MFCCs. In order to pass it to a vector-based pattern recognition machine, the data should be re-grouped or summarized. The simplest way to perform this summary of data is to take the column average of the matrix as a vector to represent the features of a specific signal. More complex methods include taking the column variance into account, or modelling each matrix as a Gaussian mixture, the latter of which contains intense computation. In this thesis, we only use the column mean vector of each matrix as the representation of that signal. Finally, the last step of the computation, discrete cosine transform, is derived from Karhunen-Loève Theorem, which is the basis of principal component analysis, so that the computed coefficient vectors are designed to be orthogonal to each other. Thus, we have achieved the same task of dimension reduction as PCA while in the mean time we have performed the domain-specific feature extraction.
Chapter 3

Statistical Classification

This chapter reviews some basic concepts in statistical classification. In statistics and machine learning, classification is the problem that categorize a new observation into one of the several classes based on some knowledge of training data and their observed classes. In this sense, classification is a supervised learning problem, and it is significantly different from cluster analysis, which is an unsupervised learning problem that attempts to group similar observations into the same cluster without the help of some training data of which the true classes are known. The most common, or intuitive problems in classification are binary problems, where the target is of only two classes. In this case, the data shall be passed to a classifier, an algorithm or machine that performs the classification. Section 3.1 reviews some of the binary classifiers. After the model is fitted from a training set, it is always of interest to examine if the model has a decent prediction ability. Such examination can be somewhat fulfilled through cross-validation, which will be reviewed in Section 3.2.

In Section 3.1, we go over three different binary classifiers, namely, discriminant function, support vector machine, and k-nearest neighbors. It is of importance to know that all of these algorithms can perform classification for multi-categories, but in this chapter they are introduced only in the binary context. Also, though some classifiers are more robust than the others theoretically, to decide which classifier is the best or the most useful in a specific task is more empirical.

Section 3.2 reviews cross-validation, a crucial technique in checking if the model has strong prediction ability. In this section, we compare the common $K$-fold algorithm and the hold-out method. In the binary classification sense, random sampling is usually not reasonable, thus an alternative method, stratified sampling, is introduced.
3.1 Binary classifier

3.1.1 Discriminant function

A standard approach to supervised classification problems is the discriminant analysis. From a Bayesian perspective, let \( Y \in \{1, 2, \ldots, K\} \) be a discrete target and \( X \) be the data matrix. The binary classification problem can be proposed like this: \textit{given some feature} \( x_i \), \textit{classify the corresponding target} \( y_i \) \textit{into one of the classes}. A straightforward and rather reasonable strategy is to classify \( y_i \) into the most probable class given the data ([James et al., 2013]). Formally, the problem can be written as

\[
\hat{f}(x) = \max_k Pr(Y = k|X = x).
\]

The target function at the right-hand side is a conditional probability and it is directly related to the famous Bayes' Theorem. Given two events \( A \) and \( B \), the conditional probability \( Pr(B|A) \) is given by

\[
Pr(B|A) = \frac{Pr(A|B) \cdot Pr(B)}{Pr(A)},
\]

or simply expresses in words as

Posterior \( \propto \) Likelihood \( \cdot \) Prior.

Intuitively, it states that our knowledge of some fact (that is, the prior) can be updated (that is, the posterior) through taking into account the data (that is, the likelihood).

Invoking this well-known theorem, the posterior probability being maximized can be re-written as the following

\[
Pr(Y = k|X = x) = \frac{f_k(x)\pi_k}{f(x)},
\]

where \( f_k(x) \) is the conditional density of \( x \) in class \( k \), \( \pi_k \) the prior probability of corresponding class \( k \), and \( f(x) \) the marginal density of \( X \). Thus the discriminant analysis is a likelihood-based technique. That is, in order to compute the posterior probability, it is necessary to have some knowledge of the distribution of \( f_k(x) \).

A common assumption when such knowledge is insufficient is to assign the likelihood a Gaussian distribution. That is to write the likelihood as

\[
f_k(x) = (2\pi)^{-p/2}|\Sigma_k|^{-1/2} \exp \left( -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) \right),
\]
Chapter 3. Statistical Classification

Sometimes the marginal distribution of X is difficult to compute, but with the help of a standard trick in Bayesian inference, we can get rid of the denominator completely since it is constant regardless of which class \( y_i \) is in. Thus, we can specify the posterior probability as proportional to the product of the likelihood and the prior probability.

\[
Pr(Y = k | X = x) \propto f_k(x) \cdot \pi_k
\]

\[
= (2\pi)^{-p/2} |\Sigma_k|^{-1/2} \exp \left( -\frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) \right) \cdot \pi_k.
\]

Since maximizing the above equation is the same as maximizing the logarithm of it, given the fact that the logarithm function is always monotonic, the above equation can be rearranged as

\[
\delta_k(x) = -\frac{1}{2} log |\Sigma_k| - \frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) + log(\pi_k). \tag{3.1}
\]

Without further assumptions, Equation 3.1 is usually called the quadratic discriminant function, since it is a quadratic form in terms of \( x \) ([Hastie et al., 2013]). The decision rule is to assign \( x \) to class \( i \) if \( \delta_i(x) > \delta_j(x) \), that is,

\[
\hat{f}_{QDA}(x) = \arg \max_k \delta_k(x) \tag{3.2}
\]

A further assumption that specifies the same covariance matrix to all classes is sometimes applied. That is, the covariance matrices can be written as \( \Sigma_1 = \Sigma_2 = \cdots = \Sigma_K = \Sigma \). Under such condition, Equation 3.1 can be further simplified to

\[
\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + log(\pi_k). \tag{3.3}
\]

Equation 3.3 is usually called the linear discriminant function, since it only involves a linear term of \( x \). The decision rule is the same as quadratic discriminant function.

Although in some applications the Gaussian assumption appears to be useful, it is in fact a rather arbitrary one. Some methods have been considered as the better alternatives to the Gaussian assumption. For instance, flexible mixtures of Gaussian density can be fitted to the data and the discriminant analysis can be performed in terms of Gaussian mixture model ([Hastie and Tibshirani, 1996]). Or in a more general sense, the conditional densities can be estimated using kernel methods and the classification is performed based on kernel density estimation ([Baudat and Anouar, 2000]).
3.1.2 Support vector machine

Ever since its invention in [Cortes and Vapnik, 1995], support vector machine has been demonstrated as the state-of-the-art technique in binary classification. In its simplest case, in which there exists a linear decision boundary, or a hyperplane, that can completely separate the data of two classes. SVM deals with the question of what a best separation of the data should be. It concludes that the best separation is the solution of an optimization problem that seeks to maximize the distance between any observations and the linear boundary ([Clarke et al., 2009]).

Mathematically, a hyperplane on a $p$-dimensional space can be defined as

$$w \cdot x - b = 0$$

where $w$ is a coefficient vector, $x$ is a point on the $p$-dimensional space and $b$ is a constant. For a dataset $D = \{(x_i, y_i) | x_i \in \mathbb{R}^p, y_i \in \{-1, 1\}\}_{i=1}^{n}$, the optimal separating hyperplane is shown on Figure 3.1. Specifically, observations lying on the dot lines are the “support vectors” and observations fall beyond those two lines are irrelevant to the problem. From geometry, the distance between a support vector and the optimal hyperplane is $\frac{1}{\|w\|}$. Conventionally, this distance is called the margin. Thus, the optimal separating hyperplane can be found by maximizing the margin, which is equivalent to minimizing $\|w\|$, subject to the condition that all observations are either lying on the support hyperplanes or beyond. Mathematically, minimizing $\|w\|$ directly is difficult but

![Figure 3.1: The optimal hyperplane is shown as a solid line.](image)
Chapter 3. Statistical Classification

this minimization is the same as to minimize $\|w\|^2$. Thus, this problem formalization can be written as

$$\arg \min_{(w,b)} \frac{1}{2} \|w\|^2$$

subject to: $y_i(w \cdot x_i - b) \geq 1$.

The constant $\frac{1}{2}$ in the target function is merely for mathematical simplicity. It can be shown that the above optimization problem can be solved through quadratic programming and the decision boundary of SVM is given by

$$\hat{f}_{SVM}(x) = \text{sign} \left( \sum \hat{\alpha}_i y_i x_i^T x + \hat{b} \right)$$

(3.5)

where $\hat{\alpha}$ and $\hat{b}$ are the estimated coefficients. Notice that in the summation sign, the features appear in terms of inner products.

Further, if the data is not separable in the linear case but can be mapped to linear separation through a function $\phi$, as is shown in Figure 3.2, the optimization problem has to be modified to

$$\arg \min_{(w,b)} \frac{1}{2} \|w\|^2$$

subject to: $y_i(w \cdot \phi(x_i) - b) \geq 1$.

![Figure 3.2: The decision boundary in the nonlinear separable case.](image)

It is obvious that in order to have a linear solution to the problem, the function $\phi$ must transform the data to a higher dimension. The general SVM decision boundary is simply to replace the $x$ vectors into a function $\phi(x)$.

$$\hat{f}_{SVM}(x) = \text{sign} \left( \sum \hat{\alpha}_i y_i \phi(x_i) \cdot \phi(x) + \hat{b} \right)$$

(3.6)
It is of extreme importance to notice that the nonlinear mapping function $\phi$ appears in the decision function in the sense of feature space inner product. Computationally, choosing the function $\phi$ can become infeasible quickly, while the well-known kernel trick should be used to avoid the explicit use of $\phi$. Assume a kernel function $K$ can be found so that $K(x_i, x) = \phi(x_i)^T \phi(x)$, the above decision boundary can be modified by replacing the inner product to the kernel function.

$$\hat{f}_{SVM}(x) = \text{sign} \left( \sum_i \hat{\alpha}_i y_i K(x_i, x) + \hat{b} \right)$$

(3.7)

This provides great convenience in which it is not necessary to compute the nonlinear mapping explicitly, but only to perform it implicitly through the kernel. Specifically, one can readily verify that Equation 3.5 is simply a special case of Equation 3.7 with a linear kernel $K(x_i, x) = x_i^T x$. Some common kernel functions include the Radial Basis Function (RBF) kernel

$$K(x_i, x) = \exp(-\gamma \| x_i - x \|^2)$$

and the polynomial kernel

$$K(x_i, x) = (x_i \cdot x + c)^d.$$ 

It is also of importance to know that there is no theoretical proof showing one kernel function is significantly better than others, so that which kernel function to use is an empirical practice and is usually decided through the comparison of prediction accuracy of SVM models using different kernels.

### 3.1.3 K-nearest neighbors

Comparing to the above two techniques, the algorithm of k-nearest neighbors is more intuitive and is often considered as a lazy learning. The idea of this algorithm states like this: given a dataset with known classes, or simply put, a training set, and some new data points with unknown classes, compare the distance of a new point and its first k nearest neighbors and assign the new point to the class that majority of these neighbors lie within. For instance, when $k = 1$, we simply assign a new data point to the same class as its single nearest neighbor ([Fokoué, 2013]). That it is considered as a lazy learning is because no formal model is needed in this algorithm. The only requirements are a dataset in which the classes of observations are already known, some measurement for distance, and an integer $k$.

Mathematically, let $Tr = \{(x_i, y_i) | x_i \in \mathbb{R}^p, y_i \in \{1, 2, \cdots, S\}\}_{i=1}^n$ be a training set and $x^*$ a new data point. The distances of $x^*$ and $x_i$’s are computed by based on some bivariate function $D(\cdot, \cdot)$ and ranked in an increasing order. Specify a set $V_k(x^*) = \{x_i |
\{x_i | D(x^*, x_i) \leq D_{(k)}\}, where \(D_{(k)}\) is the distance between the new point and the \(k\)th nearest neighbor. And the decision boundary can be written as

\[
\hat{f}_{kNN}(x^*) = \arg \max_{j \in \{1, 2, \cdots, S\}} \left\{ \frac{1}{k} \sum_{i=1}^{n} \mathcal{I}(y_i = j) \mathcal{I}(x_i \in \mathcal{V}_k(x^*)) \right\}
\] (3.8)

where \(\mathcal{I}(\cdot)\) is an indicator function.

The crucial part of k-NN algorithm is the distance function. Conventionally, the Euclidean distance

\[
D(x_i, x_j) = \sqrt{\sum_{l=1}^{n} (x_{il} - x_{jl})^2}
\]

or the Manhattan distance

\[
D(x_i, x_j) = \sum_{l=1}^{n} |x_{il} - x_{jl}|
\]

are commonly used in the computation. Also, in binary classification, \(k\) is usually chosen to be an odd integer simply to avoid the tie-up situation. The pseudo-code of k-nearest neighbors algorithm is given below.

**Begin**

- Input: \(Tr = \{(x_i, y_i)|x_i \in \mathbb{R}^p, y_i \in \{1, 2, \cdots, S\}\}_{i=1}^{n}\) as the training set and \(x^*\) a new data point.
- Order \(D(x^*, x_i)\) from lowest to highest.
- Select the \(k\) nearest observations to \(x^*\), \(\mathcal{V}_k(x^*)\).
- Assign to \(x^*\) the most frequent class in \(\mathcal{V}_k(x^*)\).

**End**

A problem of this algorithm arises when the data contain significant amount of noise. That is, if there is significant noise in the training set and there are some outliers in each class, the results of classification using k-NN would degrade. Thus in this sense, k-NN is not a robust algorithm. Some substantial work has been done to remedy this drawback. Also, the algorithm of k-nearest neighbors suffers the curse of dimensionality. That is, when the dimensionality increases, the predictive performance of this algorithm would drastically degenerates.

Even so, a theorem proven in 1960s (\cite{Cover, 1968}) has shown the powerfulness of this algorithm. The theorem states that given the Bayes prediction risk \(R^*\), which is the
lowest prediction risk one can obtain, and the the risk $R$ that is given by the nearest
neighbor algorithm, it has been proven that asymptotically

$$R \leq 2R^*$$  \hspace{1cm} (3.9)

as number of observations $n$ approaches infinity. The Inequality 3.9 states that regardless
of its simple algorithm, the prediction risk would not exceed two times of the lowest risk.

### 3.2 Cross-validation

For models that are used to provide prediction rather than some explanation, it is always
necessary to assess the model fit in the sense of prediction accuracy. When a model is
fitted to some data, there are two types of prediction, the in-sample prediction, in which
the model is tested on the same data that were used to fit it, and the out-of-sample
prediction, in which the model is tested on some other data. The corresponding errors
are called training error (or calibration error) and test error (or prediction error). Shown
in the Figure 3.3, as the model complexity increases, the training error would always
decrease and even approach 0, while the test error behaves as a V-shape. Thus the
training error would cause a false optimism for a relatively complex model. Invented
in Stone (1951), cross-validation intended to remedy the false optimism. The idea of
cross-validation is to construct a model based only on one random portion of the dataset
and validate the prediction accuracy on the other portion.

\[
\text{Figure 3.3: The discrepancy between training error and test error over different}
\text{complexity of the model}
\]
3.2.1 K-fold cross-validation and confusion matrix

There are many different methods to perform cross-validation, but the most common one is the $K$-fold cross-validation algorithm ([James et al., 2013], [Clarke et al., 2009]). Given a random sample, the algorithm follows the procedure below:

Begin

- Randomly divide the sample into $K$ equal portions.
- For $i = 1, 2, \ldots, K$, hold out portion $i$ and fit the model from the rest of the data.
- For $i = 1, 2, \ldots, K$, use the fitted model to predict the holdout sample.
- Average the measure of predictive accuracy over the $K$ different fits.

End

Different from prediction in regression, the prediction error in classification, especially in binary classification, is easy to tell: a prediction error occurs only when the predicted class is different from the true, or the observed, class. Thus for a binary classification, in which the target is only of two classes, positive and negative, the results can be arranged in a confusion matrix, as is the one shown below, which is a two-by-two table showing the relationship between the predicted values and the observed values.

<table>
<thead>
<tr>
<th>Observed class</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted class</td>
<td>Positive</td>
<td>True Positive</td>
</tr>
<tr>
<td></td>
<td>Negative</td>
<td>False Negative</td>
</tr>
</tbody>
</table>

Table 3.1: A general $2 \times 2$ confusion matrix

The cells of the table are the counted numbers and they are defined below.

- **True Positive (TP)**: The situation that the predicted class and the observed class are both positive.
- **False Positive (FP)**: The situation that an observation is predicted as positive but observed as negative.
- **False Negative (FN)**: The situation that an observation is predicted as negative but observed as positive.
• *True Negative (TN)*: The situation that the predicted class and the observed class are both negative.

Thus, it is obvious that the prediction accuracy (ACC) is the sum of diagonal elements over the total number of observations and the prediction error is simply the complement of accuracy.

\[
ACC = \frac{TP + TN}{Total}
\]

### 3.2.2 Hold-out method and stratified sampling

Sometimes when we do not have the luxury of large number of observations, it is not feasible to make the number \(K\) of folds very large. In such case, a hold-out method of cross-validation is usually preferred and its procedure is as follows:

#### Begin

- Randomly divide the sample into 2 portions, a training set containing roughly 70% of the observations, and a test set containing the rest 30% of the data.
- Fit the model from the training set.
- Use the fitted model to predict the test set, and compute the predictive accuracy.
- Iterate the above steps \(m\) times and average the predictive accuracy of each iteration as the mean predictive accuracy (MPA).

#### End

As an iterative algorithm, a single ACC is calculated in one iteration and the MPA is the average of all ACCs. The number of iteration \(m\) need to be chosen as relatively large in order to provide a roughly unbiased idea of the prediction ability of the model.

However, when a binary classification is performed, the simple random sampling is not reasonable since it may occur that a specific class dominates the sample so that the model cannot be fitted well, thus some unnecessary biasness be created during the sampling procedure. In order to remedy this problem, the hold-out method can be modified to a stratified sampling procedure should always be used. It follows this process below:
Chapter 3. *Statistical Classification*

**Begin**

- Perform random sampling within each class. That is, sample 70% of the observations from each class as training sets and preserve the rest.

- Combine the two training sets as a whole training set. Do the same to the test set. Thus, the total observations being used in the training test is still 70%.

- Fit the model from the training data and predict the test data, and compute the predictive accuracy.

- Iterate the above steps \( m \) times and average the predictive accuracy of each iteration as the mean predictive accuracy.

**End**

Finally, the mean predictive accuracy (MPA) can be expressed as

\[
MPA = \frac{\sum_{i=1}^{m} ACC_i}{m}
\]
Chapter 4

Implementation of Automatic Accent Recognition - Study and Results

This chapter implements the recognition techniques being reviewed in Chapter 2 and Chapter 3 through a designed study. Section 4.1 describes this study in detail. In Section 4.2, the accent recognition is performed on time domain. Section 4.3 intends to achieve the same goal on frequency domain.

4.1 The First Study

In order to implement the automatic accent recognition machine and to examine the prediction ability of the algorithm, an study was constructed and the signal data are collected in the study. The procedure of the study follows the steps below:

- Through an internet resource, 22 different voices are chosen, of which 11 are of American English accent and 11 are not. Of the non-American voices, there are 3 British English voices, 2 Spanish voices, 2 French voices, 2 Italian voices, and 2 German voices.

- Each voice is required to read 15 different multi-syllable English words, such as “approximation” and “beneficial”. These words were sampled randomly without replacement from a population of such words without replacement, which means that no words was assigned to two or more voices.
• A total of $15 \times 22 = 330$ soundtracks were recorded through some internal recording device with a sampling rate of 44,100 Hz.

At this stage, we would not want the signals to be contaminated by noise, so that we used the internet resource together with internal recording device. Thus, the soundtracks only contained pure signals with the synthesized voices. A demographic summary of the soundtracks is given in Table 4.1.

<table>
<thead>
<tr>
<th>Accent</th>
<th>Gender</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Female</td>
<td>Male</td>
</tr>
<tr>
<td>US</td>
<td>90</td>
<td>75</td>
</tr>
<tr>
<td>Non-US</td>
<td>90</td>
<td>75</td>
</tr>
<tr>
<td>Total</td>
<td>180</td>
<td>150</td>
</tr>
</tbody>
</table>

Notice that this study is balanced in terms of accent but imbalanced in terms of gender. Since this work only focuses on the recognition of different accents, we would ignore the gender of each voice at this point. Though each soundtrack only contain 1 single word and thus it is fairly short, with a sampling rate of 44,100 Hz, each one of the signal vectors contains more than 30,000 elements on time domain.

And based on the description above, the target, or response, of this classification problem is given by

$$y_i = \begin{cases} 
-1 & \text{if non-US,} \\
1 & \text{if US,} 
\end{cases}$$

which defines this problem as a binary classification in which we are interested in categorizing different voices into two distinct accent classes.

### 4.2 Analysis on Time Domain

Before the formal analysis, we would like to perform some brief preliminary analysis. Figure 4.1 is a graph containing the waveform of two signals of the same word “approximation”. The top panel is of British accent and the bottom panel American accent. Though it is of linguistics interest to dissect the waveforms further and to theorize some semantic features, we would not draw our attention on such analysis. Instead, we could readily conclude that it is infeasible to deduce the distinction of different accents solely based on the waveforms on time domain.

Further, it is also of interest to examine the dimensionality of the signals. Since each of the signals has its unique length, the dimensionality of the amplitude vectors would
naturally have a variation. Figure 4.2 is a histogram of lengths of the amplitude vectors.

The length of these vectors is centered at around 44050, with a standard deviation of
8955. From Figure 4.2, it seems that the mode length is between 45000 and 47500, but even at its shortest length, which is below 30000, the dimension is still considered fairly high.

4.2.1 Dimension Reduction

As was discussed before, we would consider implementing PCA as the method of dimension reduction on time domain, but as a matrix-based technique, PCA requires an \( n \times p \) feature matrix rather than \( n \) separate vectors with different lengths. Thus, we start by forming the data matrix \( \mathbf{X} \). The procedure is rather simple. Let \( p_i \) be the length of each vector, \( i = 1, 2, \ldots, n \), then the dimensionality of the matrix \( p \) is

\[
p = \max\{p_i\}, i = 1, 2, \ldots, n
\]

For all the shorter vectors, the signals are duplicated until all \( p \) dimensions are fitted, but since the ends of a signal usually damp down, the duplicated part is essentially filled with 0.

Then we perform PCA on this data matrix following the idea shown in Section 2.1. Figure 4.3 illustrates the pattern of the eigenvalues, shown on the left-hand-side Y-axis, and the cumulative variation being explained by the first \( q \) eigenvalues, shown on the right-hand-side Y-axis. Though it is obvious that the eigenvalues are decreasing while the

![Figure 4.3: A plot of the eigenvalues and the cumulative percentage](image)
cumulative variation increases, it is difficult to tell how many principal components being preserved would be optimal. We would discuss this question in the following subsection, but at this stage, we follow one convention and preserve 95% of the variation. Table 4.2 gives a portion of the eigenvalues and the corresponding cumulative variation.

Table 4.2: Number of Principal Components

<table>
<thead>
<tr>
<th># PC</th>
<th>Eigenvalue</th>
<th>Percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>206</td>
<td>75.3270</td>
<td>89.94</td>
</tr>
<tr>
<td>207</td>
<td>74.9464</td>
<td>90.08</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>249</td>
<td>52.0964</td>
<td>94.97</td>
</tr>
<tr>
<td>250</td>
<td>51.8996</td>
<td>95.07</td>
</tr>
</tbody>
</table>

From the table, if we are to preserve 95% of the total variation, the corresponding number of principal components to be kept is 250. Evidently, this number is significantly smaller than 67,072, and yet the transformed data matrix is still not of a satisfactory shape, considering the ratio between $n = 330$ and $p = 250$ is merely 1.32. Regardless, we are able to perform classification based on this matrix.

**4.2.2 Comparison of classifiers**

We start by training three types of classifiers, discriminant function, SVM, and k-NN, using the full dataset. For discriminant analysis, only linear discriminant analysis can be performed, since there is not enough information to estimate two covariance matrices if we are to use quadratic discriminant function. For SVM, we use three different kernels, namely, the linear kernel, the RBF kernel, and the 2nd order polynomial kernel. For k-NN, the number of neighbors $k$ is chosen to be 3 based on a preliminary analysis using cross-validation. The results of training accuracy and error are given in Table 4.3.

Table 4.3: Summary of Training performance on time domain

<table>
<thead>
<tr>
<th>Method</th>
<th>Error (%)</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>7.88</td>
<td>92.12</td>
</tr>
<tr>
<td>SVM-L</td>
<td>0.31</td>
<td>99.69</td>
</tr>
<tr>
<td>SVM-RBF</td>
<td>4.55</td>
<td>95.45</td>
</tr>
<tr>
<td>SVM-P</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>k-NN</td>
<td>24.85</td>
<td>75.15</td>
</tr>
</tbody>
</table>

Apart from k-NN, all other classifiers seem to perform satisfying accuracy. For some classifiers, such as SVM with linear kernel and polynomial kernel, the accuracy seems to be even unrealistic. Thus, it is necessary to examine the test accuracy. Following
the procedure stated in Section 3.2, we perform the hold-out algorithm with stratified sampling. The number of iterations $m$ is set to be a moderate 500. The pattern of prediction accuracy is shown in the Figure 4.4. Each one of the boxes in the plot represents 500 estimated prediction accuracies.

![Figure 4.4: A comparison of test error on time domain](image)

It is obvious that none of the classifiers has convincing prediction ability. Regardless, it seems that SVM with RBF kernel is slightly better than the other classifiers. Further, comparing the training accuracies in Table 4.3 and the test accuracy, we can readily notice the discrepancy, which is likely caused by the overfitting issue. Also, for k-NN algorithm, most values of test accuracy are exactly 0.5. This is because of the degeneration of the algorithm. When an algorithm like this completely fails to perform, the two classes would collapse into a single class. And since the design of this study is balanced, exactly one half of the data would be falsely classified into the other category to which they should not belong.

### 4.2.3 How many PC’s to keep?

Back to the question in Section 4.2.1, we are still not certain how many principal components are optimal. This is a very serious question in PCA and yet the answer usually lies in the empirical stage rather than in the theoretical stage. For this specific dataset, it is difficult to know how many PC’s to keep directly from Figure 4.3, as is usually
done. We find an alternative method with the help of cross-validation. Assuming that we want to preserve at least 50% of the variation, we can plot the empirical relationship between the percentage of total variation and the mean prediction accuracy. Figure 4.5 is one such plot based on the SVM-RBF classifier.

![Figure 4.5: The cumulative percent variation vs. mean prediction accuracy](image)

From the plot, we can readily notice that preserving 95% of the variation is actually an inferior choice. Instead, it seems that MPA peaks when only around 60% or 70% of the total variation is kept. However, the peak MPA is still merely around 59%.

### 4.3 Analysis on frequency domain

Again, we start by some informal analysis. Figure 4.6 gives the corresponding comparison of spectrogram as Figure 4.1. And still we are not able to deduce the difference between the accents of the two groups of people although we are given significantly more information than in the waveforms, regardless of the potential meaning in the spectrogram.

#### 4.3.1 Predictive performance

As stated in Section 2.2, the idea of the analysis on frequency domain is to transform the signals onto frequency domain using Fourier transform and extract the MFCC features
through cepstral analysis. In detail, we examine the prediction ability of the classifiers with different numbers of MFCCs, varying from as small as 12 to as large as 39. The number of filters in the filter bank is chosen to be 40 in order to extract rich information from the signal. In terms of classification, we apply linear discriminant function, quadratic discriminant function, SVM with linear, RBF, and 2nd order polynomial kernels, and k-NN. Table 4.4 provides the performance results on frequency domain. In each cell, the first value represents the training accuracy, the second value (in italic) the mean prediction accuracy, and the third value (in parenthesis) the standard deviation of the prediction accuracy.

Notice that although the training accuracy still exhibits overfitting, the prediction accuracy improves tremendously comparing to the performance on time domain. A corresponding plot of MPA values in Table 4.4 is given below. Notice that the performances of LDA and SVM with linear kernel are close to each other and are both inferior than the other classifiers. K-NN demonstrates a better prediction ability, regardless of the number of MFCCs being used. Also, it is of interest to see that there is a relatively big improvement from 12 MFCCs being used, which simply indicates $p = 12$, to $p = 26$, and yet this improvement slows down from $p = 26$ to $p = 39$. For some classifiers, the accuracy even drops down slightly from $p = 33$ to $p = 39$.

It is also of interest to see how each classifier performs relatively to the others. Figure 4.8 modifies Figure 4.7 by subtracting the average prediction accuracy at each level of
Table 4.4: The predictive performance on frequency domain of the first study

<table>
<thead>
<tr>
<th># MFCCs</th>
<th>LDA</th>
<th>QDA</th>
<th>SVM-L</th>
<th>SVM-RBF</th>
<th>SVM-P</th>
<th>k-NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>0.7606</td>
<td>0.8394</td>
<td>0.7758</td>
<td>0.9000</td>
<td>0.9788</td>
<td>0.9424</td>
</tr>
<tr>
<td></td>
<td>(0.0362)</td>
<td>(0.0329)</td>
<td>(0.0351)</td>
<td>(0.0374)</td>
<td>(0.0364)</td>
<td>(0.0306)</td>
</tr>
<tr>
<td>19</td>
<td>0.7818</td>
<td>0.9061</td>
<td>0.8152</td>
<td>0.9485</td>
<td>1.0000</td>
<td>0.9667</td>
</tr>
<tr>
<td></td>
<td>(0.0345)</td>
<td>(0.0298)</td>
<td>(0.0336)</td>
<td>(0.0356)</td>
<td>(0.0274)</td>
<td>(0.0262)</td>
</tr>
<tr>
<td>26</td>
<td>0.8636</td>
<td>0.9697</td>
<td>0.8667</td>
<td>0.9758</td>
<td>1.0000</td>
<td>0.9758</td>
</tr>
<tr>
<td></td>
<td>(0.0322)</td>
<td>(0.0262)</td>
<td>(0.0336)</td>
<td>(0.0356)</td>
<td>(0.0272)</td>
<td>(0.0217)</td>
</tr>
<tr>
<td>33</td>
<td>0.8970</td>
<td>0.9879</td>
<td>0.9212</td>
<td>0.9848</td>
<td>1.0000</td>
<td>0.9909</td>
</tr>
<tr>
<td></td>
<td>(0.0314)</td>
<td>(0.0183)</td>
<td>(0.0333)</td>
<td>(0.0248)</td>
<td>(0.0205)</td>
<td>(0.0185)</td>
</tr>
<tr>
<td>39</td>
<td>0.8970</td>
<td>0.9879</td>
<td>0.9152</td>
<td>0.9758</td>
<td>1.0000</td>
<td>0.9909</td>
</tr>
<tr>
<td></td>
<td>(0.0332)</td>
<td>(0.0219)</td>
<td>(0.0326)</td>
<td>(0.0247)</td>
<td>(0.0216)</td>
<td>(0.0178)</td>
</tr>
</tbody>
</table>

Figure 4.7: A comparison of MPA on frequency domain
number of MFCCs and illustrates the relative relationship of the classifiers. While the

algorithm of k-NN is considered as the best algorithm in this case, SVM with linear kernel
and LDA are constantly below average performance, which is somewhat reasonable since
both classifiers try to accomplish linear classification.

### 4.3.2 Computational performance

For algorithms involving intense computation, it is always of interest to compare the
computational performance of different classifiers in terms of time being used to ac-
complish the computation. Since the automatic accent recognition can be split into
the process of feature extraction and the process of pattern recognition, the analysis
of computational performance can also be split in the same manner. At the stage of
feature extraction, the computational time is mainly the time to accomplish fast Fourier
transform and computation of MFCCs. Figure 4.9 gives an intuitive idea of the relation-
ship between the computational time for feature extraction with unit of seconds and the
number of MFCCs being extracted. Though one can argue that there seems to be an
upward trend as the number of MFCCs increases, it is not significant enough to have an
impact on the total computational time. Also, since traditionally the maximum number
of MFCCs being extracted is around 40, this variation of computational time is actually
trivial.
At the stage of pattern recognition, the question in terms of computation is: which classifier accomplishes the task with the least time? Figure 4.10 illustrates one such comparison. There are at least the following points need to be drawn attention to:
• It is not a surprise to see that the algorithm of k-NN uses significantly less time than the other algorithms. This is to say that the algorithm takes great advantage of its non-parametric methodology. Without estimating any parameters, k-NN saves much more time than other classifiers.

• At the stage of pattern recognition, the number of MFCCs does affect the computational time. This effect comes from the different dimensionality of the input matrix.

• For discriminant analysis, QDA, as a more complex algorithm, does not necessarily take more computational time than LDA. In fact, as the dimensionality increases, QDA becomes more efficient than LDA.

• For SVM, the RBF kernel uses more time to achieve the classification than the other two kernels.

An alternative method of measuring the computational time is to compute the relative computational time being used in terms of “k-NN time”, the computational time of k-NN algorithm. One benefit of this calculation of relative time is that computational time varies from computer to computer, thus the values of absolute computational time are somewhat meaningless. Figure 4.11 illustrates the idea of relative computational time. Since the relative time for k-NN is always 1, it is trivial to plot it on the graph.

![Figure 4.11: A comparison of relative computational time](image)

Though it seems that the gap between k-NN and other classifiers becomes closer as the
dimensionality increases, the second fastest algorithm is still around 6 to 8 times of the “k-NN time”.

4.4 Further Discussion

Based on the analysis above, we shall at least conclude the following:

- On time domain, the classification exhibits severe overfitting issue and it is falsely optimistic. That is, the training error is misleadingly low and sometimes it even approaches 0, while the test error degrades to values around 55%.

- On frequency domain, the prediction accuracy is significantly better than on time domain based on the MFCC feature input. When the number of MFCCs is chosen to be above 26, the prediction accuracies of QDA, SVM with RBF and 2nd order polynomial kernel, and k-NN are all above 90%.

- Based on the performance of LDA and SVM with linear kernel, the data is not linearly separable, on either time domain or frequency domain. For the k-NN classifier, it faces degeneration problems on time domain while becomes the best classifier on frequency domain.

- Computationally, the number of MFCCs would not significantly affect the computational time at the stage of feature extraction, but would likely slow down the computational time when classification is performed. K-NN requires least time in performing the computation and the second best classifier in terms of computation uses 10 times as much time as k-NN.
Chapter 5

Implementation of Automatic Accent Recognition - Effect of Noise

This chapter continues the analysis that has done in Chapter 4. Specifically, this chapter examines the effect of noise on the implementations of MFCCs. Section 5.1 is a continuation of the first study, in which the signals are artificially contaminated and the prediction results are performed. Section 5.2 performs a different, real-life study in which the signals were collected under a non-ideal condition. The detail of the study is described and the results of the analysis are performed.

5.1 Continuation of the first study

5.1.1 Perturbing signal with noise

Continuing the analysis in the previous chapter, we would like to examine the performance of the accent recognition machine when there is an evident amount of noise within the sound. Using the signals in the same study, we are able to acquire the noisy sound by injecting some well-designed noise into the pure signals.

One of such noise is the autoregressive model. Autoregressive model is a type of time-series model that specifies the output has a linear dependence only to its own previous values. In general, a $p$th-order autoregressive model is defined as

$$X_t = \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t,$$  \hspace{1cm} (5.1)
where $\phi_i$’s are the parameters of the model and usually takes on values between $-1$ and $1$ in order to keep the series stationary. The error term $\epsilon_t$ is a white noise, which is a sequence of uncorrelated variables with 0 mean and finite variance $\sigma^2$ that controls how much randomness the process exhibits at each time period $t$. The simplest autoregressive model is an AR(0) model, in which the output at time $t$ is the pure white noise. And if we are to assign the error term a Gaussian distribution, Equation 5.1 can be modified to

$$X_t = \epsilon_t, \epsilon_t \sim N(0, \sigma^2).$$

In signal processing, the magnitude of the standard deviation $\sigma$ controls the amplitude of the noise, that is, how strong the noise is. Figure 5.1 compares signals with different amplitude of AR(0) noise. The top panel is a signal without noise, the same one as the bottom panel in Figure 4.1. The middle panel is a mixture of this signal with an AR(0) noise of $\sigma = 0.01$, and the bottom panel of $\sigma = 0.03$. Notice that as the amplitude of the noise increases, the signal is shadowed and gradually the noise dominates the whole soundtrack.

### 5.1.2 Predictive performance on noisy data

Table 5.1 gives a comparison between the training accuracy and mean prediction accuracy of different classifiers with noisy data. The number of MFCCs is 26. Various values of $\sigma$ are considered. In each cell, the first value represents the training accuracy,
the second value (in italic) represents the mean prediction accuracy and the third value (in parenthesis) gives the standard deviation of the prediction accuracies. Though the

<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>LDA</th>
<th>QDA</th>
<th>SVM-L</th>
<th>SVM-RBF</th>
<th>SVM-P</th>
<th>$k$-NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.8636</td>
<td>0.9697</td>
<td>0.8667</td>
<td>0.9818</td>
<td>1.0000</td>
<td>0.9758</td>
</tr>
<tr>
<td></td>
<td>(0.0322)</td>
<td>(0.0241)</td>
<td>(0.0315)</td>
<td>(0.0304)</td>
<td>(0.0236)</td>
<td>(0.0204)</td>
</tr>
<tr>
<td>0.001</td>
<td>0.8182</td>
<td>0.8848</td>
<td>0.8364</td>
<td>0.9606</td>
<td>1.0000</td>
<td>0.9485</td>
</tr>
<tr>
<td></td>
<td>(0.0344)</td>
<td>(0.0314)</td>
<td>(0.0323)</td>
<td>(0.0260)</td>
<td>(0.0267)</td>
<td>(0.0270)</td>
</tr>
<tr>
<td>0.005</td>
<td>0.8152</td>
<td>0.8636</td>
<td>0.8212</td>
<td>0.9636</td>
<td>1.0000</td>
<td>0.9455</td>
</tr>
<tr>
<td></td>
<td>(0.0348)</td>
<td>(0.0314)</td>
<td>(0.0323)</td>
<td>(0.0260)</td>
<td>(0.0267)</td>
<td>(0.0270)</td>
</tr>
<tr>
<td>0.010</td>
<td>0.8030</td>
<td>0.8879</td>
<td>0.8212</td>
<td>0.9394</td>
<td>1.0000</td>
<td>0.9303</td>
</tr>
<tr>
<td></td>
<td>(0.0360)</td>
<td>(0.0357)</td>
<td>(0.0340)</td>
<td>(0.0342)</td>
<td>(0.0312)</td>
<td>(0.0343)</td>
</tr>
<tr>
<td>0.015</td>
<td>0.8121</td>
<td>0.9000</td>
<td>0.8121</td>
<td>0.9364</td>
<td>1.0000</td>
<td>0.9061</td>
</tr>
<tr>
<td></td>
<td>(0.0332)</td>
<td>(0.0342)</td>
<td>(0.0361)</td>
<td>(0.0362)</td>
<td>(0.0364)</td>
<td>(0.0346)</td>
</tr>
</tbody>
</table>

increment of noise does have an impact on the classifiers in terms of training accuracy, it is of important that such impact is not as strong as the one in terms of prediction accuracy. That is, when the noise has relatively large amplitude, the training accuracy exhibits strongly the feature of false optimism. This can be easily demonstrated by the classifier of SVM with polynomial kernel. All training accuracies are exactly equal to 1 regardless of how much noise is contained in the sound, but the mean prediction accuracy drops from 93% to 78% as $\sigma$ increases. Figure 5.2 is a corresponding plot of the values of MPA for all classifiers.

Based on Table 5.1 and Figure 5.2, it seems that the prediction performance would decrease as the noise in the sound gets stronger. Also, although the MPA values decrease for all classifiers, it is of interest to see that classifiers like LDA and SVM with the linear kernel do not degrade as much as other classifiers.

5.2 The second study

5.2.1 Description of the study

Throughout the above analysis, we have been using the signals collected from internet with internal recording to intentionally limit the variation of signals and the impact of noise effect, and yet such effect is inevitable in real-life researches. A second study was
The study was designed like this: Each participant of the study was required to read five phrases without any preparation and their voices were recorded with a sampling rate of 44100 Hz. As most participants use more than 7 seconds to finish reading one phrase, the average number of elements contained in each of the amplitude vectors is readily beyond 300,000. The five phrases they were asked to read are given below:

- Is it ever possible for anyone learning a new language to completely erase the traces of their childhood dialect? That is the question!

- We all came here to learn, and to grow, and along the way we make great friends, and ultimately we find a wonderful job! That’s exciting!

- The weather in this city is not always agreeable, but I love this place! I love this city and its people. It’s amazing!

- Although Mathematics doesn’t look beautiful to the untrained person, those who study it love it and find in it exquisite beauty!

- Humanity as a whole is at the threshold of a monumental shift in consciousness! You can see it everywhere! It is breathtaking!
In total, we collected the voices from 117 participants, 60 of whom have a native language of American English, and 57 do not.

### 5.2.2 Analysis on frequency domain

We start by performing a brief informal analysis, mainly by plotting the waveform and spectrogram of one of the signals. Figure 5.3 gives one such plot from a person of American accent reading phrase 1. Simply by checking the ends of the signal, it is obvious that it contains significant amount of noise.

In the formal analysis on frequency domain, we maintain that the 5 phrases are analyzed separately for computational convenience rather than being connected to one signal. The number of MFCCs to be preserved is kept as 22 based on some preliminary analyses. Table 5.2 gives the comparison of training accuracy and prediction accuracy for all classifiers given different sentences. Still, the first value in each cell represents the training accuracy and the second value the mean prediction accuracy and the value in parenthesis the standard deviation of prediction accuracies. While the training accuracy remains fairly optimistic, the prediction accuracy quickly deteriorates to 60% or below.
Table 5.2: A comparison of predictive performance in the second study

<table>
<thead>
<tr>
<th>Phrase</th>
<th>LDA</th>
<th>QDA</th>
<th>SVM-L</th>
<th>SVM-RBF</th>
<th>SVM-P</th>
<th>k-NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.7350</td>
<td>0.9402</td>
<td>0.7607</td>
<td>0.8034</td>
<td>1.0000</td>
<td>0.7521</td>
</tr>
<tr>
<td></td>
<td>(0.0729)</td>
<td>(0.0709)</td>
<td>(0.0656)</td>
<td>(0.0638)</td>
<td>(0.0719)</td>
<td>(0.0681)</td>
</tr>
<tr>
<td>2</td>
<td>0.7607</td>
<td>0.9402</td>
<td>0.7778</td>
<td>0.8120</td>
<td>1.0000</td>
<td>0.8889</td>
</tr>
<tr>
<td></td>
<td>(0.0698)</td>
<td>(0.0764)</td>
<td>(0.0732)</td>
<td>(0.0627)</td>
<td>(0.0720)</td>
<td>(0.0700)</td>
</tr>
<tr>
<td>3</td>
<td>0.7863</td>
<td>0.9573</td>
<td>0.7607</td>
<td>0.7863</td>
<td>1.0000</td>
<td>0.7350</td>
</tr>
<tr>
<td></td>
<td>(0.0700)</td>
<td>(0.0715)</td>
<td>(0.0693)</td>
<td>(0.0648)</td>
<td>(0.0731)</td>
<td>(0.0683)</td>
</tr>
<tr>
<td>4</td>
<td>0.7179</td>
<td>0.9402</td>
<td>0.7521</td>
<td>0.8291</td>
<td>1.0000</td>
<td>0.8034</td>
</tr>
<tr>
<td></td>
<td>(0.0691)</td>
<td>(0.0670)</td>
<td>(0.0725)</td>
<td>(0.0618)</td>
<td>(0.0701)</td>
<td>(0.0715)</td>
</tr>
<tr>
<td>5</td>
<td>0.7606</td>
<td>0.8974</td>
<td>0.7949</td>
<td>0.7863</td>
<td>0.9915</td>
<td>0.7439</td>
</tr>
<tr>
<td></td>
<td>(0.0750)</td>
<td>(0.0726)</td>
<td>(0.0726)</td>
<td>(0.0611)</td>
<td>(0.0682)</td>
<td>(0.0656)</td>
</tr>
</tbody>
</table>

5.3 Further Discussion

This chapter has demonstrated the significant impact of noise to the accent recognition problem. In real life, such noise may come from various kinds of sources. In the simplest situation, the noise may come from the surroundings, that is, the signal is not recorded in a soundproof environment. The use of recording device with poor quality may result in noise too, and in practice this type of noise may be more dominant than the environmental noise. Last, noise may also come from the transition from auditory signal to digital signal.

Similarly, the deterioration of the predictive performance may be resulted from different reasons, too. At least the following explanations can be found:

- At the stage of feature extraction, cepstral analysis and the use of MFCCs may extract unwanted information from the sound. In the computation of MFCCs, the combination of windowing and filter bank does yield rich information, but as the sound contain evident amount of noise, the features being extracted may as well represent the noise rather than the signal.

- Following this logic, the length of the signal itself also matters. Regularly, longer signals would provide more information than shorter ones. However, as the signals are contaminated by noise, the longer the signal is, the more noisy features are extracted from the sound.

- In terms of pattern recognition, it is natural that the predictive performance degrades when the signals become noisy. And yet some classifiers may be influenced
by noise more than the other classifiers. In other words, some classifiers are less robust.
Chapter 6

Conclusion

In this thesis, we have demonstrated the multiple techniques of feature extraction and pattern recognition and the performance of their implementation given various conditions of the data. We hereby conclude the results of our research by summarizing the analysis and providing some future work.

6.1 Performance of accent recognition machine

6.1.1 Time or frequency?

In Chapter 4, we have demonstrated that the recognition on time domain with PCA is very poor, where merely around 60% of the signals were correctly classified. However, this prediction accuracy can reach as high as 95% on frequency domain with cepstral analysis. Further, the analysis on time domain results in the overfitting issue that would cause false optimism if we are to compute only the training accuracy. This is in fact not a surprising result, since the information or rather feature is richer on frequency domain. In terms of dimension reduction, both PCA and cepstral analysis reduce the high dimensionality of the raw data drastically, but cepstral analysis reduces the dimensionality to a more satisfying result than PCA. Thus, frequency domain should be preferred.

6.1.2 Which classifiers to use?

Though we have shown that the algorithm of k-NN is the best in terms of both predictive performance and computational performance, the question of “which classifiers to use” is not as simple as it seems to be. Intuitively, there is an infinite number of
learning machines that can achieve the accent recognition task. Whether or not there is a potential classifier that can provide the best performance is always left unknown. Nevertheless, we can still obtain some idea of such classifier. First, we shall at least see that the features are not likely to be linearly separable. This can be proven by the failure of LDA and SVM with linear kernel. Second, SVM with some nonlinear kernel functions is able to yield decent prediction accuracy, but it takes much more time than k-NN, which most of the time can perform at least as decent as SVM. Third, k-NN provides significantly different results on time domain and on frequency domain. This is because k-NN can be easily affected by the noise contained in the data.

6.1.3 When does it fail?

It is also of interest to see that the combination of MFCC features and some binary classifiers would deteriorate quickly when the signals are contaminated by noise. The possible reasons lie both on the stage of feature extraction and on pattern recognition. Though it is natural that the prediction accuracy would decrease as the data becomes noisy, such decay seems to be somewhat amplified by the technique of MFCC. In other words, the rich features being extracted from the signal do not necessarily represent the signal itself, but represent the noise instead. Such decline may also come from pattern recognition, where some classifiers are not robust to noise.

6.2 Future work

We have shown that the combination of MFCCs and SVM or k-NN is a powerful technique in performing automatic accent recognition. And yet since the area of cepstral analysis is fairly new, there is great potential to improve our work in the future. Below is a list of such potential future works:

- In the computation of MFCCs, we chose to use the mean vector of a MFCC matrix to represent the feature of each speech. That is to say, the MFCCs are averaged over time. We can also take the standard deviation into account. Or, in a more sophisticated way, the MFCCs can be modelled using Gaussian mixtures.

- In terms of the deteriorated predictive performance when the sound exhibits noise, it is of great interest and importance to implement a de-noising technique. Usually the useful information would be damaged at a certain level when de-noising is performed. Thus intuitively, a decent de-noising method should maximize the reduction of noise while minimize the loss of information.
• In SVM and k-NN, some domain-specific kernel functions or distance functions can be designed so that the classification can be directly performed without using Fourier analysis and the further cepstral analysis.

Though these future tasks may cover various branches of studies, they all share the same target, which is to further explore the automatic accent recognition and improve its predictive performance.
Appendix A

Selected R Code

A.1 R Code in Performing Analysis on Time Domain

# Loading packages
library(audio, tuneR, signal, MASS, kernlab, class)

# Reading in signals
filename <- dir()
ntrack <- length(filename)

Gen <- substr(filename, 4, 4)
Gen.ind <- as.factor(ifelse(Gen == "F", 1, 0))

Nat <- substr(filename, 1, 2)
Nat.ind <- as.factor(ifelse(Nat == "US", 1, 0))

ltrack <- NULL

for (i in 1:ntrack)
{
    track <- load.wave(filename[i])
    ltrack[i] <- length(track)
}

X <- matrix(0, ncol=max(ltrack), nrow=ntrack)
for (i in 1:ntrack) {
  track <- load.wave(filename[i])
  ltrack <- length(track)
  X[i,1:ltrack] <- track
}

########################################
##### PCA on dual space ######
########################################

D <- X%*%t(X)
E <- eigen(D)
lambda <- E$value
V <- E$vector

q <- min(which(cumsum(lambda)/sum(lambda)>0.95))
coly <- q + 1
Z <- V[,1:q]
ZY <- data.frame(Z,Nat.ind)

########################################
##### Training accuracy ######
########################################

lda.zy <- lda(Nat.ind~.,data=ZY)
pred.lda <- predict(lda.zy,data.frame(Z),type='response')$class
conf.lda <- table(Nat.ind,pred.lda)
acc.lda <- sum(diag(conf.lda))/sum(conf.lda)

svm.zy <- ksvm(Z,Nat.ind,kernel="vanilladot")
pred.svm <- predict(svm.zy,Z,type='response')
conf.svm <- table(Nat.ind,pred.svm)
acc.rbf <- sum(diag(conf.svm))/sum(conf.svm)

svm.zy <- ksvm(Z,Nat.ind,kernel=polydot(2))
pred.svm <- predict(svm.zy,Z,type='response')
conf.svm <- table(Nat.ind,pred.svm)
acc.poly <- sum(diag(conf.svm))/sum(conf.svm)
Appendix A. *R Code*

\[
\text{svm.zy} <- \text{ksvm}(Z, \text{Nat.ind}, \text{kernel}="\text{vanilladot}\")
\]
\[
\text{pred.svm} <- \text{predict}(\text{svm.zy}, Z, \text{type}='\text{response}')
\]
\[
\text{conf.svm} <- \text{table}(\text{Nat.ind}, \text{pred.svm})
\]
\[
\text{acc.ln} <- \text{sum}\left(\text{diag}\left(\text{conf.svm}\right)\right)/\text{sum}\left(\text{conf.svm}\right)
\]
\[
\text{knn.zy} <- \text{knn}(\text{train}=Z, \text{test}=Z, \text{cl}=\text{Nat.ind}, k=2)
\]
\[
\text{conf.knn} <- \text{table}(\text{Nat.ind}, \text{knn.zy})
\]
\[
\text{err.knn} <- \text{sum}\left(\text{diag}\left(\text{conf.knn}\right)\right)/\text{sum}\left(\text{conf.knn}\right)
\]

A.2 R Code in Performing Analysis on Frequency Domain

```
############################
##### Constructing MFCCs #####
############################

ncep <- 40
MFCC <- matrix(0,nrow=ntrack,ncol=(ncep-1))

for (i in 1:ntrack)
{
  track <- load.wave(filename[i])
  track <- Wave(left = as.numeric(track), samp.rate=44100, bit=16)
  mfcc <- melfcc(track, numcep=ncep, nbands=40)[,2:ncep]
  MFCC[i,] <- apply(mfcc,2,mean)
}

############################
##### Training Accuracy #####
############################

XY <- as.data.frame(cbind(MFCC,Nat.ind))

lda.tr <- lda(Nat.ind~., data=XY)
lda.pred <- predict(lda.tr, as.data.frame(MFCC), type='response')$class
lda.conf <- table(lda.pred,Nat.ind)
lda.acc <- sum(diag(lda.conf))/sum(lda.conf)
qda.tr <- qda(Nat.ind~.,data=XY)
qda.pred <- predict(qda.tr,as.data.frame(MFCC),type='response')$class
qda.conf <- table(qda.pred,Nat.ind)
qda.acc <- sum(diag(qda.conf))/sum(qda.conf)

lin.tr <- ksvm(MFCC,Nat.ind,kernel="vanilladot")
lin.pred <- predict(lin.tr,MFCC,type='response')
lin.conf <- table(lin.pred,Nat.ind)
lin.acc <- sum(diag(lin.conf))/sum(lin.conf)

rbf.tr <- ksvm(MFCC,Nat.ind,kernel="rbfdot")
rbf.pred <- predict(rbf.tr,MFCC,type='response')
rbf.conf <- table(rbf.pred,Nat.ind)
rbf.acc <- sum(diag(rbf.conf))/sum(rbf.conf)

ply.tr <- ksvm(MFCC,Nat.ind,kernel=polydot(2))
ply.pred <- predict(ply.tr,MFCC,type='response')
ply.conf <- table(ply.pred,Nat.ind)
ply.acc <- sum(diag(ply.conf))/sum(ply.conf)

knn.pred <- knn(train=MFCC,test=MFCC,cl=Nat.ind,k=3)
knn.conf <- table(knn.pred,Nat.ind)
knn.acc <- sum(diag(knn.conf))/sum(knn.conf)
Bibliography


