7-2015

Text-Independent Speaker Identification using Statistical Learning

Alli Ayoola Ojutiku
University of Arkansas, Fayetteville

Text-Independent Speaker Identification using Statistical Learning

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Electrical Engineering

by

Alli Ayoola Ojutiku
Obafemi Awolowo University
Bachelor of Science in Electronic and Electrical Engineering, 2008

July 2015
University of Arkansas

This thesis is approved for recommendation to the Graduate Council.

Dr. Jingxian Wu
Thesis Director

Dr. Jing Yang
Committee Member

Dr. Baohua Li
Committee Member

Abstract

The proliferation of voice-activated devices and systems and over-the-phone bank transactions has made our daily affairs much easier in recent times. The ease that these systems offer also call for a need for them to be fail-safe against impersonators. Due to the sensitive information that might be shred on these systems, it is imperative that security be an utmost concern during the development stages. Vital systems like these should incorporate a functionality of discriminating between the actual speaker and impersonators. That functionality is the focus of this thesis.

Several methods have been proposed to be used to achieve this system and some success has been recorded so far. However, due to the vital role this system has to play in securing critical information, efforts have been continually made to reduce the probability of error in the systems. Therefore, statistical learning methods or techniques are utilized in this thesis because they have proven to have high accuracy and efficiency in various other applications. The statistical methods used are Gaussian Mixture Models and Support Vector Machines. These methods have become the de facto techniques for designing speaker identification systems. The effectiveness of the support vector machine is dependent on the type of kernel used. Several kernels have been proposed for achieving better results and we also introduce a kernel in this thesis which will serve as an alternative to the already defined ones. Other factors including the number of components used in modeling the Gaussian Mixture Model (GMM) affect the performance of the system and these factors are used in this thesis and exciting results were obtained.

Acknowledgements

I would like to give gratitude to the Almighty Allah for making my program a success.

I would also like to offer my sincere gratitude to my advisor, Dr. Jingxian Wu, for his support during my master's program and especially during my research. The in-depth ideas and intuition he offered were invaluable and above all, his patience was top-notch.

The contribution of my committee members, Dr. Jing Yang and Dr. Baohua Li is also acknowledged, they made valuable suggestions towards the completion of this thesis.

I acknowledge the volume of work Dr. D. A. Reynolds has put into researching speaker recognition, his work is detailed and I was able to learn the concepts of this thesis from some of his papers. Some of the methods he proposed have become the standards in this field and I am glad this thesis was able to improve on some of his ideas. Thank you for the hard work.

I would also like to thank all the students in my research group. We had a very good semester in terms of our research and seminars.

Finally, my family and friends are really appreciated for their support, encouragement and prayers.

Table of Content

Chapter 1: Introduction……………………………………………………………………………………………………….1

1.1 Speech Systems…………………………………………………………………………………………………….1

1.2 Classification of Speaker Identification……………………………………………………………………..2

1.2.1 Text-dependent Speaker Identification…………………………………………………………………3

1.2.2 Text-Independent Speaker Identification……………………………………………………………..3

1.3 Motivation………..4

1.4 Literature Review……………………………………………………………………………………………………6

1.5 Thesis Overview…………………………………………………………………………………………………….7

1.6 Notation………8

Chapter 2 : Speaker feature Extraction…………………………………………………………………………10

2.1 Overview of Feature Extraction. ………………………………………………………………………...11

2.1.1 Spectro-temporal Features. ……………………………………………………………………………11

2.1.2 Prosodic Features………………………………………………………………………………………………12

2.1.3 High Level Features……………………………………………………………………………………………13

2.1.4 Short-term spectral features……………………………………………………………………………..13

2.1.5 Mel Frequency Cepstral Coefficient (MFCC)…………………………………………………………13

2.2 Description of MFCC Generation…………………………………………………………………………….14

2.2.1 Implementation steps………………………………………………………………………………………..14

2.3 Error Compensation Methods………………………………………………………………………………..21

Voice Activity Detection (VAD) ……………………………………………………………………………………21

Feature Normalization………………………………………………………………………………………………..21

Chapter 3: Feature Modeling……………………………………………………………………………………….23

3.1 Overview of Feature Modeling……………………………………………………………………………….23

3.1.1 Vector Quantization (VQ)….. ……………………………………………………………………………...23

3.1.2 Dynamic Time Warping (DTW)……………………………………………………………………………..23

3.1.3 Hidden Markov Model (HMM)……………………………………………………………………………...24

3.1.4 Gaussian Mixture Model (GMM)… ……………………………………………………………………….24

3.2 Gaussian Mixture Model……………………………………………………………………………………….24

Expectation Maximization (EM) Algorithm……26

3.2.1 Universal Background Model……27

3.2.2 Adapted Gaussian Mixture Model……28

Chapter 4: Speaker Identification……31

4.1 Overview of Speaker Identification/Detection Methods……31

4.1.1 Euclidean distance……31

4.1.2 Mahalanobis distance……32

4.1.3 Bhattacharya Distance……32

4.1.4 Kullback-Leibler Divergence Measure……33

4.1.5 Likelihood function……33

The likelihood ratio……35

4.2 Score normalization……36

Chapter 5: Statistical Learning and Support Vector Machines……38

5.1 Introduction to Statistical learning……38

5.1.1 Supervised Learning……39

5.1.2 Unsupervised Learning……40

5.2 Statistical Learning in speaker Identification Systems……40

5.2.1 Support Vector Machine……41

Non-Linear SVMs……44

5.2.2 Kernels……45

5.3 Speaker Identification System using SVMs……46

5.3.1 New Kernel……48

5.3.2 Randomization……49

5.3.3 SVM Training……50

5.3.4 SVM Testing……51

Chapter 6: Experimental Design and Results……53

6.1 Experiment Design……53

6.1.1 Speech Corpus……53

6.1.2 Front End Processing……53

6.1.3 GMM Adaptation……54

6.1.4 GMM Supervector……54

6.1.5 Support Vector Machine……55

6.2 Results……55

6.3 Discussion of Results……68

Chapter 7: Conclusion and Future Work……72

7.1 Conclusion……72

7.2 Future Work……73

References……74

List of figures

Figure 2.1 Components of a Speaker Identification System…………………………………………………………10

Figure 2.2 Mel Frequency Triangular Filterbank ……………………………………………………………………..19

Figure 4.1: Speaker Identification System based on Likelihood Ratio…………………………………………………36

Figure 5.1: Class separation in Support Vector Machines…………………………………………………………….41

Figure 5.2 Transformation of Data from Input to Feature Space………………………………………………………..44

Figure 6.1: Flow chart of result computation……………………………………………………………………………..56

Figure 6.2: DET curves for 32-speaker speaker identification system…………………………………………………..59

Figure 6.3: DET curves for 16-speaker speaker identification system…………………………………………………..60

Figure 6.4: DET curves for new kernel with polynomial order of 3 to 7………………………………………………….61

Figure 6.5: DET curves for new kernel with polynomial order of 3 to 5………………………………………………….62

Figure 6.6: DET curves for new kernel with polynomial order of 4 to 6………………………………………………….63

Figure 6.7: DET curves for new kernel with polynomial order of 5 to 7………………………………………………….64

Figure 6.8: DET curves for new kernel with polynomial order of 6 to 15………………………………………………..65

Figure 6.9: DET curves for new kernel with polynomial order of 6 to 9………………………………………………….66

Figure 6.10: DET curves for new kernel with polynomial orders of 9, 10 and 15…………………………………….67

Figure 6.11: Plot of speaker detection error probability as a function of polynomial order…………………68

Chapter 1: Introduction

Speaker identification is the determination of the identity of a speaker from his or her speech. It is sometimes referred to as speaker recognition or speaker verification. These terms are sometimes used to distinguish the specific functionality of the system. Speaker verification is used to refer to a system that confirms the claimed identity of a speaker. So, it is a one to one comparison between the test speech features and a model of the supposed speaker's speech. Meanwhile, speaker recognition can refer to a system that tries to discern the identity of a speaker out of a number of possible speakers. In both cases, the identity of the speaker is the essence of the operation.

Because humans easily discern the identity of a person if they are used to hearing the person speak, it shows that the human voice has features or characteristics that are peculiar to each individual. How to identify and extract these features and characteristics and feed them to a speaker identification system is therefore a concern. A few types of speech characteristics have been identified and they will be discussed subsequently in this thesis. Similar to how humans must have at least heard a voice once to be able to identify the speaker, the voices of the prospective speakers have to be enrolled in the system before an identification operation can be carried out.

1.1 Speech Systems

Voice features extracted from a person's speech have identifying capabilities and as such, have been utilized to implement different types of technologies with related but specific functionalities. The following are examples of such systems.

1. Gender Identification System

   Males and females are known to have different qualities of voices. While males are usually known to have deep sounding voices which correspond to having timbres and formants at low frequencies, females are known to possess voices with higher pitches when compare to the males. This difference, including some others such as the size of the vocal fold, the vocal tract and the manner of speaking can be used in Gender identification systems to establish the sex of a speaker.

2. Language Identification System

   Several approaches have been utilized in achieving an efficient language identification system. Most of the methods are based on using several statistical methods. The general idea is the creation of training models for several languages using features peculiar to them. The models are then compared to the test languages to confirm a match.

3. Speech Recognition System

   This application draws similarity to the speaker identification system but differs on what the output is. While a speaker identification system establishes "who" is speaking, the speech recognition system outputs "what" is being said. This system makes use of Hidden Markov Models (HMM) and Gaussian Mixture Models (GMM) among other techniques. The performances of the systems have risen overtime and they have numerous commercial applications. An example of this system is the SIRI application on apple devices. This application takes a speech from a user, detects what is being said and performs instructions based on results of the transcription. Such applications are also found in automobiles where uttering simple commands are used to execute simple operations without having to touch buttons.

1.2 Classification of Speaker Identification

Speaker identification systems can be classified based on several criteria. One such classification depends on the nature of the sample speech or text and another depends on the type of features extracted from the speech. Classification based on the sample text is reviewed below.

1.2.1 Text-dependent Speaker Identification:

This type of system identifies a speaker by using a pre-meditated word or pass-phrase. The system has several models of the same phrase from different speakers. A test speaker has to utter the same phrase or word in order to be identified. The test speech from that speaker is then modeled into a form similar to that stored in the system and the test model can be aligned with the stored models to establish which one has the closest match.

Another variant of this class is called the text-prompted Speaker identification system. This type has a set of phrases that are accepted by the system. So at the point of testing, the system requests that one of the accepted phrases be spoken and testing is done subsequently.

While these types of speaker identification systems are simple to design, they fall short in terms of security. An impersonator could practice how someone else talks if he has knowledge of the pass-phrase and this leaves the system vulnerable. An improvement over this is achieved in the prompted speech by having several pass-phrases the systems can select from. This increases security if there is a large pool of pass-phrases. This category still has some disadvantages because the cooperation of the user is needed to ensure the system works as expected. Therefore, in some applications where user cooperation is not desired, it will be impossible to utilize this class of speaker identification systems. The computational complexity of this system is low and is used in systems where security is not a major concern.

1.2.2 Text-Independent Speaker Identification:

This category of systems operates without regard to what exactly is being spoken. It attempts to efficiently detect the identity of a speaker by extracting speaker-dependent information from the test speech. This information is assumed to be available regardless of the words being spoken or the speed at which the words are uttered. The voice samples of all likely speakers need to be available to the system and they will serve as the training samples. Depending on the technique used, a model representing each speaker is generated and these will serve as the objects of comparison during testing. As opposed to the text-dependent type, security in this system is improved upon. Impersonation would be difficult because an impersonator might not be aware that a speaker identification process is in progress since user cooperation is not desired and there is not a specific word or a pool of words that are accepted. On the flip side, however, the computational complexity of this category is quite high. This might result into longer computation time and higher resource requirements.

There is therefore, a trade-off between performance and complexity. With the number of high speed chips in the market, computational time should not be a major concern as well as the needed resources. In order to utilize the advantages of the text-independent speaker identification system, this method will be the focus of this thesis and all subsequent references to speaker identification systems will henceforth be assumed to be text-independent. Below is a brief summary of the difference between both categories. [1]

Text-dependent	**Text-Independent**
Spoken text is known before hand by system or is prompted	Spoken text is random
User cooperation is required	User cooperation is not required
Knowledge of test speech improves performance	Higher computational complexity due to random nature of test speech.

Table 1.1: Differences between Text-dependent and Text-independent Speaker Identification Systems [1]

1.3 Motivation

The sophistication of gadgets and devices has grown exponentially in recent years with the incorporation of functionalities that were previously assumed impossible. A single device can perform video chatting, finger print recognition, video streaming, online banking and shopping etc. These functionalities have eased the way we do a good amount of things in our daily affairs. The downside however, is that with all these sophistication, we become more vulnerable to identity thefts and other forms of impersonations that might have very dire consequences.

In addition, due to the fast paced nature of the world today, men of the underworld have also developed innovative ways to outsmart people and wreak havoc. As a consequence, law enforcement needs to step up to the plate to ensure that lives and property of citizens are not threatened. Technology can be of help to curb this social problem.

An efficient speaker identification system can easily detect a fraud if an imposter is trying to perform an illegal over-the-phone financial transaction. This can be done without the knowledge of the imposter. An intercepted phone conversation about a crime can also be used to prevent the crime if the city or country for example, has a database of the voices of all residents. The speaker identification system would take the phone conversation, perform some front end operations and compare it to each of the voices in the database so that a match can be made and the crime prevented.

These few examples go to show the motivation for this work and a system like this has to be almost fail-safe due to the sensitivity of the issues that might be involved. More efficient ways of developing these systems are being researched to reduce the probability of error in these speaker identification systems. The following are a few applications of such systems.

1. Access control
2. Law enforcement
3. Speech data management
4. Transaction authentication
5. Personalization

1.4 Literature Review

Work on this thesis began with a review of previous work in this field. Reynolds D. A. gave an overview of a speaker identification system in his training document: Overview of automatic speaker recognition [1]. It took a quick look at the theoretical knowledge needed to understand how a speaker identification system works. From features extraction to feature modeling, he discussed the basic components of a speaker identification system and how testing is carried out to obtain results. A discussion of the speech production process was contained in there including the characteristics of the desired voice features for a speaker identification system. The classes of models for voice features were discussed which included the generative and discriminative types. The training document offered a simplistic overview to what a speaker identification system is about and anyone interested in working on these systems can use the document as a good foundation. Another such introductory material is [2], it does a good job in giving small details about the concepts needed to understand a speaker identification system. [3] and [4] also provide a good starting point to learning about speaker identification systems.

In [5] and [6], the Gaussian Mixture Model method of modeling was explored in detail and the performance was measured with respect to several parameters. The performance was observed as a function of the number of components used to model the GMM, the length of the speech signal used for training and the length of the test speech.

[7] and [8] took a different approach to the GMM method, it involved adapting speaker models to a universal background model which was formed from the voice samples of all possible speakers. Results in [7] showed that this method produced a better recognition performance compared to the system where each speaker model is created independently.

A more modern method of speaker identification was presented in [9]. Support vector machines which use a discriminative rather that generative method of classification were used to model the training samples. The support vector machine approach is more exciting lately because of its effectiveness and how much they are now used in several top-notch applications.

Support vector machines are a 2 class classification method which separates data into 2 classes by creating a hyperplane between then. It has a training algorithm that seeks to minimize an error function with respect to 2 constraining expressions. [10] and [11] are good guides to learning support vector machines. Support vector machines are used in conjunction with GMM supervectors in [9] to obtain high performance results. A couple of kernels were proposed and these kernels utilized the GMM supervectors.

A kernel is simply a function that returns that value of the dot product between 2 signals in a space of higher dimensionality. It is the most important part of the support vector machine and there are a set of conditions that help to determine if a function can serve as a kernel function. The conditions are referred to as the Mercer conditions.

The next section will describe this thesis work in general by giving an overview of the work done and the techniques used. It seeks to serve as a summary to the rest of this thesis. A quick perusal of the section will offer an insight into what is to be expected in the remainder of this work.

1.5 Thesis overview

The work of this thesis covers the utilization of some elements of statistical learning methods to design a text-independent speaker identification system. The statistical learning elements used are the well-known techniques in the field of speech and speaker recognition. GMM and Support vector machines are those statistical learning methods that would be used to implement the system.

After this introductory chapter is chapters 2, 3 and 4 which explore the speaker identification system in an in-depth fashion. They review the details of how speaker identification is achieved. Chapter 2 goes from how the feature vectors are extracted from voice samples to the types of features used and how all front end processing of voice samples are obtained. Chapter 3 discusses the modeling techniques available for speaker identification experiments. It gives an insight into the GMM and how it is obtained including the adapted GMM. Then, feature testing is discussed in chapter 4. Chapter 5 looks into statistical learning and support vector machines and also addresses the theory of support vector machine kernels.

Chapter 6 discusses the experimental design used for this research from the front end processing and GMM supervectors to support vector machines and kernel functions. Also included in chapter 6 is the detail of the speech database used. The results obtained were also included in chapter 6 including discussions of the results. The last chapter concludes the thesis by summarizing what was learned from this work and taking a peep into the future on what is expected in the next few years. It is worth a mention that while there are systems that might be capable of detecting the identities of multiple speakers in a single audio recording, the scope of this thesis is limited to single speaker identification. Therefore, subsequent references to speaker identification system in this thesis are assumed to be the single-speaker identification system.

The experimental simulations for this thesis were carried out in Matlab.

1.6 Notation

Lowercase alphabets represent the time domain and uppercase non-bold face alphabets represent the frequency domain. Matrices are denoted by boldface alphabets, vectors by a letter with an arrow above and a scalar is lowercase and carries no arrow. The speech signal will be denoted by $\vec{s} = [s(0), s(1), \dots, s(NT-1)]^T$ where NT is the total number of samples in the entire speech utterance and T is the matrix transpose, and once it is broken down into frames, it is denoted by $\vec{s}_i = [s(0), s(1), \dots, s(N-1)]^T$. Where N = 882 is the number of samples per frame, i represents the index of a frame. When the Fourier transform is taken, the frequency domain signal is represented by $\vec{S}_i = [S_i(0), S_i(1), \dots, S_i(K-1)]^T$ where i is the index of the frame, K is the FFT length and T represents the matrix transpose. With i still retains the meaning it had in the time domain and k ranges between 1 and the size of the FFT = 512. The output of the feature extraction stage (the feature vectors) will be denoted by the letter $\vec{x}_i = [x(0), x(1), \dots, x(Nc-1)]^T$ where i is the index of the frame and *Nc =12* is the number of cepstral coefficients per frame and T is the matrix transpose.

Chapter 2: Speech Feature Extraction

Having gone over the general idea of a speaker identification system in the introductory chapter, this chapter will elaborate on the theory and methodology of the speech feature extraction. Like with every abstract concept, the key to understanding the methodology is getting a clear picture of the theory behind the various methods. This chapter intends to do just that.

The speaker identification system consists of various steps and techniques and they can be broadly categorized into 3 phases. The first phase is known as the feature extraction phase which involves extracting the speaker dependent speech information of each speaker. The second phase is referred to as training or enrollment and it consists of the techniques that contribute to assisting the system learn the characteristics of a sample data before a recognition or detection operation can be carried out. The third phase is called testing or detection and it covers the methods used to classify test samples. These phases will be covered in this chapter, chapter 3 and chapter 4 respectively. Figure 2.1 shows a diagram which describes the 3 phases.

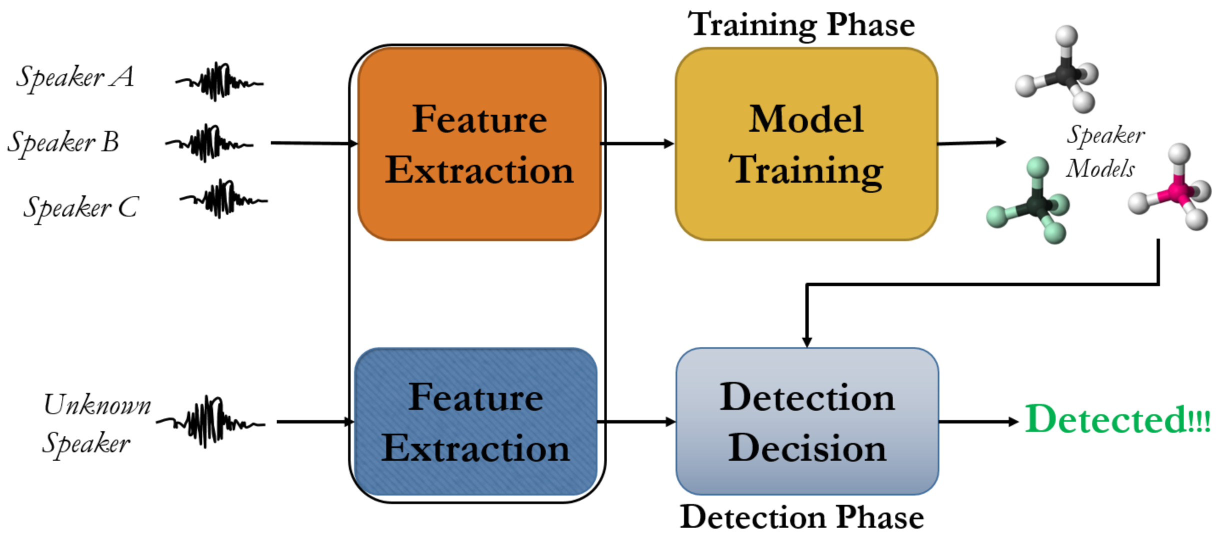

Figure 2.1 Components of a Speaker Identification System

2.1 Overview of Feature Extraction

By auditory perception, humans are able to distinguish the identity of speakers. It therefore means the human speech contains information that is specific to each individual. The ability to access this information will ease to a large extent, our speaker identification task. Feature extraction is therefore, the extraction of speaker dependent features from a speech signal. The speech signal consists of several types of features and these features differ in the sort of information they carry. Some features describe the structure of the vocal tract while some other gives information about the speaking habits such as stress patterns, accent, frequently used words etc. what exact type of feature to use for an operation will be the natural question. And for this, there is not a hard and fast rule. A speech feature to be utilized for an operation should however have a few characteristics for it to be fit for the speaker identification process [2]. A valid speech signal feature should have [2]

1. High inter-speaker feature variability and low intra-speaker feature variability
2. High immunity to impersonation
3. Low variability on changing environmental and speaking conditions
4. High immunity to changing well-being of the speaker
5. High frequency of occurrence in speech and should come naturally
6. A fair ease of measurement

It is worth a mention that no single type of feature possesses all these desirable attributes. A tradeoff therefore has to be made to select the best for a particular operation.

The following are a few types of speech features.

2.1.1 Spectro-temporal Features

These types of features convey information about formal transitions and energy modulation within a speech signal [2]. It has proven to be a good way of representing speaker information. The most common form of its implementation is through first and second order time derivations of the spectral features. These time derivatives are referred to as delta and double delta coefficients respectively or differential and acceleration coefficients respectively. These derivatives are appended to the short term spectral feature vectors so the resulting feature vector has a longer dimension than short term features. Using the spectro-temporal features this way is known to improve the speaker identification systems' performances. [2]

The delta coefficient is computed with the formula below.

$$d_t(c) = \frac{\sum_{p=1}^{P} p(x_t(c+p) - x_t(c-p))}{2\sum_{p=1}^{P} p^2} \qquad 0 \leq c \leq Nc \qquad (1)$$ [12]

$x_t(c)\ is\ the\ short - term\ spectral\ coefficient\ (to\ be\ discussed\ later)$. Nc is the number of cepstrum coefficient and c is the index of the cepstrum coefficients. t is the frame index.

A typical value for P is 2 and the double-delta uses the same formula except that x_t is replaced with $d_t(c)$. [12]

2.1.2 Prosodic Features

Prosody refers to the components of speech that are non-segmental. It has to do with the properties of other larger units of speech like intonation, stress, rhythm, tone etc. Unlike short term features, prosodic features span over longer segments of the speech signal and it contains information about a person's speaking style, emotions, and form of utterance [2]. The fundamental frequency is the most important prosodic feature and like the spectro-temporal feature, it is sometimes appended to the short term spectral features in order to achieve higher and better performance. [2]

2.1.3 High Level Features

This feature type draws information about the type of words that are frequented in a speaker's conversations. This information is used to characterize the speaker by converting the speech signal into a sequence of tokens. The arrangement of the tokens within a sequence is used to represent each speaker. These tokens can be words, syllables, phonemes etc. [2]

2.1.4 Short-term spectral features

The speech signal is continuously changing due to the movement of the components of the vocal tract. When the speech signal is then broken down into 10-30ms frames, it is assumed that these frames are constant in time and convey a single piece of acoustic information. The spectral envelopes of these frames usually obtained through Discrete Fourier Transform (DFT) are then used to represent the information conveyed by the signal. These envelopes convey information about the anatomy and resonant properties of the vocal tract. The short-term spectral features have been shown to be very effective in modeling speaker-independent speech characteristics. And they are relatively easy to extract. This is the feature type used in this thesis.

These features have been used in several operations and the choice for a particular process would depend on the designer's preferences. The short term spectral features were used in this thesis because of the ease of extraction and their limited variability over varying speaker moods. The next subsection discussed a specific type of short term feature used in this thesis.

2.1.5 Mel Frequency Cepstral Coefficient (MFCC)

The MFCC is the most used type of short term spectral feature in speaker identification systems due to its accurate estimation of the spectral envelope. An accurate estimation of the spectral envelope translates to an accurate modeling of the vocal tract which in turn infers that the phonemes produced are well and precisely estimated. The Mel frequency scale is used to create a representation of the human auditory system. This scale is linear below 1000Hz and non-linear above 1000Hz. It is the representation of the short term power spectrum of a speech signal using the Discrete Cosine Transform (DCT). The DCT helps to de-correlate the energies of the spectrum and this allows the estimation of the variances of the features instead of the covariance matrix. The diagonal matrix helps to reduce the computational complexity of the implementation.

The following section describes the MFCC generation process.

2.2 Description of MFCC Generation

The MFCC short-term spectral features are the most widely used features for speaker identification systems and this section is dedicated to describing the generation steps and the reasons behind them. The following are the steps involved:

1. Divide the digitalized speech signals into frames.
2. Compute the power spectrum of each windowed frame and estimate the energy
3. Generate a triangular filterbank and apply it to the power spectrum then sum all energies in a filter
4. Take the logarithm of all filterbank energies.
5. Compute the DCT of all logarithm filterbank energies
6. Drop the first coefficient and choose the second to the $\frac{Nf}{2}$th coefficients as the MFCC. Where Nf is the number of filters.

2.2.1 Implementation Steps

a. Frame the digitalized speech signal into 20ms frames. This step is performed because it is assumed that the audio signal changes rapidly but is considered statistically constant when very short frames are considered. These frames are assumed to estimate the shape of the spectral envelope if they are appropriately sized. If they are too short, they might not have enough samples to model the spectrum and if they are too long, they might not be considered as constant any longer. The frames need to be assumed constant over its length so that it can represent a single phoneme. The frames are made to overlap by about 10ms. The number of samples in a single frame will depend on the audio signal sampling frequency. If the sampling frequency is 44100Hz for example, and the frame duration is 20ms, the number of samples per frame would be [12]

Sampling frequency = 16000samples in 1 sec

N samples in 0.020secs

N = 0.02 x 44100 = 882 samples per frame

Frames that are not 20ms long can be patched with zeroes to make up the no of samples.

b. The power spectrum of each windowed frame is then computed. The windowed frame is the frame passed through a windowing function (e.g. hamming window). Windowing helps to smoothen the transition between one frame and the next. The power spectrum is computed using a DFT where the length of DFT is usually 512 but only the first 257 samples ($\frac{512}{2}$ +1) are kept.

$$S_t(k) = \sum_{n=0}^{N-1} s_t(n)h(n)e^{-j2\pi kn/N} \qquad 0 \le k \le K-1 \qquad (2)$$

K is the length of the DFT, N is the number of samples in a frame $s_t(n)$ *i*s the speech signal in the time domain and $h(n)$ is the hamming window coefficient. $\vec{S}_t$ *i*s the speech signal in the frequency domain and t is the frame index.

Motivation for this step comes from the human ear (the cochlea) which is sensitive to different frequencies. The cochlea responds to sounds by the vibration of some of its parts and the rate depends on what the frequency of the sound is. The DFT therefore aims to represent this function of the human ear by elaborating on the magnitude of the audio signal available at each frequency. The square of the absolute value of the complex spectrum is calculated to obtain the power estimates [12].

$$P_t(k) = \frac{1}{N}|S_t(k)|^2 \qquad\qquad 0 \leq k \leq K-1 \qquad\qquad (3)$$

$\vec{P}_t$ is the power spectrum estimate and t is the index of a frame. K is the length of the DFT and N is the number of samples per frame

c. A triangular filterbank will now be generated. The filterbank has a purpose of also mimicking the human ear. The cochlea does not discern between sounds with very close frequencies and this behavior is even pronounced at higher frequencies. The filterbank is a set of narrow triangular filters which are closely spaced at frequencies below 1000Hz and a set of wider triangular filters which are more widely spaced out at frequencies above 1000Hz [12]. Therefore, the filters take the estimate of energies within a frequency range and sum them up to have an idea of how much energy exists in the frequency range covered by each filter. This models how the human ear works and after this step, we will have a vector that represents the energy bins located in that frame. The length of the vector is equal to the number of filters and the number of filters used is usually from 20-40 (26 for this thesis). The filterbank is therefore in the form of 26 vectors each of length 257 (for FFT length = 512). Each vector is mostly zero but is non-zero within the range of frequency it is concerned with [12]. The triangular filterbank is actually a Mel frequency triangular filterbank (or Mel filterbank for short). Below are steps that show how to compute the Mel filterbank. [12]

   i. The upper and lower frequency ranges are converted to the Mel frequency scale. As stated previously, the Mel frequency helps model how the human ear functions [12]. The upper frequency bound cannot be greater than half the sampling frequency

(Nyquist Criterion). So if the sampling frequency is 44.1 KHz, as it is in this thesis, the upper frequency bound will be ≤ 22.05 kHz. The formula below is used to convert to the Mel frequency band [12].

$$MF(f) = 1125 \ln\left(1 + \frac{f}{700}\right) \tag{4}$$

$MF(f)\ is\ the\ Mel\ frequecy\ equivalent\ of\ f.\ \ f\ are\ the\ normal\ frequency\ points$

Lower bound, 0Hz = 0 Mels, Upper bound, 22.05 Hz = 3916.4 Mels.

ii. A number of points equal to the number of filters are then linearly created between the upper and lower Mel frequencies. For example, if we need 26 filters, we will create 26 extra Mel frequency points between the lower and upper Mel frequencies to give 28 Mel frequency points as shown below. [12]

m= [0, 145.1, 290.1, 435.2, 580.2, 725.3, 870.3, 1015.4, 1160.4, 1305.5, 1450.5, 1595.6, 1740.6, 1885.7, 2030.7, 2175.8, 2320.8, 2465.9, 2610.9, 2756.0, 2901.0, 3046.1, 3191.1, 3362.2, 3481.2, 3626.3, 3771.3, 3916.4]

iii. The Mel frequency points and then converted back to normal frequencies using the equation shown below.

$$MF(m)^{-1} = 700\left(\exp\left(\frac{m}{1125}\right) - 1\right) \tag{5}$$

$MF(m)^{-1}\ is\ the\ conversion\ from\ Mel\ frequency\ to\ normal\ frequecy\ range$

$m\ are\ the\ Mel\ frequency\ points$

Then we have 28 frequency points which are not linearly spaced out as shown below. [12]

f = [0, 96, 206, 331, 472, 634, 817, 1026, 1264, 1534, 1841, 2191, 2589, 3041, 3556, 4142, 4808, 5566, 6429, 7410, 8526, 9796, 11240, 12883, 14752, 16879, 19298, 22050]

iv. In order to put the frequencies at the required frequency resolutions, we convert them to an FFT bin number according to the formula below. [12]

$$fb(j) = floor\left((K+1) * \frac{f(j)}{fs}\right) \qquad 0 \leq j \leq Nf + 1 \qquad (6)$$

$fs\ is\ the\ sampling\ frequency, Nf\ is\ the\ number\ of\ filters$

$f(j)\ are\ the\ frequency\ points\ obtained\ above$

$K = 512\ is\ the\ length\ of\ the\ FFT$ and $fb\ are\ the\ frequency\ bin\ points.$

A sample of the bin points are shown below.

fb = [0, 1, 2, 3, 5, 7, 9, 11, 14, 17, 21, 25, 30, 35, 41, 48, 55, 64, 74, 86, 99, 113, 130, 149, 171, 196, 224, 256]

v. The filterbanks are then computed using the formula below. [12]

$$H_l(k) = \begin{cases} \dfrac{k - fb(l-1)}{fb(l) - fb(l-1)} & fb(l-1) \leq k \leq fb(l) \\ \dfrac{fb(l+1) - k}{fb(l+1) - fb(l)} & fb(l \leq k \leq fb(l+1) \\ 0 & otherwise \end{cases} \qquad 0 \leq k \leq K/2 \qquad (7)$$

$K\ is\ the\ length\ of\ the\ FFT$

$l\ ranges\ 1 \leq l \leq Nf, where\ Nf = 26\ is\ number\ of\ filters$

Following is a diagram of what the Mel frequency filterbank looks like

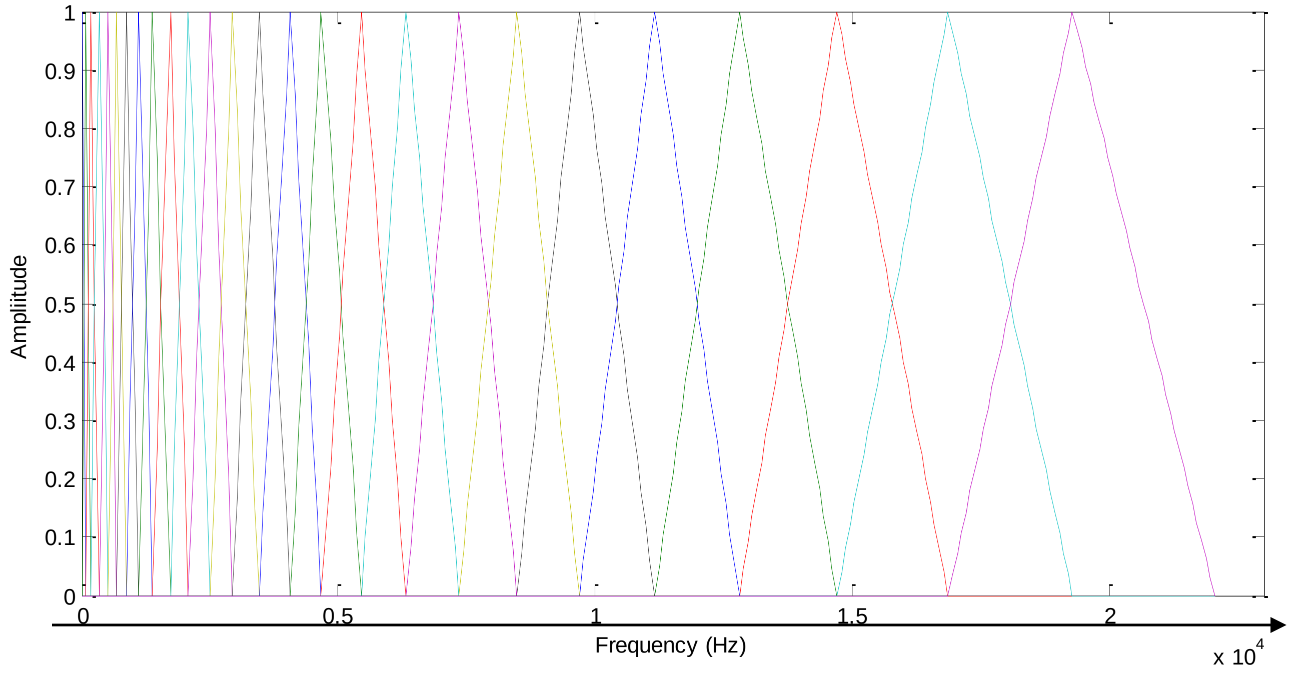

Figure 2.2 Mel Frequency Triangular Filterbank

d. The filterbank energies are then computed as shown below.

$$E(l) = \sum_{k=0}^{K/2} H_l(k) * P_t(k) \qquad 0 \leq k \leq \frac{K}{2} \qquad 1 \leq l \leq Nf \qquad (8)$$

K is the length of the DFT, Nf =26 is the number of filters, $\overrightarrow{P_t}$ is a (257 X 1) vector and $\overrightarrow{H_l}$ is a (257 X 1) vector. Each $\overrightarrow{H_l}$ represents one filter and there are 26 filters used in this thesis. The elements of the power spectrum vector $\overrightarrow{P_t}$, obtained from step b above are multiplied by the corresponding elements of each of the Mel frequency filters, $\overrightarrow{H_l}$ and summed up. We have one value from the operation described in the last sentence and eventually we obtain 26 values from the 26 filters. These coefficients represent the filterbank energies.

We take the logarithm of the energies to mimic the perceptive behavior of the human ear. For the ear to perceive a sound 2 times as loud, the energy has to be increased about 8 fold, so this operation tries to match this [12].

e. DCT is performed on the logarithm filterbank energies (computed in the previous step) to de-correlate it.

$$x_t(c) = \sum_{l=1}^{Nf} cos\left(\frac{c\left(l-\frac{1}{2}\right)\pi}{Nf}\right)[logE(l)] \qquad 1 \le c \le Nc \tag{9}$$

$l\ is\ the\ index\ of\ the\ lth\ filter, Nf\ is\ the\ number\ of\ filters\ and\ t\ is\ the\ index\ of\ the\ frame$

$c\ is\ the\ index\ of\ the\ cepstal\ coeficeints\ and\ Nc\ is\ the\ number\ of\ cepstrum\ coefficients$

f. The first coefficient is dropped because it depends only on the DCT and it is characteristically a number greater than the other energies. The second to *Nf*/2 others are chosen. Where Nf is the number of filterbanks used for the processing. The upper $\frac{Nf}{2} + 1$ coefficients are also dropped because they are thought to represent the fast changing part of the filterbank energies and they cause a deterioration of the speaker identification performance. Therefore for a 26 filterbank, operation, only coefficients 2 -13 are selected [12]. A sample of a frame of feature coefficients is shown below.

$\vec{x}_t$ = [-79.3475, -0.9081, -8.5389, 0.6032, 2.3761, 0.2477, -2.1180, 0.9455, 1.9868, -1.8905, 0.2718, 2.1765, 1.4305, 2.0616, 1.2119, 0.4137, 0.7492, 0.0904, -1.1775, -0.4730, 0.4516, 0.2948, -0.5236, 0.5048, 1.0820, 0.0216]^T

Dropping the 1st coefficient (obviously larger value) and selecting the 2nd to the 13th, we have

$\vec{x}_t$ = [-0.9081, -8.5389, 0.6032, 2.3761, 0.2477, -2.1180, 0.9455, 1.9868, -1.8905, 0.2718, 2.1765, 1.4305]^T.

where T is the vector transpose and t is the frame index

2.3 Error Compensation methods

For a speaker identification system to be robust and precise in its detection, it is necessary that all potential causes of error be eliminated. One such cause of error is a mismatch in training and testing conditions. Effort therefore needs to be made to ensure a similarity in training and testing conditions. This will improve the system performance and help to reduce errors. These efforts come in several forms and can be applied to different stages of the speaker identification systems. A few error compensation or avoidance methods are discussed below.

Voice Activity Detection (VAD)

VAD as the name suggests is the scheme that detects voice activity in a speech signal. Testing and training speech sometimes contain spaces of silence which might introduce some error during enrollment. These spaces of silence can have a grave effect on the speech feature vectors especially when there is insufficient training data. To avoid this scenario, VAD was conceived. The VAD works on the principle of measuring the signal energy in a speech signal and setting a threshold to screen out samples with insufficient energies. Then only samples with sufficient energy are accepted for front end processing.

Feature Normalization

Normalization of the feature vectors is performed to get rid of external influences on the speech signals. The speech signal can be influenced adversely by intersession variability, several effects from the channel, recording microphone, environmental noise, electronic noise etc. the magnitude of this disturbance will vary depending on the type of recorder used and the serenity of the environment. A robust speaker identification system should be able to overcome this challenge and feature normalization helps to achieve this.

A simple, yet effective method of mean normalization which involves the subtraction of the mean feature from the entire utterance can be used [5]. This method is called cepstral mean subtraction (CMS). A few sections back, a description of the cepstral features was given and it should be recalled that the final step was to take the DCT of the logarithm filterbank. The introduction of the logarithm makes this method possible [5].

A subtraction of the noise component then becomes easier due to the property of the cepstral features . The standard deviation of the features can also be used to divide the features to equalize their variances. These methods together get rid of the channel effects to an extent and that is the method used in this thesis (Cepstral Mean and Variance Normalization).

$$\vec{x}_t{}' = \frac{\vec{x}_t - \overrightarrow{mx}}{\vec{\sigma}} \tag{10}$$

where $\overrightarrow{mx}$ is the mean feature vector obtained as follows

$$\overrightarrow{mx} = \frac{1}{T}\sum_{t=1}^{T} \vec{x}_t \tag{11}$$

where t is frame index and the T is the total number of frames present

$\vec{\sigma}$ is the standard deviation of the feature vectors

$\vec{x}_t$ is a feature vector with i denoting the index of the frame and

$\vec{x}_t{}'$ is the cepstral mean and variance normalized feature vector

Chapter 3: Feature Modeling

Having extracted the features from the speech signals, the next stage is the creation of a feature model. This model is a statistical representation of the acoustic content of the speech signals. Therefore every speaker in the database will have a model which is a representation of that speaker's phonemes. The models will serve as the objects of comparison when new speech signals are tested to obtain the speaker identity. Several feature modeling techniques exist and they are categorized according to some general characteristics.

3.1 Overview of Feature Modeling

3.1.1 Vector Quantization (VQ)

Vector quantization is one of the simplest speaker modeling techniques. In this method, a small set of feature vectors is used to represent the acoustic characteristics of the speaker. These small set of features is obtained using clustering methods such as k-means. The set of features is referred to as the codebook of the speaker. During detection or testing of new speech features, the features are compared to the codebooks of all the speakers in the database and the speaker whose codebook has the highest likelihood gets detected. This model is simple to implement but is does not work very efficiently when the number of feature vector is large.

3.1.2 Dynamic Time Warping (DTW)

This method models speech utterances as a sequence of feature vectors. This sequence is aligned with the template feature vector sequence using a DTW algorithm. Then the overall distance between template and test sequence is used to make a decision [14]. Because this method of modeling depends on aligning template with test sequence to observe difference, it is not applicable in text-independent speaker identification systems. In text-dependent speaker identification systems, however, it finds great use because the uttered words would have a similar phonetic structure to the template sequence, then a direct comparison of both can be done.

3.1.3 Hidden Markov Model (HMM)

HMM is a stochastic method which models the speech signals as a sequence of states and transitions between those states. Speech segments are categorized into broad phonetic classes using the multiple states. In the training phase, the phonetic categories of each speaker is generated and saved as a template. During testing, after the generation of testing phonetic categories, the reference templates of each speaker are compared to the test in order to find a match. This method is applied more in speech recognition systems. It is not used in text-independent speaker recognition either because the rate of recognition was shown to correlate with the number of mixtures regardless of the number of states [14].

3.1.4 Gaussian Mixture Model (GMM)

The GMM is also a stochastic model which represents speaker feature vectors as a weighted sum of M component density functions. The modeling method has become the industry standard for implementing speaker identification systems. GMM in conjunction with its derivatives will be discussed in the next section.

3.2 Gaussian Mixture Model

A GMM can be represented by the expression below.

$$p(\vec{x}|\lambda) = \sum_{i=1}^{M} w_i b_i(\vec{x}) \qquad (12) \quad [5]$$

M is the number of component Gaussian distributions in the GMM and $\vec{x}$ is a D dimensional feature vector [5]. $w_i$is the mixing weight of each Gaussian density $b_i(\vec{x})$ and satisfies

$$\sum_{i=1}^{M} w_i = 1 \tag{13}$$

$b_i(\vec{x})$ is a multivariate Gaussian function and it is expressed as

$$b_i(\vec{x}) = \frac{1}{2\pi^{\frac{D}{2}}|\Sigma_i|^{\frac{1}{2}}} exp\{-1/2(\vec{x}-\vec{\mu}_i)\acute{}\boldsymbol{\Sigma}_i^{-1}(\vec{x}-\vec{\mu}_i)\} \tag{14}$$ [5]

$\vec{\mu}_i$ is the mean vector and $\boldsymbol{\Sigma}_i$ is the covariance matrix. D=12 is the dimension of the feature vector $\vec{x}_t$ and Gaussian distribution $b_i(\vec{x})$. The parameters of the Gaussian mixture are represented by [5]

$$\lambda=\{w_i, \vec{\mu}_i, \boldsymbol{\Sigma}_i\} \qquad i = 1, \ldots M \tag{15}$$

$\vec{\mu}_i$ is the mean and Σ_i, a diagonal matrix, is the variance of the feature vectors. This diagonal matrix does a good job at modeling the features because the feature vectors had been de-correlated with the DCT during feature generation. The full covariance matrix can be used and a single full covariance matrix can as well be used for all the components [5].

The intuition behind modeling the speaker features using the GMM can be visualized if each component multivariate Gaussian is made to represent a single acoustic class. So the number of components in the GMM will be the number of acoustic classes represented by the model. To obtain a speaker training model, the training data has to be used to evaluate the estimation of the GMM parameters. Several methods are available for this, but the most popular is the Maximum Likelihood (ML) estimation method. The idea behind this method is that given a set of speech signal features (training data) the model parameters which maximize the likelihood of the GMM are estimated.

If training vectors are **X** = $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_T\}$ where D=12 is the dimension of the feature vector $\vec{x}_t$

The GMM likelihood is $p(\boldsymbol{X}|\lambda) = \prod_{t=1}^{T} p(\vec{x}_t|\lambda)$ (16)

Maximizing the likelihood of the above expression with the ordinary form of the maximum likelihood will not be feasible because the expression is non-linear due to the GMM. The computation can still be obtained, but with a different algorithm called the Expectation Maximization (EM) algorithm [5].

Expectation Maximization (EM) Algorithm

This is an iterative algorithm which is used to compute the maximum likelihood of difficult problems. Computation of the GMM parameters is made possible by enlarging the samples with latent data. The latent data, the given data and estimates of the GMM parameters are therefore used to obtain the desired GMM parameters. For example, in estimating the GMM parameters using the EM algorithm, the latent data is the probability of data frame t belonging to GMM component i. This information is missing. Instead, the expectation of the probability of data frame t belonging to GMM component i is calculated, and called the responsibility [15]. The estimates of the parameters are then used to assign the responsibilities according to the relative density of the observation of the training data [15]. This is the expectation phase and it facilitates the calculation of the GMM parameters. The maximization phase involves updating the estimates of the parameters using the responsibilities in weighted maximum likelihoods [15].

The EM estimates the GMM parameter $\boldsymbol{\lambda}_n$ (where $\boldsymbol{\lambda}_n$ is the result obtained from the nth iteration of the GMM model) from the previous iteration, $\boldsymbol{\lambda}_{n-1}$ and the iteration is continued until some threshold of convergence is attained such as $p(X|\boldsymbol{\lambda}_n) \geq p(X| \boldsymbol{\lambda}_{n-1})$. The first iteration needs to start with an initial estimated model (that can either be random or obtained from a clustering method) and the next iteration is computed from the most recent one. The following formulas are used on each step of the iteration. [5]

EXPECTATION PHASE:

Mixture Weights: $w_i = 1/T \sum_{t=1}^{T} p(i|\, \overrightarrow{x_t},\, \lambda)$ (17) [5]

Means: $\hat{\hat{\mu}}_i = \frac{\sum_{t=1}^{T} p(i|\, \overrightarrow{x_t}, \lambda)\vec{x}_t}{\sum_{t=1}^{T} p(i|\, \overrightarrow{x_t}, \lambda)}$ (18) [5]

Variances: $\hat{\hat{\sigma}}_i^{\;2} = \frac{\sum_{t=1}^{T} p\left(i \middle| \overrightarrow{x_t},\, \lambda\right)(\vec{x}_t - \hat{\hat{\mu}}_i)(\vec{x}_t - \hat{\hat{\mu}}_i)'}{\sum_{t=1}^{T} p(i|\, \overrightarrow{x_t}, \lambda)}$ (19)

[5]

$\hat{\hat{\sigma}}_i^{\;2}$ is the new variance, $\hat{\hat{\mu}}_i$ is the new mean, t is the frame index, T is the number of frames and i is the Gaussian component index.

MAXIMIZATION PHASE:

A posteriori probability:

$$p(i|\, \overrightarrow{x_t}, \lambda) = \frac{w_i b_i(\overrightarrow{x_t})}{\sum_{k=1}^{M} p_k b_k(\overrightarrow{x_t})} \quad (20) \quad [5]$$

M is the number of component Gaussian distributions

3.2.1 Universal Background Model

After the creation of speaker models, some methods of testing (hypothesis testing) involve comparing the speaker model to an imposter model or an alternative model. In [7], the idea of the universal background model (UBM) was introduced. The UBM is an alternative speaker model that is representative of the all the expected alternative speakers during testing. Instead of having different alternative speakers to test different test feature vectors, the UBM can be used as a universal alternative model against which the test vectors are compared. The UBM is therefore expected to be a representation of the expected speech in both type and quality [7]. For example, if test vectors are expected to come from male and female, the UBM should contain both types of speech samples. If the test samples can come from speakers with different accents, the UBM should be made from speeches of speakers with all such expected accents. This universal background model is modeled from scratch using speech samples from all the representative speakers. There are a few ways to create the UBM. One way, which is the method used in this thesis is the pooling of all feature vector samples from the available speakers in the speech database and creating UBM using the EM algorithm. Another method of creating the UBM is by training individual GMM models specific to constituent populations and pooling the individual GMM models to create a larger model – the GMM UBM. For example in a database that contains British and Americans, a British GMM model is trained from speech samples of all the British speakers and an American GMM model is trained from all speech samples from all American speakers. The 2 GMM models are then pooled to obtain the larger GMM UBM. Both methods work well but what should be kept in mind is that there must be a balanced representation of the constituent sub-populations in the resulting UBM otherwise there would be a bias towards a particular group during testing of the test features. [7]

3.2.2 Adapted Gaussian Mixture Model

It has been observed that during training, it is more efficient to create speaker models by adaptation of a world or universal model. This has been studied [7] and identified to produce better recognition results. An explanation of this is that due to speaker variability, it is better to adapt speaker models from a universal model to ensure a better correlation between the speaker model and the universal model. To train a new speaker, the parameters of the UBM are adapted to the feature vectors of the new speaker. The method used is a form of Bayesian adaptation or the Maximum a posteriori (MAP) method. The first step which is the calculation of the sufficient statistics is similar to the expectation phase of the EM algorithm. Adapted GMM models are calculated using the expressions below.

A posteriori probability:

$$P(i|\,\overrightarrow{x_t},\lambda) = \frac{w_i b_i(\overrightarrow{x_t})}{\sum_{k=1}^{M} w_k b_k(\overrightarrow{x_t})} \qquad (21)\ \ [7]$$

The Sufficient Statistics [7]:

Mixture Weights: $$n_i = \sum_{t=1}^{T} P(i|\,\overrightarrow{x_t}) \qquad (22)\ \ [7]$$

Means: $$E_i(\overrightarrow{x_t}) = \frac{\sum_{t=1}^{T} P\left(i\middle|\,\overrightarrow{x_t}\right)\vec{x}_t}{n_i} \qquad (23)\ \ [7]$$

Variances: $$E_i\left(\overrightarrow{x_t}^2\right) = \frac{\sum_{t=1}^{T} P\left(i\middle|\,\overrightarrow{x_t}\right)\overrightarrow{x_t}^2}{n_i} \qquad (24)\ \ [7]$$

T is the total number of feature vectors, i is the index of the mixture component and M is the number components. $\overrightarrow{x_t}^2$is the short form for diag($\overrightarrow{x_t}\,\overrightarrow{x_t}'$). The sufficient statistics from the training data is then used to update UBM. [7]

Adaptation [7]**:**

$$\widehat{w}_i = \left[\frac{\alpha_i{}^{w} n_i}{T} + (1-\alpha_i{}^{w})w_i\right]\gamma \qquad (25) \qquad [7]$$

$$\hat{\vec{\mu}}_i = \alpha_i{}^{m} E_i(\overrightarrow{x_t}) + (1-\alpha_i{}^{m})\vec{\mu}_i \qquad (26) \qquad [7]$$

$$\hat{\vec{\sigma}}_i{}^2 = \alpha_i{}^{v} E_i\left(\overrightarrow{x_t}^2\right) + (1-\alpha_i{}^{v})\left(\vec{\sigma}^2 + \vec{\mu}_i{}^2\right) - \hat{\vec{\mu}}_i{}^2 \qquad (27) \qquad [7]$$

$\widehat{w}_i$ is the adapted mixture weight, $\hat{\vec{\mu}}_i$ is the adapted mean vector and $\hat{\vec{\sigma}}_i{}^2$ is the adapted diagonal variance vector.

w_i , $\vec{\mu}_i$ *and* $\vec{\sigma}^2$ are the weights, means and variances of the UBM respectively.

$\alpha_i{}^{\rho}$ is the adaptation coefficient for each parameter $\rho \in \{w, m, v\}$and is defined in [7] as

$$\alpha_i{}^{\rho} = \frac{n_i}{n_i + r^{\rho}} \qquad (28)$$ [7]

Where r^{ρ} is the relevance factor. γ is the scale factor computed over all adapted mixture weights and they sum to unity. [7]

For this thesis, only the mean vectors are adapted as it was found that this performs almost as well as adapting the 2 other parameters.

Chapter 4: Speaker Identification

Having gone over the speech feature extraction and training phase (also called enrollment), this section discusses the detection or testing phase. It is necessary to have a method of detection. This phase attempts to tell us the identity of the speaker of a new speech signal. Since we have utilized training data from all speakers to enroll the respective speakers, we now have our reference models. This models can be used as a means of comparison against new feature vectors (test features).

4.1 Overview of Speaker Identification/Detection Methods

Several techniques are available to perform testing of new speech signals. Below is a discussion of a few of them.

4.1.1 Euclidean Distance

This technique is the simplest measure of distance and it produces good results in some applications. The measure of distance is carried out as shown in the formula below

$$d(\overrightarrow{mx}, \vec{\mu}) = \sum_{i=1}^{M} (\overrightarrow{mx} - \vec{\mu}_i)'(\overrightarrow{mx} - \vec{\mu}_i) \qquad 1 \leq i \leq M \qquad (29)$$

Where $\overrightarrow{mx} = \frac{1}{T}\sum_{t=1}^{T} \vec{x}_t$ (30)

$\overrightarrow{mx}$ is the mean of all test feature vectors and T is the total number of frames and $\vec{\mu}_i$ is the mean parameter of the speaker model. The total distance will be summed up over all the distances from the component means.

Because speaker models are usually created using GMMs, this method does not give very good results for text-independent speaker identification system. This is so because there is no consideration for the covariance and weights of the models. This method is suitable for text-dependent speaker identification systems when schemes like VQ AND DTW are used.

4.1.2 Mahalanobis Distance

This method can produce a better performance result for speaker identification systems due to the consideration of the covariance in the distance formula shown below.

$$d(\overrightarrow{mx}, \lambda_s) = \sum_{i=1}^{M} (\overrightarrow{mx} - \vec{\mu}_i)' \boldsymbol{\Sigma}_i^{-1} (\overrightarrow{mx} - \vec{\mu}_i) \qquad 1 \le i \le M \tag{31}$$

$\overrightarrow{mx}$ is the mean test feature vector and is defined in the previous subsection, λ_s is a speaker model, $\vec{\mu}_i$ is the mean parameter of the speaker model and $\boldsymbol{\Sigma}_i$ is the covariance matrix of the speaker model. The total distance can be summed up over all the distances from the component means. This method is used to measure the distance between 2 multi-dimensional quantities. It will be the same as Euclidean distance if the covariance is replaced by a matrix of ones. It finds use in other related applications like cluster analysis and classification techniques.

4.1.3 Bhattacharya Distance

This method is similar to the Mahalanobis method because the covariance is considered. it however goes a little further by incorporating the covariance of the test samples. So in an operation where the inclusion of the test covariance is useful, then this method will outperform the Mahalanobis distance [3].

The formula is given below

$$d(\lambda_r, \lambda_s) = \frac{1}{2} \ln \left(\frac{\left| \frac{\boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j}{2} \right|}{|\boldsymbol{\Sigma}_i|^{\frac{1}{2}} |\boldsymbol{\Sigma}_j|^{\frac{1}{2}}} \right) + \frac{1}{8} (\vec{\mu}_i - \vec{\mu}_j)^T \left(\frac{\boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j}{2} \right)^{-1} (\vec{\mu}_i - \vec{\mu}_j) \tag{32}$$

λ_r is the test speaker model and λ_s is the target speaker model. i is the index for the component parameters of the test speaker model and j is the index for the component parameters of the target speaker model. $\vec{\mu}$ is the mean vector and $\boldsymbol{\Sigma}$ is the covariance matrix. The second expression looks similar to Mahalanobis distance with the exception of the average covariance matrix.

4.4.4 Kullback-Leibler Divergence Measure

This is also a measure of distance which can calculate the difference between probability distributions using information theory. It is expressed as shown below.

$$D(p_i||p_j) = \sum_{\vec{x}} p_i(\vec{x}) ln \frac{p_i(\vec{x})}{p_j(\vec{x})} \tag{33}$$

$$where\ p_i(\vec{x}) = p(\vec{x}|i)\ and\ p_j(\vec{x}) = p(\vec{x}|j)$$

Where p_i is the likelihood of occurrence of $\vec{x}$ given that it belongs to class i and p_j is the likelihood of occurrence of $\vec{x}$ given that it belongs to class j. i and j represent 2 different classes and $\vec{x}$ is a feature vector. The divergence is defined as the amount of information needed to distinguish p_i and p_j or the amount of information lost when p_i is used to approximate p_j. While this distance measure is said to be non-symmetric, it can be symmetrized and finds a lot of use in many applications.

4.1.4 Likelihood Function

The likelihood function finds a lot of use in statistical inferences and is use to estimate a parameter from a set of statistics. The likelihood of a parameter Ɵ, given data **X** is the probability of **X** given parameter Ɵ. The likelihood function is widely used in speaker identification systems.

In a speaker identification system, a group of S speakers $\{s_1, s_2, \ldots, s_S\}$ in our database have speech features represented by their GMM parameters$\{\lambda_1, \lambda_2, \ldots, \lambda_s\}$. During speaker identification, the role of the system is to find the parameter that matches a speaker the most, given feature vector, **X**. i.e

$$p(\lambda_r|\boldsymbol{X}) \qquad\qquad 1 \le r \le S \tag{34}$$ [5]

This can be evaluated using Bayes rule as follows

$$\arg max\, p(\lambda_k|\boldsymbol{X}) = \frac{p(\boldsymbol{X}|\lambda_r)P(\lambda_r)}{p(\boldsymbol{X})} \qquad 1 \le r \le S \tag{35}$$

Where S is the number of speakers, r is the index of the rth speaker, $P(\lambda_r)$ is the probability of parameter r (= 1/S) and all parameters are equally likely therefore a constant

$p(\boldsymbol{X})$ is the same for all speakers so the above expression simplifies to

$$\arg max\, P(\boldsymbol{X}|\lambda_r) \qquad 1 \le r \le S \tag{36}$$ [5]

The identification process therefore is a likelihood function and logarithm can be introduced plus the independence property of the observations to help simplify the resulting expression because

$$\arg max\, logP(\boldsymbol{X}|\lambda_r) = \arg max \log \prod_{t}^{T} P(\vec{x}_t|\lambda_r) \qquad 1 \le r \le S \tag{37}$$

$$= \arg max \sum_{t=1}^{T} log\, P(\vec{x}_t|\lambda_r) \qquad 1 \le r \le S \tag{38}$$

T is the total number of test frames. And S is the number of speakers in the database.

It should be recalled that the GMM,

$$p(\vec{x}_t|\lambda) = \sum_{i=1}^{M} w_i b_i(\vec{x}_t) \tag{39}$$

Where M is the number of component distributions, $b_i(\vec{x}_t)$ is the multivariate normal distribution and w_i is the weight of the normal distributions.

$$\arg max\, logP(\boldsymbol{X}|\lambda_r) = \arg max \sum_{t=1}^{T} log \sum_{i=1}^{M} w_i b_i(\vec{x}_t) \tag{40}$$

$$\arg max\, logP(\boldsymbol{X}|\lambda_r) = \arg max \sum_{t=1}^{T} log \sum_{i=1}^{M} w_i \frac{1}{2\pi^{\frac{D}{2}}|\boldsymbol{\Sigma_i}|^{\frac{1}{2}}} exp\{-1/2(\vec{x}_t - \vec{\mu}_i)\acute{}\boldsymbol{\Sigma_i}^{-1}(\vec{x}_t - \vec{\mu}_i)\} \tag{41}$$

Where $\vec{\mu}_i$ is the mean vector, $\boldsymbol{\Sigma_i}$ is the covariance matrix and D=12 is the dimension of $\vec{x}_t$

The likelihood ratio

The likelihood function described in the last section selects the speaker with the maximum likelihood value as the speaker. In speaker identification applications, a similar technique can also be used for detection but is more robust and allows for performance measurement. It is a basic hypothesis test which assigns the results of the test to one of the hypothesis depending on the value of the likelihood ratio.

Hypothesis Test:

X is from target speaker: H_o

X is from non-target speaker: H_1

$$\frac{P(\boldsymbol{X}|H_0)}{P(\boldsymbol{X}|H_1)} \begin{cases} \geq \ \emptyset \ accept\ H_0 \\ \leq \ \emptyset \ reject\ H_0 \end{cases} \qquad (42)\ [7]$$

Where H_0 is the hypothesis that **X** is from the target speaker and H_1 is the hypothesis that **X** is from the imposter speaker. $P(\boldsymbol{X}|H_0)$ is the likelihood of the hypothesis H_0 given **X** and $P(X|H_1)$ is the likelihood of hypothesis H_1 given **X**. $\emptyset$ is some threshold that needs to be fixed to ensure that the system determines which of the hypothesis is correct.

From the previous section, we discussed how to obtain H_0, which is the likelihood that **X** is from one of the speakers. The hypothesis that **X** is not the target speaker, however has not been discussed and we go on to do so. During testing, we are not sure of the identity of the speaker of the test features. We need to find a suitable model that represents the H_1 hypothesis, the non-target speaker. A solution can be found in the Universal background model concept discussed previously. The UBM is a speaker model that contains features from all representative speakers in our database. A single model represents all speakers [7]. This model can be used as the hypothesis, H$_1$, and it has been shown to produce impressive results [7]. After obtaining the result of the Log-likelihood Ratio, the decision threshold needs to be set in order to observe the performance. The threshold can help determine the probability of False Alarm (FA) and Missed Detection.

The log of the likelihood ratio is the Log-likelihood ratio and it simplifies the calculation.

$$\text{Score} = \log P(\boldsymbol{X}|H_0) - \log P(\boldsymbol{X}|H_1) = log\frac{P(\boldsymbol{X}|H_0)}{P(\boldsymbol{X}|H_1)} \tag{43}$$

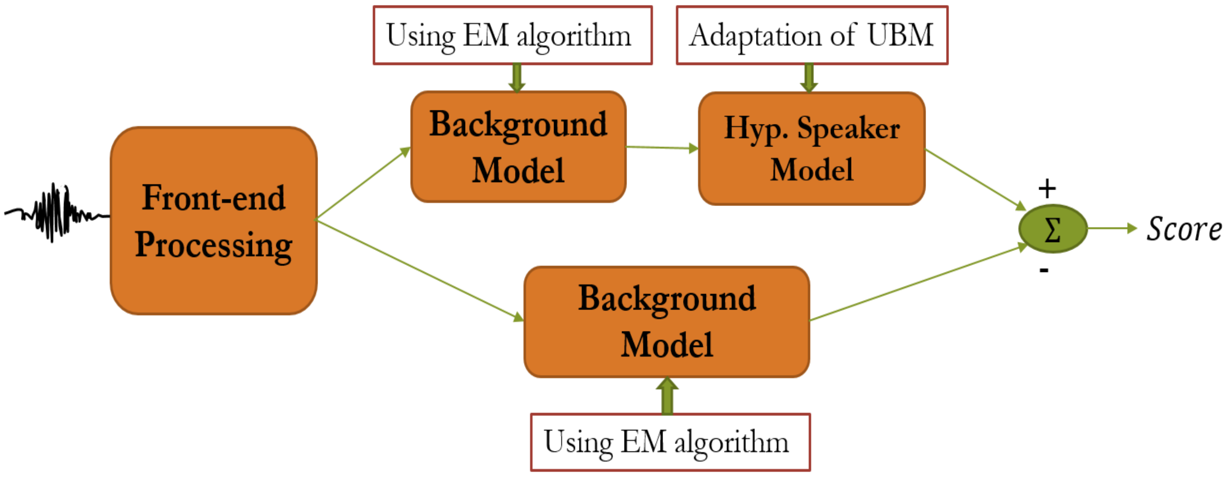

Figure 4.1: Speaker Identification System based on Likelihood Ratio

4.2 Score Normalization

The score after testing is normalized relative to other speakers. This is done to make the scores of different speakers reside in a similar range [2]. This eases the threshold setting step and ensures a single threshold can be used across all speakers. One such type is Zero-Normalization (Z-Norm) when the statistics (mean and variances) are target speaker dependent and they are obtained by matching a set of non-target features against the target model and obtaining the mean and standard deviation of the resulting scores. Another method is the Test-Normalization (T-Norm) and it differs from Z-Norm in that the statistics depend on the test utterance and are obtained by matching unknown test features and no-target models then calculating the mean and standard deviation. [2]

$$s' = \frac{s - sm}{\sigma} \tag{44}$$

s' is the normalized score

s is the un-normalized score

sm is the score mean

σ is the standard deviation of the scores

Chapter 5: Statistical Learning and Support Vector Machines

5.1 Introduction to statistical learning

As can be inferred from the name, statistical learning refers to using statistics to learn from data. It involves using the knowledge and techniques of statistics to understand the pattern, trend or other latent information embedded in data. This learned information will generally be used to classify that same data or a new set of similar data. In simple terms, statistical learning is the ability to utilize information from past and current data to predict the outcome or category of future data. Statistical learning is the engine for machine learning which finds tremendous use across various industries today. This technique is relevant due to the complex nature of the world today. With almost everything around us moving at an extra fast pace, it became necessary of researchers, statisticians and other professionals to find ways to use available data to make better decisions [15].

An example of its application is in the financial industry. Trading in the stock market is full of risks and predications and the economy of a nation depends a lot on it. The power of statistical learning has been put to use by using trade patterns of the past to predict the behaviors of stocks in the present. Bioinformatics has also benefited immensely from statistical learning, with the current capability and technology, it is now possible to make an early detection of a cardiac arrest or stroke by using information from the patient's past record and his living condition. This is due to the effectiveness of statistical learning or machine learning. Several other industries have used statistical learning to advance the effectiveness of different applications. Data scientists utilize this in detecting patterns in data that can help detect and thwart a network intrusion attempt. Computer scientists use machine learning to filter emails coming into an email box and thereby avoid the clogging of user email boxes by these un-solicited emails. Image detection and recognition supplications have become more efficient with the incorporation of this technique. Electrical engineers and computer scientists have also recorded giant strides in using statistical learning to improve the efficiency of speech and speaker recognition systems. That use will be showcased in this thesis. [15]

Due to the ubiquitous nature of statistical learning and machine learning, it has been categorized based on a couple of salient features. One of such features is the nature of output. Statistical learning is categorized based on the existence and or form of the output from the algorithm. This classification is

1. Supervised learning 2. Unsupervised learning

5.1.1 Supervised Learning

For this category, we have an input and an output. The input data is passed through the learning algorithm to understand the pattern and trend so the learned information is stored in a function referred to as a model (prediction model). This model is then the prediction tool to discern the output from a system. For example, in a spam server, statistical learning uses some information obtained from real emails and spam emails to develop a model. This model then classifies incoming emails as SPAM or NOT SPAM. In this case the output of the system is if the email is spam or not. Simply put, supervised learning involves utilizing previous or current data information to make a decision about the output of future data. In this category, there is always an output.

The type of output has also been used to classify supervised learning methods. Supervised learning can be a regression problem or a classification problem. A regression problem is one whose output is quantitative of continuous valued. An example would be determining the likelihood of a patient having a heart attack in the next week. The output will be a number or percentage which is continuous (within a particular range of course). A classification problem, on the other hand, deals with problems whose output is one of a discrete number of values or classes. The email example stated earlier is an example. Output is either SPAM or NOT SPAM. A speaker recognition or identification system is also designed as a classification problem. The system detects if the identity of a speaker is one of several speakers in the database. It can also be designed to detect if the new test speech is for speaker A or not speaker A. in this case, it is a binary classification problem.

5.1.2 Unsupervised Learning

This class of statistical learning deals with understanding the pattern of a given data input and that is it. Ok, there is a little more. It involves understanding the pattern of a data and clustering the data into sections based on information learned. So, as opposed to the previous category, there is no new data which will be categorized using a model built with information learned from input data. This category gets a little more complicated and is the method used in anomaly detection. Anomaly detection involves learning information from a collection of data and detecting if the data contains components which do not fit the general characteristics exhibited by the others. Then an algorithm which clusters the data into pools and omits the outlying (extraneous) data is used. This method is applied in stopping computer hacking, preventing financial frauds or in detecting medical problems.

5.2 Statistical Learning in Speaker Identification Systems

Speaker identification systems have become more efficient and more accurate in the past decade or more. This success, in part, is due to the power of statistical learning methods. Statistical learning has several components and these components have their perks. Depending on what the objective is, the choice of which one to use for a particular application needs to be made. Speaker identification systems have been designed to utilize several of these statistical learning algorithms. Speaker identification systems have been designed with neural networks, regression and functional analysis methods. In this thesis, 2 basic and important statistical learning components are used. The EM algorithm and the support vector machine (SVM). THE EM algorithm was discussed in chapter 2. It was used to compute the maximum likelihood of the parameters of the GMM. The method was utilized because Maximum likelihood (ML) estimation could not be efficiently used to compute the results for a non-linear GMM likelihood function. Using the EM algorithm, however, simplified the process but in an iterative manner.

The SVM concept will be the focus of the next section. Its theoretical basis will be explored and its concepts will be extended to a speaker identification system. While the SVM has been utilized by researchers for speaker identification systems in the past, this thesis looks to improve on previous knowledge by going a step further with the creation of a new kernel function that is aimed at improving results. It is hoped that by the time a novice in SVM goes thought the next section, he or she will have a grasp of what it is about, how it functions and how SVMs are applied to speaker identification systems. Later in the chapter, the setup for SVM used in this thesis is described including the proposed new algorithm for higher recognition rate of speakers in speaker identification systems.

5.2.1 Support Vector Machine

SVM is a 2 class classification system that makes classification decisions by creating a boundary between the 2 classes. This boundary separates the classes and is referred to as a hyperplane. The idea is therefore to create a hyperplane that best separates elements of the 2 classes.

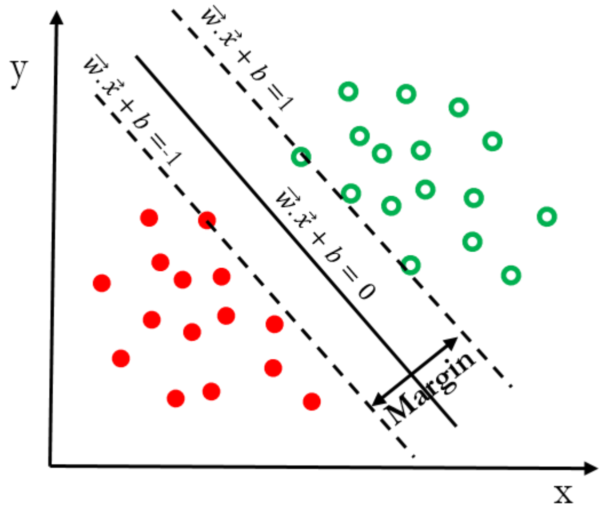

Figure 5.1: Class separation in Support Vector Machines

The hyperplane is represented by

$$\vec{w}.\vec{x} + b = 0 \tag{45}$$

Where $\vec{w}.\vec{x}$ represents the dot product of $\vec{w}\ and\ \vec{x}$. $\vec{x}$ is the data and $\vec{w}$ is the normal to the plane and b is called the bias. The margin is shown in the figure above and it is one of the important parts of the SVM. The creation of the best hyperplane to classify data is achieved by seeking to maximize the margin between data on both sides and the hyperplane. So for a linearly separable data represented by {$\vec{x}$, y} where $\vec{x}$ belongs to a d-dimensional real space and $y \in \{-1, 1\}$. y is called the class label of the data. The number of observations is equal to the number of class labels. Data points above and below the margin are represented by the following expressions respectively. [16]

$$\vec{w}.\vec{x} + b \geq 1\ for\ y = +1 \tag{46}$$

$$\vec{w}.\vec{x} + b \geq -1\ for\ y = -1 \tag{47}$$

When both are combined

$$y.(\vec{w}.\vec{x} + b) \geq 1 \tag{48}$$

Where $\vec{w}.\vec{x}$ is the dot product of $\vec{w}\ and\ \vec{x}$.The hyperplane is generated by the minimization of the norm of $\vec{w}$, subject to $y.(\vec{w}.\vec{x} + b) \geq 1$. This is an optimization problem and is represented as [10]

$$\min\ \left|\left|\vec{w}\right|\right|^2 \tag{49}$$

$$subject\ to\quad y.(\vec{w}.\vec{x} + b) \geq 1 \qquad (50)\ [10]$$

This can be solved using quadratic programming. To obtain the dual of this problem, we can use the Lagrange method as thus [10]

$$L = \frac{1}{2} * \left|\left|\vec{w}\right|\right|^2 - \sum_{t=1}^{T} \alpha_i y_i (\vec{w}.\vec{x}_t + b) - 1 \tag{51}$$

Where $\vec{w}.\vec{x}_t$ is the dot product of $\vec{w}\ and\ \vec{x}_t$ and α_t is the Lagrange multiplier and T is the number of observations in the training data

The dual is as follows

$$L = \sum_{t=1}^{T} \alpha_i - \frac{1}{2} * \sum_{t=1}^{T} \sum_{j=1}^{T} \alpha_t \alpha_j y_t y_j \vec{x}_t . \vec{x}_j \qquad (52) \quad [10]$$

$$subject\ to\ \sum \alpha_t y_t\ and\ \alpha_t\ \geq 0 \qquad (53)$$

where $\vec{x}_t.\overrightarrow{x_t}$ is the dot product of $\vec{x}_t\ and\ \vec{x}_j$ The above can be solved using quadratic programming and a set of values for α will be zero but some will be greater than 0. The indices of the Lagrange multipliers which are not zero correspond to the indices of the support vectors of the data. Therefore the solution for the hyperplane is $\vec{w}_o$, described as [10]:

$$\vec{w}_o = \sum_{t=1}^{T} \alpha_t y_t \vec{x}_t \qquad (54) \qquad [10]$$

When the data in question is not clearly linearly separable, the solution will be tweaked a little. To accommodate data that might violate the margin, a slack parameter is introduced. This slack modifies the optimization objective function slightly as follows

$$\frac{1}{2} * \left||\vec{w}|\right|^{2} + C \sum_{t=1}^{T} L(\xi_t) \qquad (55) \quad [10]$$

$$subject\ to\ y.(\vec{w}.\vec{x}_t + b) \geq 1 - \ \xi_t \qquad (56)$$

Where $\vec{w}.\vec{x}_t$ is the dot product of $\vec{w}\ and\ \vec{x}_t$ and ξ_t is the risk associated with violating the margin and C is the cost of violation of the margin. Then the dual optimization problem is the same but the constraint is a little different.

$$\sum_{t=1}^{T} \alpha_i - \frac{1}{2} * \sum_{t=1}^{T} \sum_{j=1}^{T} \alpha_t \alpha_j y_t y_j \vec{x}_t . \vec{x}_j \qquad (57)$$

subject to $\sum \alpha_t y_t$ and $0 \le \alpha_t \le C$ (58) [16]

Where $\vec{x}_t . \vec{x}_j$ is the dot product of $\vec{x}_t$ *and* $\vec{x}_j$.

Non-Linear SVMs

In real world applications however, it rarely happens that a set of data points will be separable with a linear hyperplane. What we find is a set of data points that are inter-mixed with data from another class. So it will take a tortuous line to form a boundary between the 2 if a separation is to be achieved (in 2 dimension of course). Given what was discussed till this moment, it is obviously impossible to achieve this. A solution to this deadlock came in what is referred to as a kernel. A solution to this problem is to transform the data in the input space to a feature space of high dimensionality. Then, the concept of SVM is applied in this space. The hyperplane is achieved in that feature space and it is used to separate the data in the feature space. After separation, the feature space data points and the hyperplane are transformed back to the input space and a boundary is achieved. But this boundary is not linear in the input space. It can be circular or wiggly, depending on the type of transformation used. The figure below shows a pictorial description. [10]

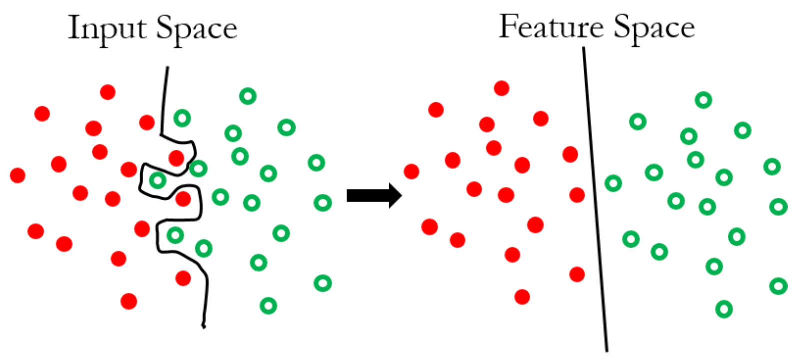

Figure 5.2 Transformation of Data from Input to Feature Space

5.2.2 Kernels

A kernel function is a mathematical function used to achieve the transformation of a data set from the input space to the feature space. It is expressed as $K(\vec{x}_t, \vec{x}_j) = \emptyset(\vec{x}_t)\emptyset(\vec{x}_j)$ [10]. Where $\vec{x}_t$ is the training vector and $\vec{x}_j$ is the testing vector and t and j are the indices for the training and testing vectors respectively. The transformation from input to feature space is represented by $\emptyset(\vec{x}_t)$**.** It can be recalled that the solution to the hyperplane is expressed as $\vec{w}_{o=}\ \sum \alpha_t y_t\, \vec{x}_t$ and the hyperplane is represented by

$$\vec{w}.\vec{x} + b = 0$$

Where $\vec{w}.\vec{x}$ is the dot product of $\vec{w}.\, and\ \vec{x}$. Substitution of $\vec{w}_o$ in (54) into $\vec{w}.\vec{x} + b = 0$ gives

$$\textstyle\sum_{t=1}^{T} \alpha_t y_t \vec{x}_t.\vec{x} + b \tag{59}$$

Where $\vec{x}_t.\vec{x}$ is the dot product of $\vec{x}_t\ and\ \vec{x}$. $\vec{x}_t$ is the training vector, $\vec{x}$ is the testing vector, α_t is the Lagrange multiplier, b is the bias, and y_t is the input vector label. In a linear SVM, the SVM is characterized by the dot product $< \vec{x}_t\ \vec{x}_j >$. In non-linear SVMs, the kernel function achieves the transformation of data from input space to feature space by replacing the dot product, $\vec{x}_t, \vec{x}_j$ in (57) and (59) by $\emptyset(\vec{x}_t)\emptyset(\vec{x}_j)$ [10] . The dual objective function is now represented as [10]

$$\sum_{t=1}^{T} \alpha_i - 1/2 \sum_{t=1}^{T}\sum_{j=1}^{T} \alpha_t\, \alpha_j y_t y_j K(\vec{x}_t, \vec{x}_j) \tag{60}$$

$$\text{Subject to} \sum \alpha_t y_t = 0 \quad and\ \ 0 \leq\ \alpha_t\ \leq C \tag{61}$$

And the SVM can now be represented as $\sum_{t=1}^{N} \alpha_t y_t\, K(\vec{x}_t, \vec{x}) + b$ (62)

The kernel makes sure that we do not explicitly transform the data to the feature space. The most commonly used kernels are the polynomial and radial basis function (RBF) kernels and they are expressed as shown below.

Polynomial: $K(\vec{x}_t, \vec{x}_j) = (1 + \vec{x}_t.\vec{x}_j)^n$ (63)

Where $\vec{x}_t.\vec{x}_j$ is the dot product of $\vec{x}_t$ and $\vec{x}_j$. *n* is the order of the polynomial kernel

RBF: $K(\vec{x}_t, \vec{x}_j) = \exp(\frac{||1+\vec{x}_t.\vec{x}_j||^2}{2\sigma^2})$ (64)

σ is the width of the RBF, $\vec{x}_t$ is the training vector and $\vec{x}_j$ is the testing vector.

Where t and j are the indices of the training and testing vectors respectively.

There are several other kernels that have been used in different applications and the matching of a kernel to a problem would have to consider the characteristics of the data's input space. For a kernel to be formulated for a particular application, it has to meet a set of requirements known as the Mercer's conditions. [16]

Having explored an overview of statistical learning and SVMs the stage is now set to see how SVM is applied in speaker identification systems. The kernel used in this thesis was formulated from the GMM supervector linear kernel in [9] and the polynomial kernel as shown in the next subsection.

5.3 Speaker Identification System using SVMs

Speaker identification systems have been discussed in the last 3 chapters. In those chapters, the implementation of speaker identification systems using statistical methods was explored. This method creates or trains statistical models for speakers using GMMs and a test utterance from an unknown speaker is compared to the models to find out which one has the closest match. The comparison is done through Maximum Likelihood or Log Likelihood Ratio. By extension, using SVMs for speaker identification systems has a generic similarity. The process also consists of feature extraction, training and testing.

SVMs have enjoyed tremendous success in speaker identification system implementations. Since an SVM is as good as the kernel used, the kernel function is the most important part of the SVM with regards to its functionality. Researchers have worked tirelessly to come up with kernels that continue to achieve better performances in terms of lower false alarms and missed detection probabilities. In [16], a special type of the polynomial kernel was used for speaker identification and verification. The kernel used was called the normalized polynomial kernel. The well-known polynomial kernel was normalized to prevent the generation of a badly conditioned Hessian to prevent a breakdown of the quadratic programming optimization. The kernel reportedly achieved impressive results comparable to that achieved on the YOHO database. [17] introduced a kernel called the Log-likelihood Ratio (LLR) based sequence kernel. In that work, the kernel was composed with the Mercer condition in mind. The sequence kernel was represented by the dot product of 2 computable quantities instead of the dot product of 2 incomputable functions $\emptyset(.)$**.** The quantity used was the LLR using the Universal Background Model (UBM). So the kernel was a dot product between the LLRs of the training and testing data. [17] also formed another LLR based sequence kernel by combining the LLR UBM kernel and a LLR T-norm kernel. Both kernel functions were added to form a new kernel they called the composite kernel function [17]. These kernels performed very well but the composite outperformed its 2 component kernels. The Kullback-Leibler (KL) divergence measure was used to develop a kernel in [9]. The GMM supervector linear kernel [9] was used to bind the KL divergence measure between GMMs. 2 GMMs $\boldsymbol{\lambda_a}$ and $\boldsymbol{\lambda_b}$ were adapted from a UBM to form supervectors, $\overrightarrow{sv}_{\boldsymbol{a}}, \overrightarrow{sv}_{\boldsymbol{b}}$**,** respectively where , $\overrightarrow{\boldsymbol{sv}} = [\vec{\mu}_1{}^T, \vec{\mu}_2{}^T, \dots \vec{\mu}_M{}^T]^{\boldsymbol{T}}$**.** The dimension of $\vec{\mu}_1{}^T$ is D=12 and the number of Gaussian components, M =512. Therefore the dimension of the supervector $\overrightarrow{\boldsymbol{sv}}$ is D*M (12*512).The GMM supervector linear kernel is defined as follows.

$$K(\overrightarrow{sv}_{\boldsymbol{a}}, \overrightarrow{sv}_{\boldsymbol{b}}) = \sum_{i=1}^{M} w_i (\vec{\mu}_i{}^a)^T \boldsymbol{\Sigma}_i{}^{-1} (\vec{\mu}_i{}^b) \tag{65}$$

Where $\overrightarrow{sv}_{\boldsymbol{a}}$ represents the supervectors from speaker 'a' and $\overrightarrow{sv}_{\boldsymbol{b}}$ are the supervectors from speaker 'b'. $\vec{\mu}_i$ is the component mean vector, w_i is the component weight of the GMM UBM and $\boldsymbol{\Sigma}_i$ is the component diagonal covariance of the UBM. M is the number of components of the GMM. [9]

It should be noted that the kernel covers just one observation of data. That means that each observation of the train data (and test data) is a supervector. The technique implicitly maps an adapted speaker model to a supervector, making it suitable for a kernel. So the kernel is linear with respect to the supervectors. To implement this kernel, there has to be enough utterances that will each be adapted to a GMM model. Each model maps to a supervector and each supervector represents an observation in the SVM. The concept of the GMM supervector used in [9] will also be used in this thesis.

This thesis also contributes to knowledge by designing a new kernel from the idea of the GMM supervector linear kernel in [9] and the randomization of insufficient data to obtain a superfluous amount of data for testing and training the SVM. These will be discussed in the following subsections including SVM training and testing.

5.3.1 New Kernel

The GMM supervector linear kernel [9] is expressed as shown below

$$K(\overrightarrow{sv}_{\boldsymbol{a}}, \overrightarrow{sv}_{\boldsymbol{b}}) = \sum_{i=1}^{M} w_i (\vec{\mu}_i{}^{a})^T \boldsymbol{\Sigma}_i{}^{-1} (\vec{\mu}_i{}^{b}) \qquad (66) \quad [9]$$

Where $\overrightarrow{sv}_{\boldsymbol{a}}$ represents the supervectors from speaker 'a' and $\overrightarrow{sv}_{\boldsymbol{b}}$ are the supervectors from speaker 'b'. $\vec{\mu}_i$ is the component mean vector, w_i is the component weight of the GMM UBM and $\boldsymbol{\Sigma}_i$ is the component diagonal covariance of the UBM. M is the number of components of the GMM. [9]

And the polynomial kernel is expressed as

$$K(\overrightarrow{sv}_{\boldsymbol{a}}, \overrightarrow{sv}_{\boldsymbol{b}}) = (1 + [\overrightarrow{sv}_{\boldsymbol{a}}]^T \overrightarrow{sv}_{\boldsymbol{b}})^n \qquad (67)$$

n is the order of the polynomial kernel

The value of the polynomial order was varied from 2 to 20 in the experimental implementation. It was discovered that when using the polynomial kernel, a polynomial order of 3 gave the best result. Performance of the kernel in [9] and the polynomial kernel were compared and the polynomial kernel was out-performed. A look at both kernels reveals that they have subtle similarities. The GMM supervector liner kernel looks like a polynomial kernel of unity order which is scaled with variances and its prior probability.

New kernels have been formulated by the addition or multiplication of existing kernels. The idea conceived in this thesis was to combine the characteristics of the 2 kernels above as follows.

$$K(\overrightarrow{sv}_{\boldsymbol{a}}, \overrightarrow{sv}_{\boldsymbol{b}}) = \sum_{i=1}^{M} [1 + w_i (\vec{\mu}_i{}^{a})^T \boldsymbol{\Sigma}_i{}^{-1} (\vec{\mu}_i{}^{b})]^n \qquad (68)$$

The new kernel produced exciting results which will be reviewed in the next chapter. As far as the new kernel is, it is valid, because it meets the Mercer's conditions. Its Hermitian is positive semi-definite and the kernel is a sum of the products of the test and training features.

5.3.2 Randomization

A method of randomization was used to achieve the SVM in this thesis. During these experiments, there was a dearth of speaker speech signals. For each speaker, only about 3mins of conversation was available per phase (training and testing). The GMM supervector method utilized required that enough different speech signals be available from each of the speakers. The SVM using the method in [9] takes in supervectors as inputs, and a supervector is the concatenation of the component means of the adapted GMM of a speaker from one conversation (about 90 sec long). So if the training of each speaker would be we done, there has to be many observations available to the SVM for adequate training of models. These observations correspond to several adapted GMM models from several utterances from each speaker. This method therefore allowed the creation of multiple samples from the 3 min sample available. The approximately 3 min (180 sec) long speech signals per speaker were sliced into 2-sec blocks. 45 2-sec slices of speech signals were then picked at random and concatenated to form a speech utterance that would appear unique to the system. By using this method, it was possible to generate as much unique speech utterances as needed to model the SVM speaker models. This same method was also used to generate multiple test observations for SVM testing.

All simulations for this experiment were carried out within Matlab. The statistical toolbox was used for the EM algorithm implementation and for SVM testing and training.

5.3.3 SVM Training

The training of an SVM model requires using input feature vectors that are representative of the 2 classes - a positive class and a negative class. If we have 4 speakers for example, and we want to train a model for speaker 1, the train feature vectors will consist of feature vectors from speaker 1 and feature vectors from a combination of speakers 2-4. Features vectors from speaker 1 will serve as the positive class and those from 2 – 4 will be the negative class. If speaker 2 needs to be trained, the positive class will contain speaker 2 and the negative class will have speech from the other speakers (1,3 and 4). This indicates that the number of training models we have will be determined by the number of speakers available in our database – 1 SVM model per speaker. In addition, the amount of input data should be sufficient to aid speedy detection but should not be excessive because this can give rise to a large number of support vectors which can increase storage requirements and computation time. [16]. When an SVM is trained, what we are actually calculating are the $\vec{w}$ and b in the equation below

$$\vec{w}_s . \overrightarrow{sv} + b_s \qquad s = 1,2, \dots S, \tag{69}$$

$S = number\ of\ speakers$ and Where $\vec{w} . \overrightarrow{sv}$ is the dot product of $\vec{w}$ and supervector $\overrightarrow{sv}$.

For each training model, we have a set of $\vec{w}_s$ and b_s.

A problem that might result due to poor training is when one of the speakers is not represented in the imposter speakers' features. A misclassification of this speaker might result if this speaker feature vectors are tested using such a poorly trained model. [16]

5.3.4 SVM Testing

During testing of speaker feature vectors, the test features are compared to all the training models to obtain the closest match. At this stage, the SVM can either output a class label in terms of the formula

$$sign(\vec{w}.\vec{sv} + b) \tag{70}$$

Where $\vec{w}.\vec{sv}$ is the dot product of $\vec{w}$ and supervector $\vec{sv}$.

Or the output can be in the form of a score as shown below.

$$Score = \vec{w}.\vec{sv} + b \tag{71}$$

So if a set of T test supervectors are tested, the score can be averaged over all T supervectors as below

$$Score = \frac{1}{N} * \sum_{j=1}^{N} \vec{w}.\vec{sv}_j + b \tag{72}$$

If $w_{o=}\ \sum \alpha_t y_t \vec{sv}_t$ is substituted in the equation, we get

$$Score = \frac{1}{N} * \sum_{t=1}^{T} \sum_{j=1}^{N} \alpha_t y_t \vec{sv}_t . \vec{sv}_j + b \tag{73}$$ [16]

Where $\vec{sv}_j . \vec{sv}_t$ is the dot product of $\vec{sv}_j$ and $\vec{sv}_t$, T is the total number of support vectors obtained from the training data $\vec{sv}_t$, N is the number of test supervectors $\vec{sv}_j$, y is the assigned class label, α is the Lagrange multiplier of the support vector and *b* is the bias.

This score obtained can be measured against a set threshold so if the score exceeds the threshold, that speaker is accepted and rejected when it is below the threshold. This type of scoring can be used to obtain a Detection Error Tradeoff (DET) curve. By varying the threshold, the rate of errors are varied. These errors are the probabilities of false detection or false alarm (FA) and the probability of missed detection. The obtained errors can then be used to characterize how accurate the speaker identification system is. An alternative curve is the Receiver Operator Characteristics (ROC) which plots the probability of detection against the probability of false detection. Either of the 2 plots can be used depending on the type of result we want the plot to convey. To make a decision between a set of S speakers, we choose the training model that generates the highest testing score. The identity of that model is given to the testing data.

$$\arg\underset{s}{max} \frac{1}{N} * \sum_{j=1}^{N} \sum_{t=1}^{T} \alpha_{st} y_{st} \, \overrightarrow{sv}_{st} . \overrightarrow{sv}_{j} + b \qquad\qquad 1 \leq s \leq S \tag{74}$$

$where\ S\ is\ the\ number\ of\ speakers\ in\ the\ database$ *T* is the number of training supervectors and *N* is the total number of testing supervectors.

Chapter 6: Experimental Design and Results

6.1 Experimental Design

With the discussion of all the essential components of this thesis work, we now discuss the specifics of the parameters used for this experiment.

6.1.1 Speech Corpus

The speech database used for this experiment is the CHAINS speech corpus. It contains 36 speakers with recordings in 2 different sessions of 2 months apart. There were 6 speaking styles during the 2 sessions- 3 different styles in each speaking session. Each speaker read a series of passages in 6 different styles. 3 of the styles came after 2 months of reading the first 3 styles. There were 36 speakers, 20 males and 16 females [18]. For the experiment, 16 males and 16 females were selected in order to have an unbiased UBM [7]. 1 speaking style from the 1st set of 3 was chosen (called SOLO) and the speakers read the passages at conversational pace. This set of audio recordings was used for testing. Another speaking style (called FAST) was taken from the second set of 3 styles and the speakers read the passage at a much faster rate. This set of speech signals were used for training purposes. All the recordings for each speaker in each of the speaking sessions (e.g. SOLO) was pooled. For each speaker, we had a recording for training and a recording for testing.

6.1.2 Front End Processing

This step involves extracting the speech features vectors from the speech signal. Audio signals were discretized using the sampling frequency of the recording. For the audio samples used in this thesis, the sampling frequency was 44.1 KHz. The audio signal was framed in 20ms chunks and they overlap by 10msec.

1 sec → 44100 samples

0.02 sec → 0.02*44100 = 882 samples per frame.

The number of FFT samples used was 512 using the hamming window for windowing. Only the first 257 samples of the FFT are kept. 26 triangular Mel frequency filters are generated with their frequencies ranging between 0 and 22050 Hz (44100*0.5 Hz). The filterbank is a 26 X 257 matrix with each 257 vector representing one filter. After obtaining the DCT of the log filterbank energies (26 coefficients), the 1st coefficient was dropped and coefficients 2 – 13 were kept. Therefore, for each 20msec frame of audio signal, we obtain a [1 X 12] feature vector.

6.1.3 GMM Adaptation

The extracted feature vectors are ready for modeling at this stage. Before we go ahead with that, a UBM will have to be created which will serve as the alternate speaker if LLR testing is used and from which the individual speaker GMM models are adapted. Studies in [5] show that higher number of GMM components improve the speaker identification system performance. In this thesis, 512 components were used to model the GMM UBM and all resulting models. Therefore, a GMM in this thesis will have the following dimensions.

$$\lambda=\{w_i, \vec{\mu}_i, \boldsymbol{\Sigma}_i\} \quad i = 1,2,\dots,M$$

$$w = \{1\ x\ 512\}, \qquad \mu = \{12\ x\ 512\}, \qquad \Sigma = \{12\ x\ 12\ x512\}$$

Where M =512 and is the number of component distributions and 12 is the dimension of the feature vectors. When GMMs were adapted, only the means were adapted and a relevance factor of 16 was used.

6.1.4 GMM Supervector

A GMM supervector in this thesis is the stacking up of the component mean vectors of an adapted speaker model to form a single larger vector. The size of a single supervector is

$$[1\ x\ (12 * 512)] = [1\ x\ 6144]$$

Where 12 is the dimension of the feature vector and 512 is the number of components of the GMM.

6.1.5 Support Vector Machine

SVM was trained with data from 2 classes, the target speaker and the imposter speaker. The training feature vectors consisted of:

100 supervectors from the target speaker

155 supervectors from the 31 other speakers (5 supervectors from each speaker).

255 supervectors generated 1 speaker model.

The effectiveness of an SVM implementation can be measured by the potency of the kernel used. The kernel used for this thesis was newly conceived from the idea used in [9], it gave improved results over the GMM supervector linear kernel in [9]. In this thesis the idea of the polynomial kernel and the GMM supervector linear kernel used in [9] are combined to obtain the new kernel presented below.

$$K(\overrightarrow{sv}_{\boldsymbol{a}}, \overrightarrow{sv}_{\boldsymbol{b}}) = \sum_{i}^{M} [1 + w_i (\vec{\mu}_i{}^{a})^T \boldsymbol{\Sigma}_i{}^{-1} (\vec{\mu}_i{}^{b})]^n \tag{75}$$

Results obtained in this thesis show that the optimum performance was achieved when a polynomial order of 6 was used. The results are presented later in this chapter.

6.2 Results

32 (16 male and 16 female) of the 36 speakers in the CHAINS corpus [18] were selected for the experiments due to the explanation offered in [7]. The speech signals used for modeling the Universal background model (UBM) was balanced in terms of the representation of the sub-groups contained in the corpora [7]. Therefore, 16 males and 16 females were used for this work. Each speaker had 33 conversations [18]. The 33 conversations are short, so the features from them were concatenated to give a feature vector set of approximately 3 minutes long. UBM was formed by pooling features from all

speakers and modeling. The GMM generated has 512 components and the individual speaker models for training and testing were adapted from the UBM. The result computation process is summarized in a flow chart shown in Figure 6.1.

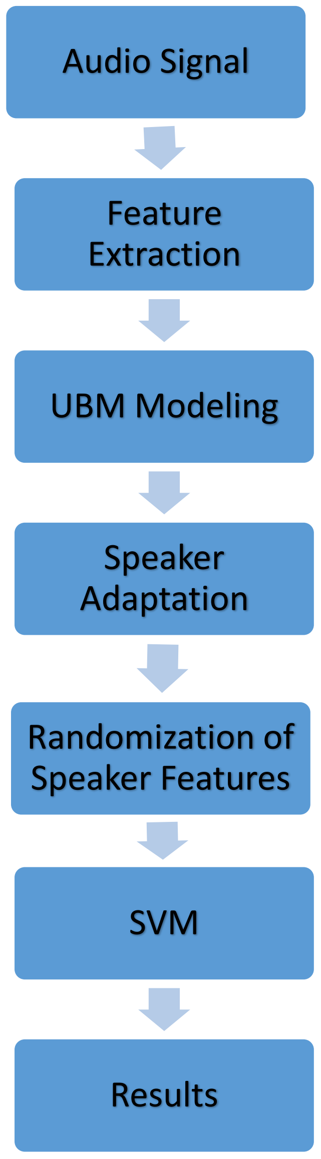

Figure 6.1: Flow chart of result computation

As discussed in the previous paragraph, the amount of speech available in the CHAINS speech database [18] is approximately 3 minutes for each speaker per speaking style. This available speech was used to model the speakers.

The SVM method in use in this thesis makes use of the supervector technique as described in [9]. This technique uses the GMM supervectors as a higher dimensional feature space for the SVM kernel functions. By extension, that makes each observation of the training and testing data a supervector. Therefore, a short 3-minute utterance gives one supervector. The SVM requires multiple supervectors for training and testing. This presented a challenge which had to be solved to avoid a problem of under-training during SVM model creation.

A solution to the challenge was obtained by using the normalization method explained in section 5.3.2. A number of feature vectors were randomly selected, concatenated, and the results would be a unique speech feature vector set as far as the speaker identification system is concerned. This method can be used to generate $^{90}_{n}C$ unique feature vectors and by extension $^{90}_{n}C$ supervectors per speaker, per speaking style. Where n is the number of 2 second mini-blocks. For this thesis, 45 2 second mini-blocks were selected randomly and concatenated to give a 90 second feature vector set. Then this selection was repeated 100 times to give 100 feature vector sets per speaker, per speaking style. The 100 feature vector sets were adapted to the UBM to obtain 100 supervectors per speaker, per speaking style. Since 2 speaking styles were chosen for this thesis (SOLO and FAST [18]), each speaker had 200 supervectors. The first 100 was used for testing and the other 100 for training.

SVM was trained using the polynomial kernel, the GMM supervector linear kernel [9] and the new kernel formulated in this thesis. Training was carried out using 100 supervectors from a target speaker and 155 supervectors from the imposter speakers. All 255 supervectors were used to train the SVM while assigning the appropriate class labels to each of the supervectors.

Testing was done by comparing 100 testing supervectors to each of the speaker classifiers to obtain the closest match. So this set produced 3200 true trials and 99200 false trials. A score was obtained from each testing of a supervector. The output from the SVM testing is in form of a numerical score. The magnitude of the score depends on the level of similarity between the training and testing data. For a test supervector whose speaker is the target speaker, the score has a lower magnitude compared to the score of a non-target speaker. The identity of a speaker can therefore be obtained by taking the average of the scores of N test supervectors. For N test observations, all N observations where compared to the 32 speaker SVM models. The scores from each speaker model were averaged and the identity of the speaker with the minimum score was assigned to that set of test supervectors.

In order to view how accurate the scores obtained from the SVM testing is, the entire target scores for each SVM classifier for all 32 speakers were pooled. All the non-target scores from all SVM classifiers were also pooled. A set of thresholds were set to obtain varying values of false alarm and missed detection probability. The missed detection and false alarm probabilities were plotted for the polynomial kernel, the GMM supervector linear kernel, and the new kernel as shown in figure 6.2 for the 32 speaker database. Figure 6.3 is the result of a 16 speaker implementation. Subsequent results obtained are also shown in the figures 6.4 to 6.10. These results are the DET curves for the new kernel with the order of the polynomial componenet varied from 3-10 and 15. The wisdom behind these results are discused in the next section.

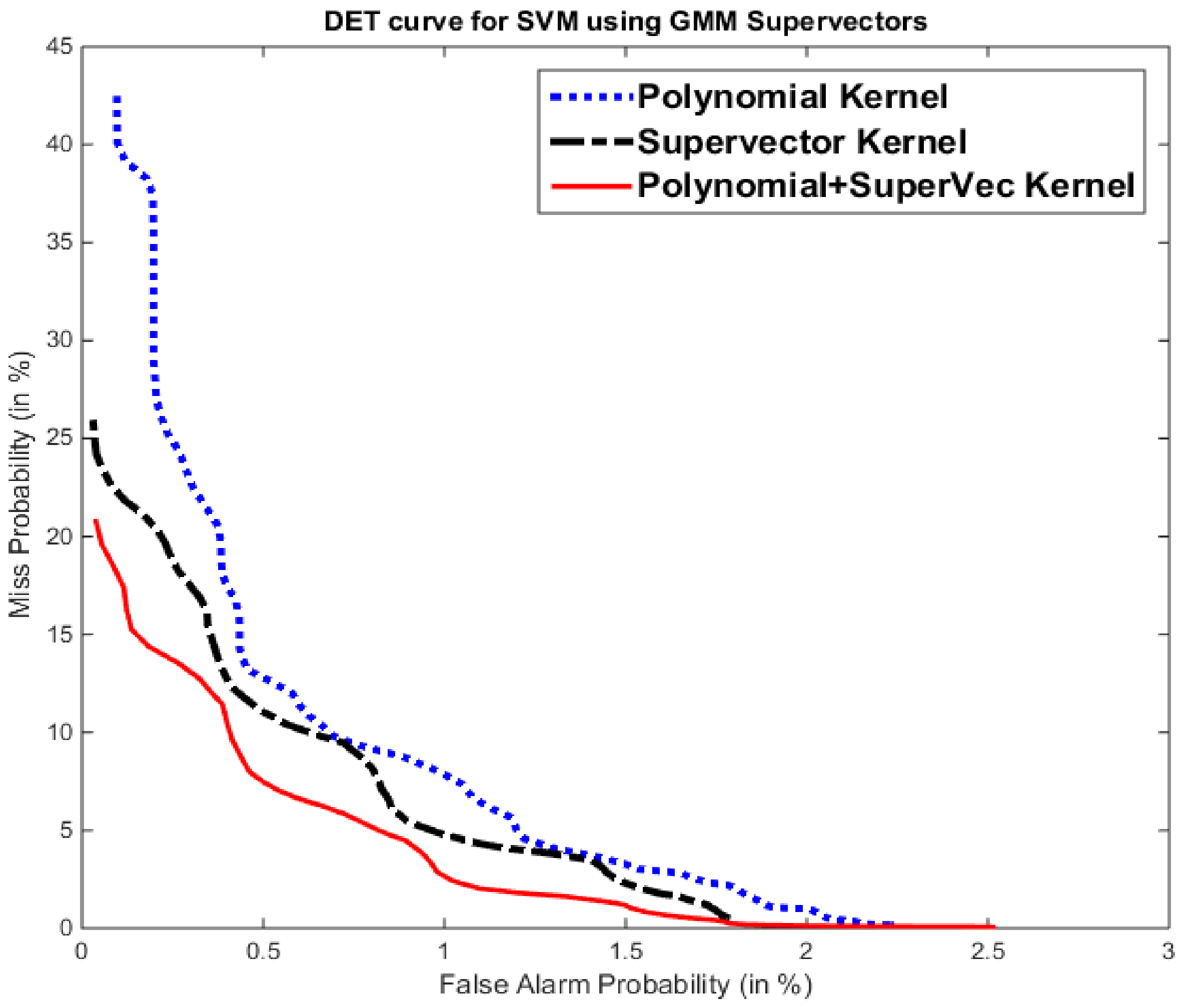

Figure 6.2: DET curves for 32-speaker speaker identification system

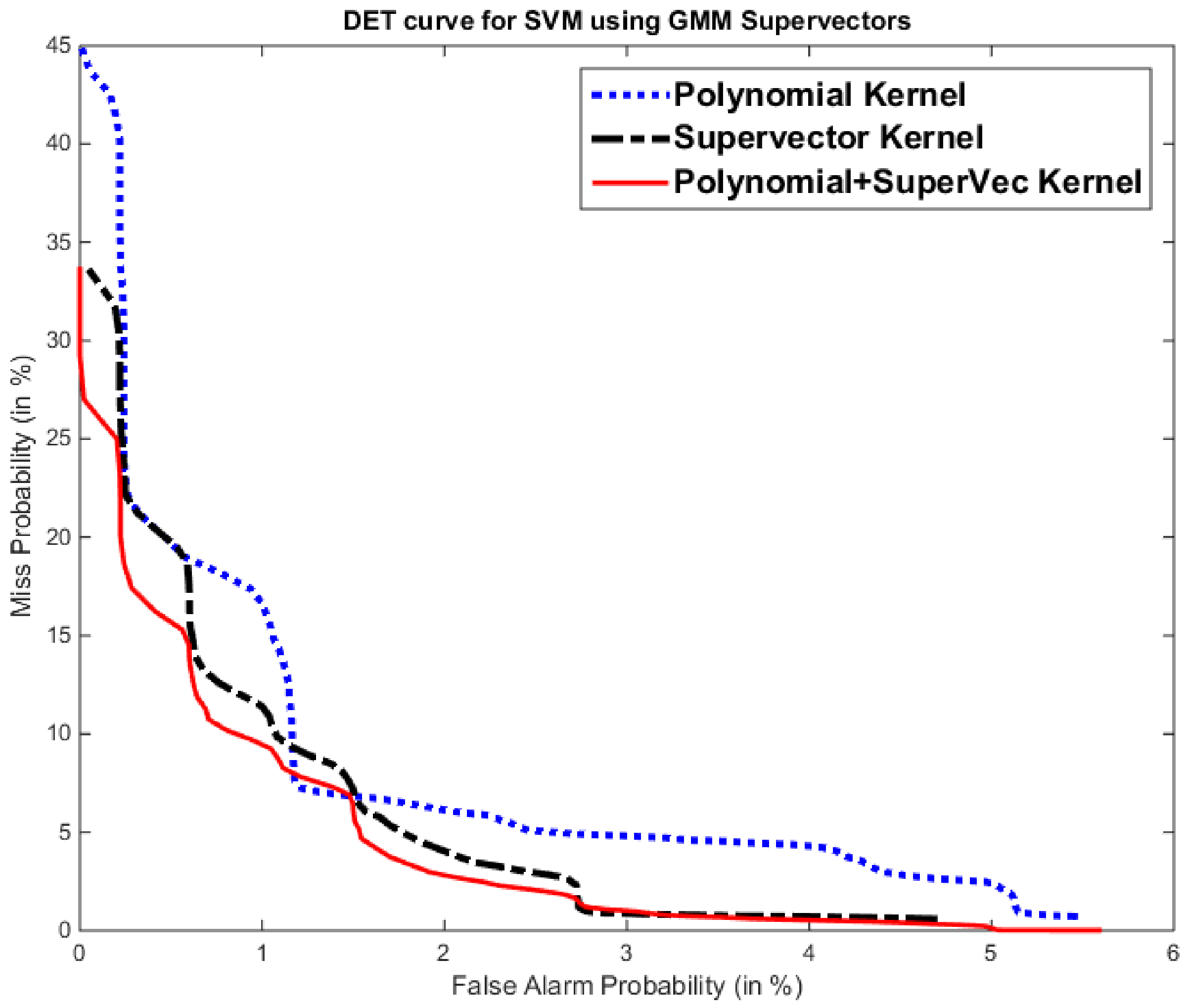

Figure 6.3: DET curves for 16-speaker speaker identification system

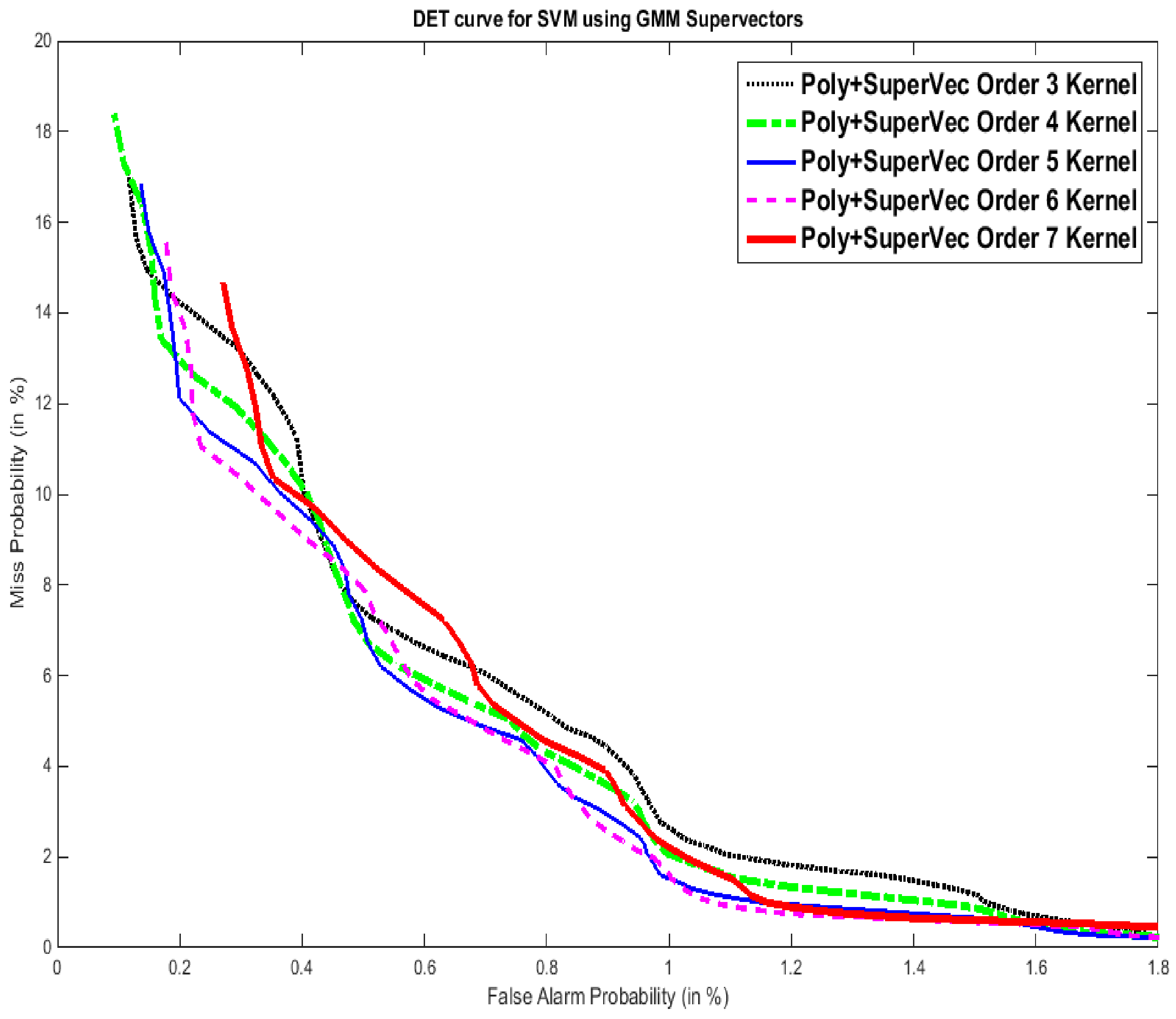

Figure 6.4: DET curves for new kernel with polynomial order of 3 to 7

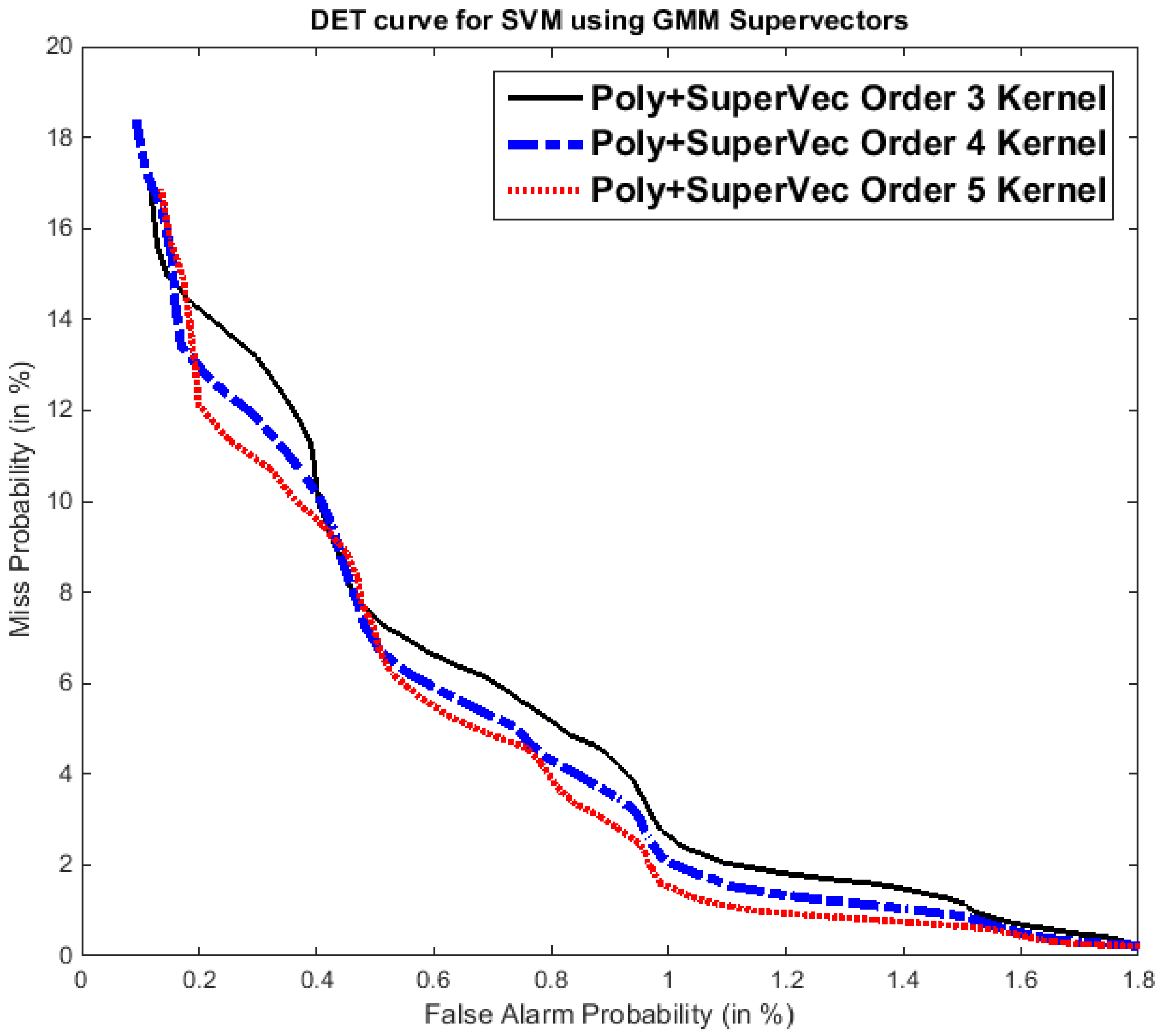

Figure 6.5: DET curves for new kernel with polynomial order of 3 to 5

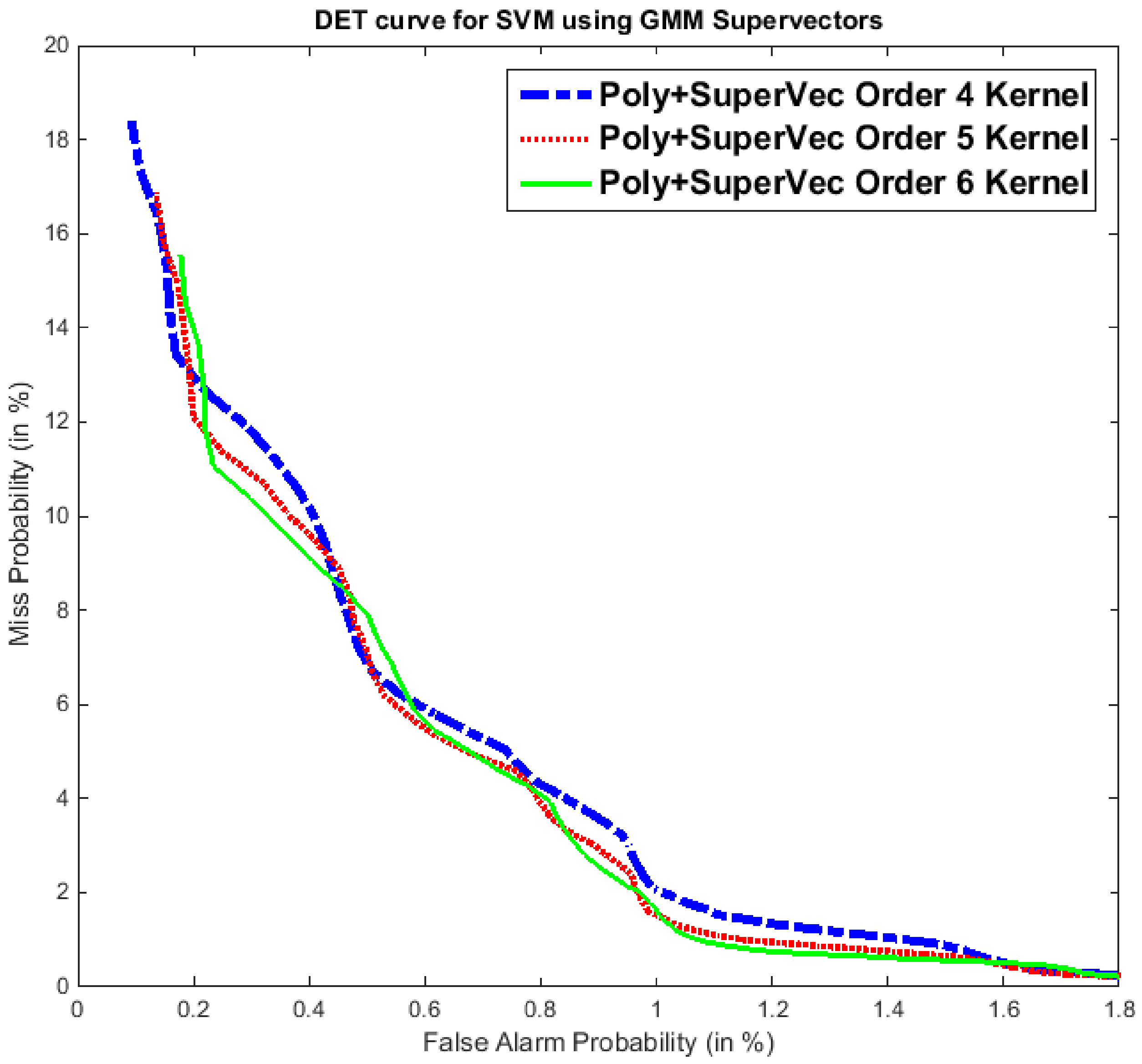

Figure 6.6: DET curves for new kernel with polynomial order of 4 to 6

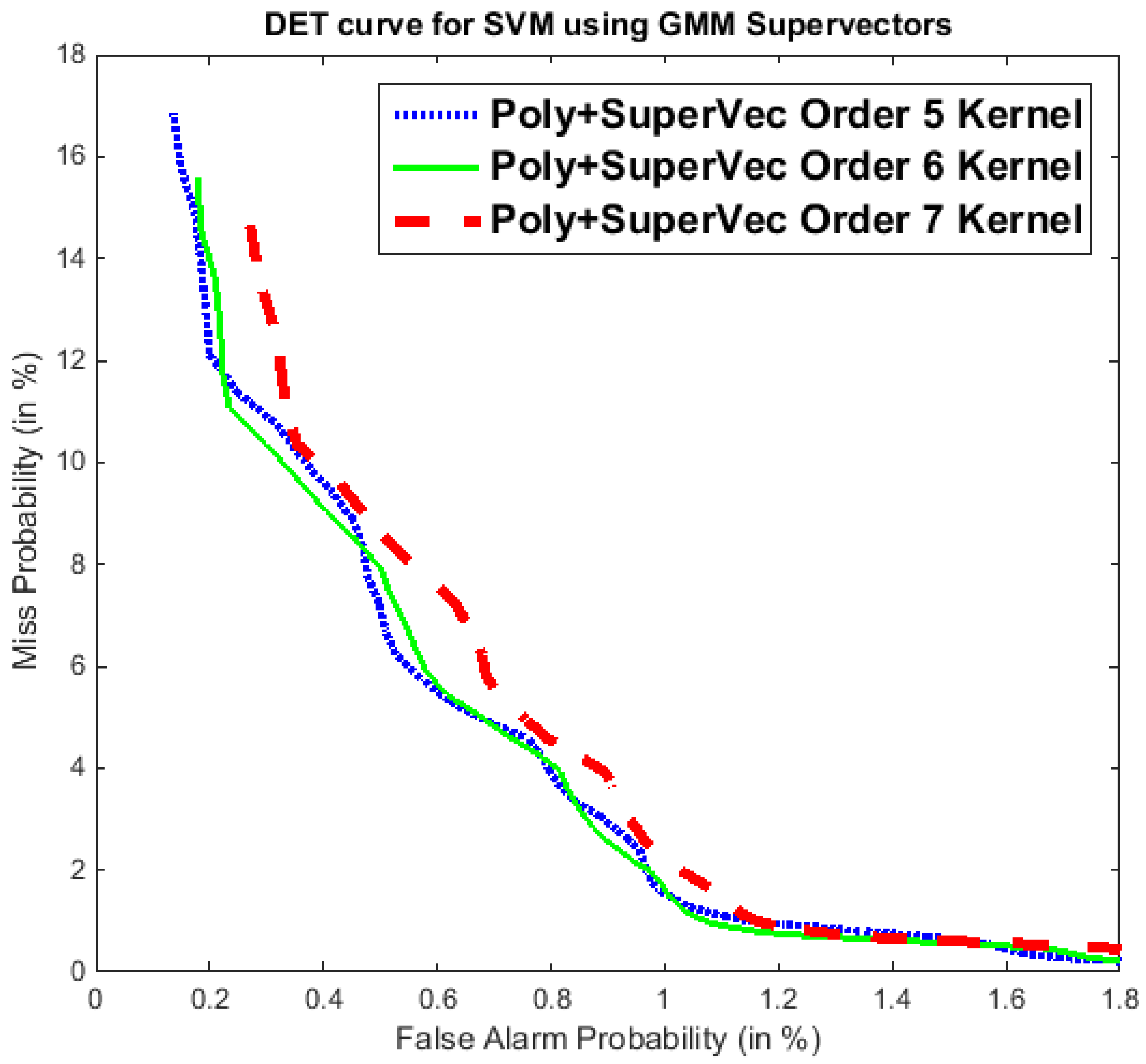

Figure 6.7: DET curves for new kernel with polynomial order of 5 to 7

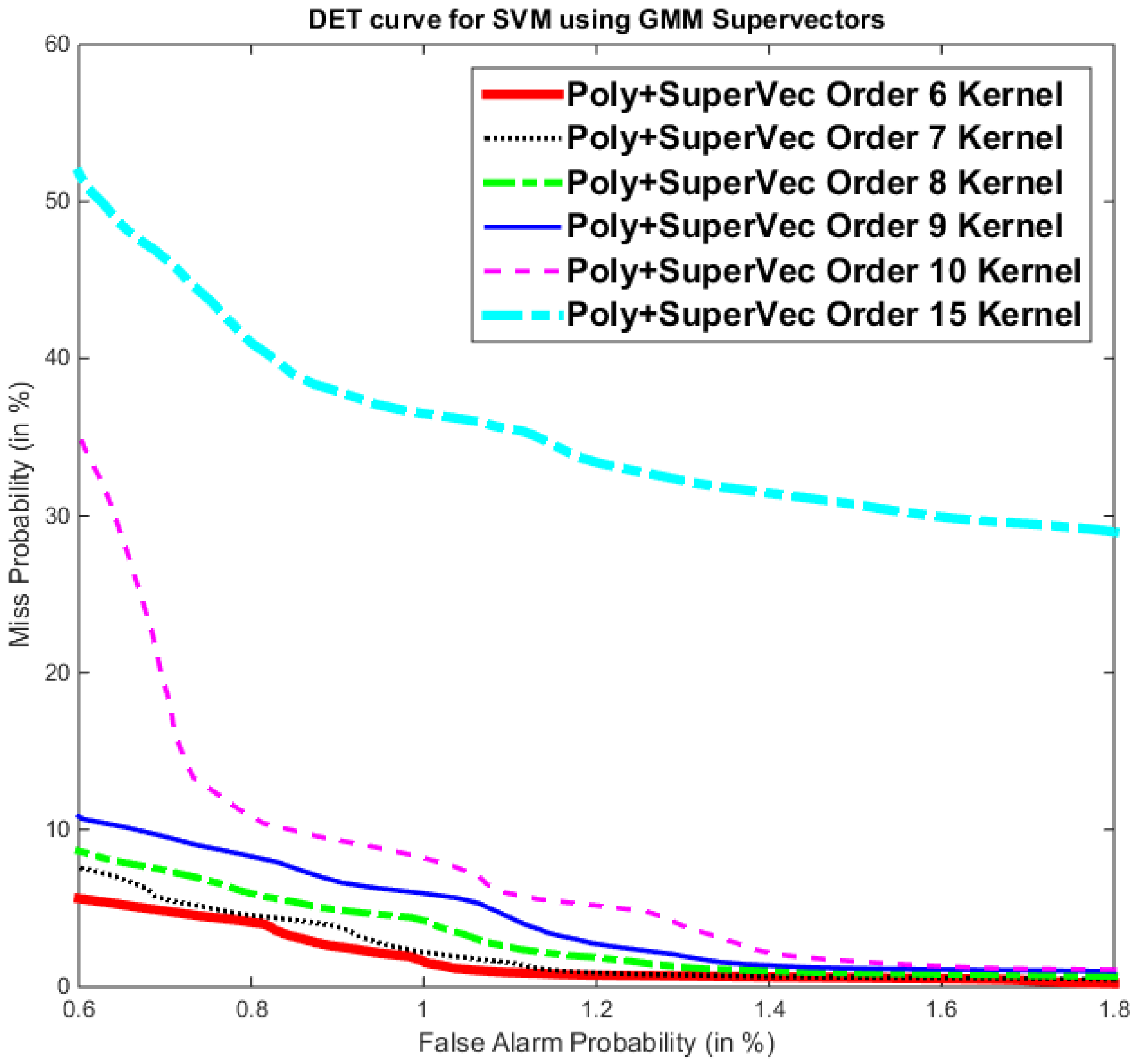

Figure 6.8: DET curves for new kernel with polynomial order of 6 to 15

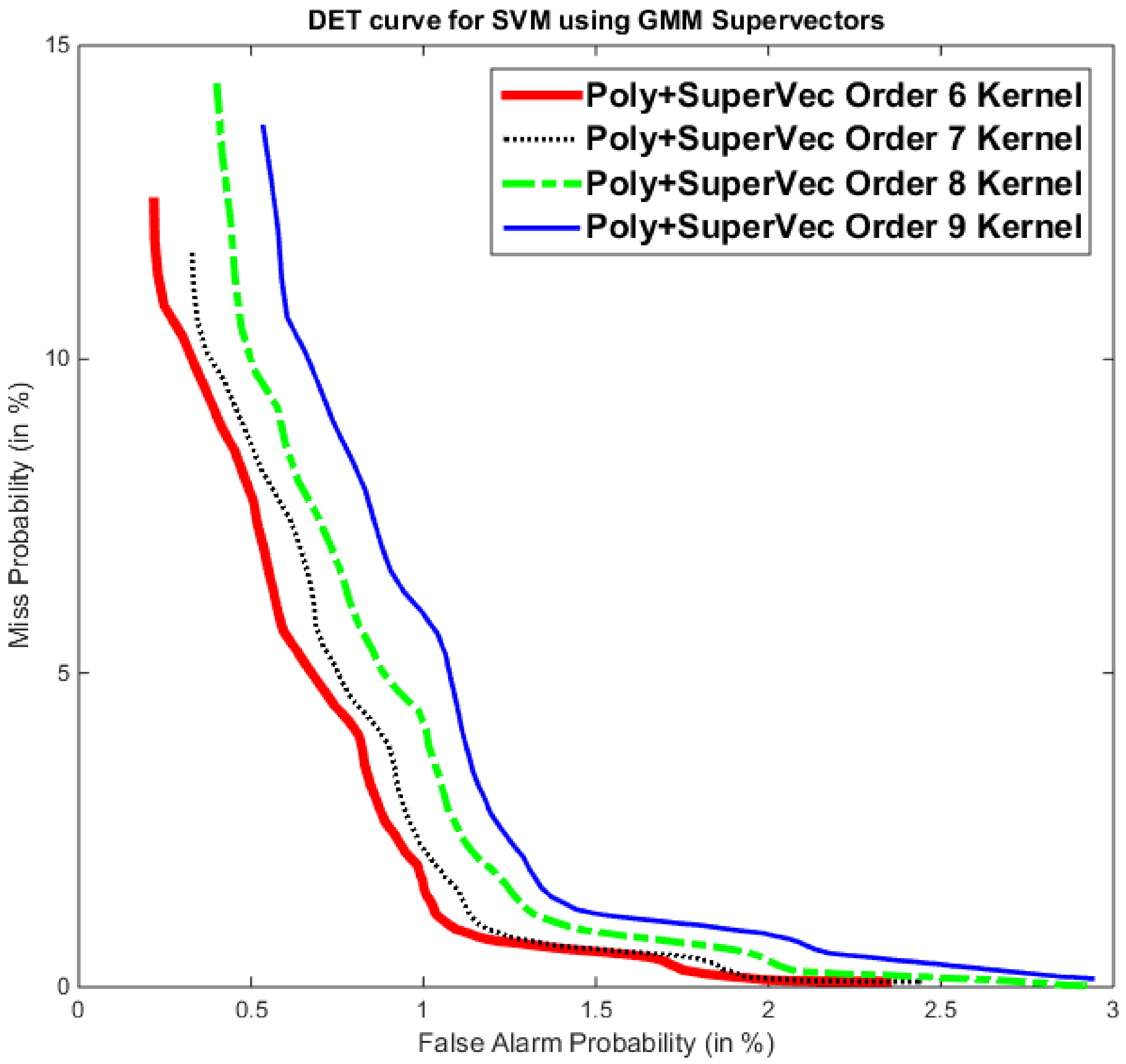

Figure 6.9: DET curves for new kernel with polynomial order of 6 to 9

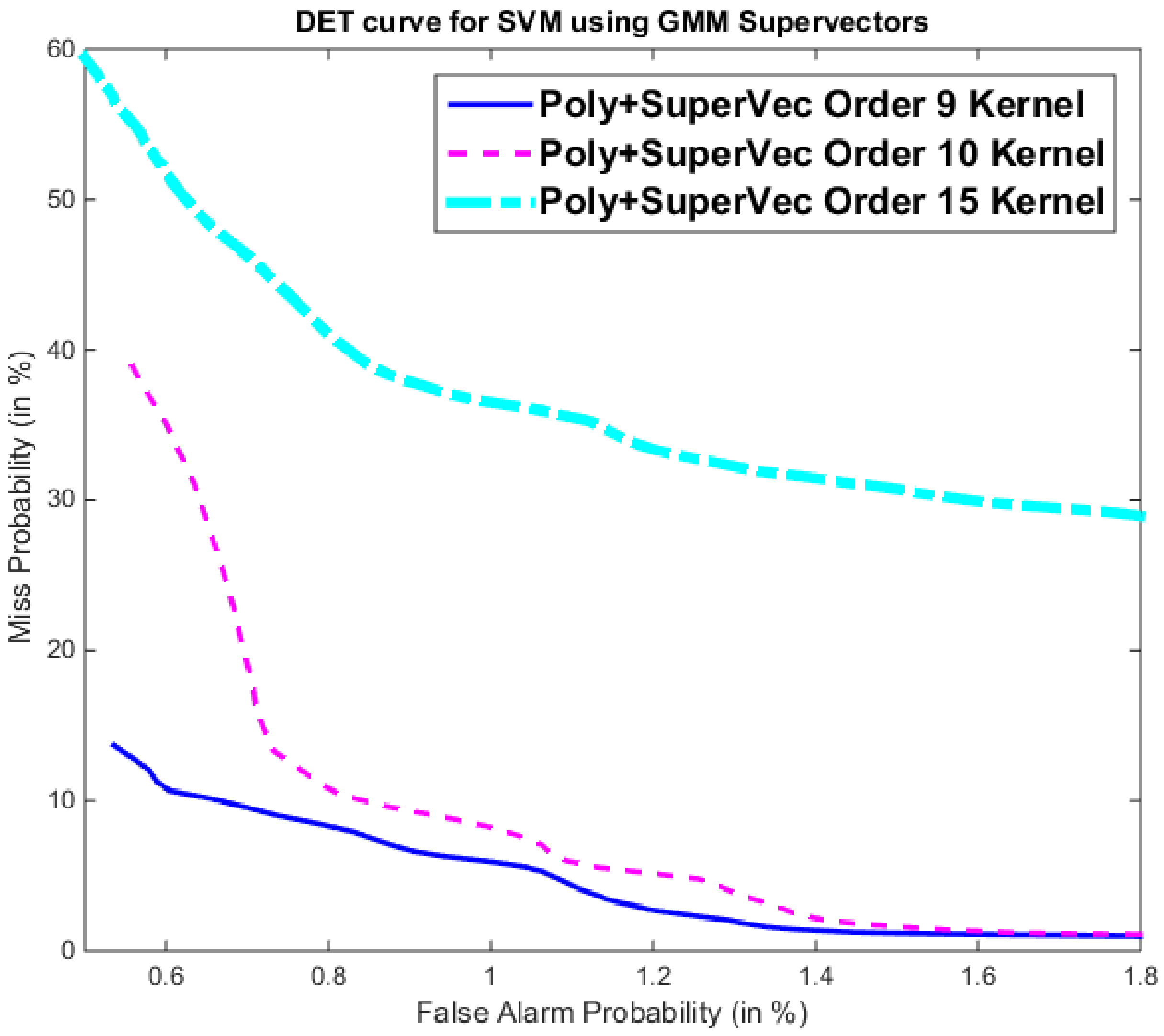

Figure 6.10: DET curves for new kernel with polynomial orders of 9, 10 and 15

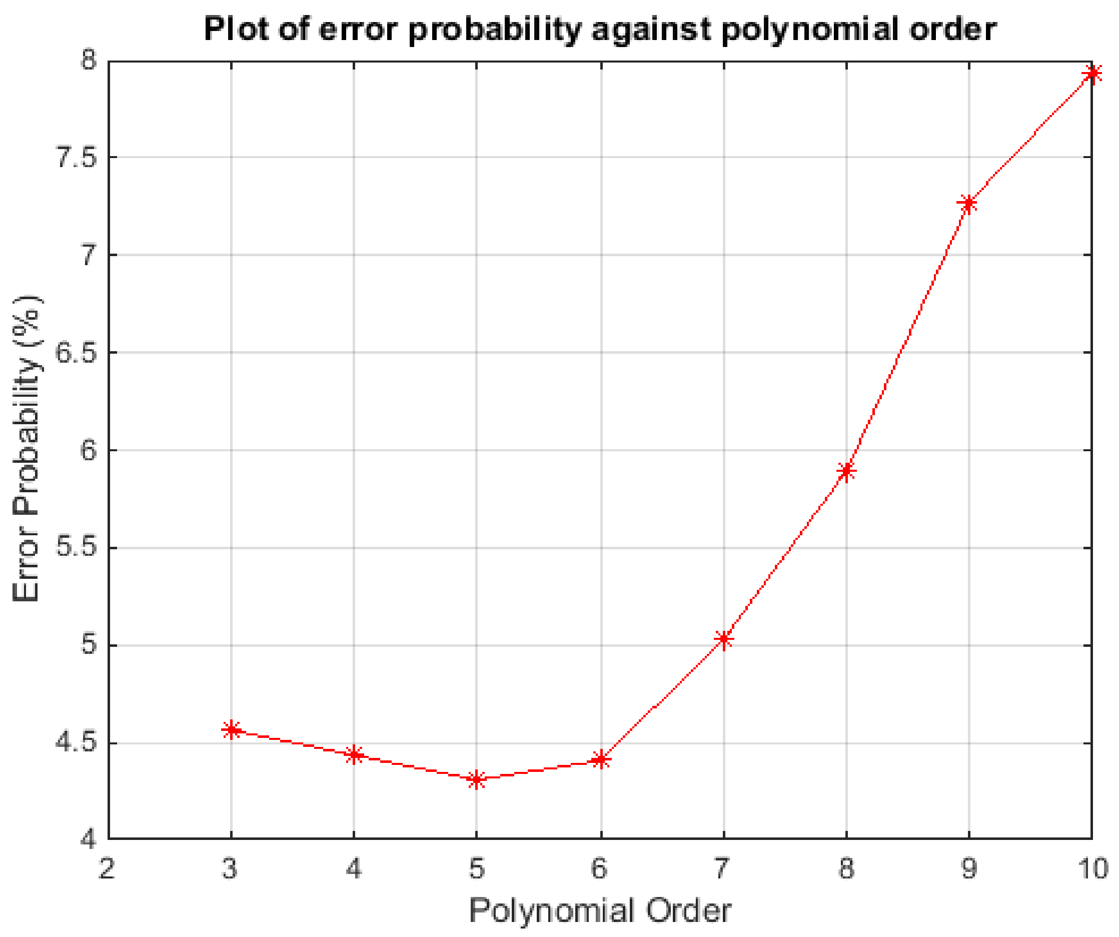

Figure 6.11: Plot of speaker detection error probability as a function of polynomial order

6.3 Discussion of Results

Figure 6.2 and Figure 6.3 show the decision estimation tradeoff (DET) curves for the results obtained from the SVM. This results show the DET curves for SVMs using 3 different kernels. Figures 6.2 and 6.3 however, differ in the number of speakers in the database used in the operation. 32 speakers were used for figure 6.2 and other results and 16 speakers were used to obtain figure 6.3. The 2 figures show good performances but it can be observed that the missed detection probability in 6.3 is slightly better than in figure 6.2. This suggests a slight deterioration of the error performance with an increase in the number of speakers. The false alarm probability in figure 6.2 is a bit better than in figure 6.3 though. This might be due to the 32 speaker GMM UBM being a better representation of a global model than the 16 speaker GMM UBM is.

The polynomial of polynomial order 3 produced the best performance for polynomial orders between 2 and 20 but the polynomial kernel had the worst performance compared to the other kernels. The likely reason for this is even though the polynomial kernel is a well known valid kernel which in this implementation used the power of GMM supervectors, it is just a simple inner product of a training and test vector. It does not put the prior probabilities and the feature variances into consideration. These 2 parameters always have a way of offering a better representation of the prevalence of each data component. This suggest why the polynomial kernel is not the best kernel to use for SVMs in speaker identification systems.

The next kernel to be discussed is the GMM supervector linear kernel presented in [9]. This kernel was derived from the Kullback-Leibler divergence measure and it used the UBM and GMM supervectors. In [9], the performance of this kernel was juxtaposed with that of another and this kernel gave the better performance in terms of having the lowest error of the 4 methods compared [9]. The success of this kernel can be attributed to its derivation from the Kullback-Leibler divergence measure and it takes cognizance of the prior probabilities and variances of its data components. Each data is scaled by the square root of the variance and the prior probabilities [9]. As can be seen in figure 6.2, this method has produce a way better result than the polynomial kernel especially at thresholds that correspond to very low false alarm probabilities.

The intent for this thesis was to create a kernel whose performance will surpass that of existing ones. How to obtain it was the question left unanswered until it was. It is known in SVM that kernels must have characteristics that meet some requirements. Those requirements are the Mercer's conditions. New kernels have been created by the addition of 2 existing kernels as obtained in [17]. Results in this thesis were obtained from a kernel which was adapted to the characteristics of another kernel.

Figure 6.2 shows the new kernel which is the GMM supervector linear kernel [9] adapted to the characteristics of the polynomial kernel using a polynomial order of 3. This kernel produced a better performance and it goes to show that kernel functions can also be formed by the adaptation of one to another, provided the Mercer's conditions are met. The DET curves in figure 6.2 and figure 6.3 show that the new kernel outperforms the previous 2 others. This result looks promising for subsequent SVM applications of speaker identification systems because the characteristics of very good kernels can be adapted to one another to obtain more accurate systems.

Another striking observation from the result in figure 6.2 is the low probability of false alarm. It can be noticed from the plot that for the polynomial kernel which has the worst performance, the highest probability of false alarm is about 2.25%. For the probability of missed detection, the worst result (polynomial kernel) gave a probability as high as 42%. The interpretation of this is as follows. The probability of false alarm is the probability of deciding a non-target speaker is the actual speaker. That is, the probability that an imposter would get away with claiming someone else's identity. On the other hand, the probability of missed detection is the probability that the system decides the actual speaker is an imposter. The implementation of this result is very interesting because of the applications of speaker identification systems. If an over-the-phone bank transaction utilizes this identification system for example, this system has a very low probability of allowing an imposter to pass as the real owner of the account. This means the system has a high security level. It has a high probability of preventing a fraud from occurring. On the flip side though, this system has a much higher probability of missed detection which if used in an over-the-phone bank transaction, might screen the real users out sometimes. This might be more acceptable to users because they would rather sacrifice a few extra attempts than lose their hard earned money to fraudsters. This shows that the speaker identification system in this thesis has an overall high level of security.

As discussed in the beginning of this section, a polynomial order of 3 was used in the polynomial kernel because it gave the best accuracy for polynomial orders between 2 and 20. For the new kernel, a polynomial order of 3 was also used. Therefore, the curve for the new kernel in figure 6.2 has a polynomial order of 3. Experimental simulations were carried out to vary the polynomial order of the new kernel. This was done to ascertain if the order has any effect on the DET curves. Figure 6.4 to figure 6.10 show the results obtained for polynomial orders 3-10 and 15. The information obtained from observing the results are interesting.

It can be seen that on increasing the polynomial order from 3-6, the error performance of the system steadily got better. The difference in performance between polynomial orders 5 and 6 is not very pronounced as can be seen in figure 6.6 and figure 6.7. This means that for the new kernel, feature spaces of higher dimensions generated better results. As soon as the polynomial order reaches 7, however, it can be observed that there is a deterioration of the error performance as shown in figure 6.7. For polynomial orders of 7 and above, performance gets worse. It can then be inferred from figures 6.4 to 6.10 that the optimum polynomial order for the new kernel is either 5 or 6 as these are the curves with the least overall missed detection and false alarm probabilities. Figure 6.11 then shows the error probability plotted as a function of polynomial order. The error probability was obtained by counting the number of failed detections per polynomial order and dividing it by the total number of tests per polynomial order. The optimum polynomial order is 5 as seen in figure 6.11. The optimum new kernel obtained in this thesis is therefore

$$K(\overrightarrow{sv}_{\boldsymbol{a}}, \overrightarrow{sv}_{\boldsymbol{b}}) = \sum_{i}^{M} [1 + w_i ({\vec{\mu}_i}^{a})^T {\boldsymbol{\Sigma}_i}^{-1} ({\vec{\mu}_i}^{b})]^5 \tag{76}$$

Chapter 7: Conclusion and Future Work

7.1 Conclusion

With this thesis coming to a close, a step back will be taken to view the significance of the work done here. The main objective of this work was to create a speaker identification system using statistical learning but with a performance which surpasses that of previous works. The new kernel created in this thesis did just that and had a performance better that the kernel introduced in [9] and the most significant part is the way in which the kernel was obtained. This thesis was able to show that the performance of a kernel can be improved by adapting the kernel to the characteristics of another good-performance kernel. The new kernel achieved better results than the one in [9] and it shows this method actually works well. Other researchers can adopt this adaptation method to improve kernel performance and create new ones in addition to the method of addition and multiplication of existing kernels.

In addition, the method of randomly selecting pieces of available data to make several data frames was a success. The method can be utilized when the amount of training and testing data is insufficient and the method used (like the SVM) requires enough data to make better predictions of the identity of speakers.

Another significance of this work is the low false alarm probability obtained from the result. It goes to show that the approach used here (same as used in [9]) better suits a speaker identification system because of the seeming security it offers against imposters.

Lastly, the results obtained here are from a 16 speaker and 32 speaker corpus. It was observed that there was not a significant deterioration of the results between the 16 and 32 speaker models. It is therefore envisaged that as the number of speakers increase, there would not be a severe effect on the performance of the speaker identification system using the GMM SVM methods.

7.2 Future Work

Moving on from here, the hope is that subsequent works on speaker identification systems would obtain much better performances using the SVMs with much potent kernels or using other state-of-the-art methods just to make sure the applications that utilize this technology get the desired value. A challenge that was faced during this work is the speaker corpus. Despite the availability of robust speech corpuses which have many speakers and have very long conversations per speaker, they are relatively expensive. Future work will be much easier if there could be free or affordable speech databases available to researchers and students. This is important because, the larger the database, the better its results can be extrapolated to real life applications. Lastly, the computation time required for the simulations of this work is very much. Future works need to develop better ways to execute the simulations in much shorter periods of time.

References

[1] Reynolds, D. A. (2002). An overview of automatic speaker recognition. In Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)(S. 4072-4075).

[2] Kinnunen, T., & Li, H. (2010). An overview of text-independent speaker recognition: From features to supervectors. Speech Communication, 52(1), 12-40. doi:10.1016/j.specom.2009.08.009

[3] Campbell Jr, J. P. (1997). Speaker recognition: a tutorial. Proceedings of the IEEE, 85(9), 1437-1462.

[4] Bimbot, F., Bonastre, J. F., Fredouille, C., Gravier, G., Magrin-Chagnolleau, I., Meignier, S., Merlin, T., Ortega-Garcia, J., Petrovska-Delacretaz, D., & Reynolds, D. A. (2004). A tutorial on text-independent speaker verification. EURASIP journal on applied signal processing, 2004, 430-451.

[5] Reynolds, D. A., & Rose, R. C. (1995). Robust text-independent speaker identification using Gaussian mixture speaker models. Speech and Audio Processing, IEEE Transactions on, 3(1), 72-83.

[6] Reynolds, D. A. (1992). A Gaussian mixture modeling approach to text-independent speaker identification. Georgia Inst. Of Technology, Sept 1992.

[7] Reynolds, D. A., Quatieri, T. F., & Dunn, R. B. (2000). Speaker verification using adapted Gaussian mixture models. Digital signal processing, 10(1), 19-41.

[8] Digalakis, V. V., Rtischev, D., & Neumeyer, L. G. (1995). Speaker adaptation using constrained estimation of Gaussian mixtures. Speech and Audio Processing, IEEE Transactions on, 3(5), 357-366.

[9] Campbell, W. M., Sturim, D. E., & Reynolds, D. A. (2006). Support vector machines using GMM supervectors for speaker verification. Signal Processing Letters, IEEE, 13(5), 308-311.

[10] Abu-Mustafa, Y., "Support Vector Machines", March 3rd, 2015 [Online]. Available: https://work.caltech.edu/library/140.pdf

[11] Weston, J. Support Vector Machine. Online, pristupljeno, 13, 13.

[12] Mel Frequency Cepstral Coefficient (MFCC) tutorial Jan 19, 2015 [Online]. Available: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

[13] A Tutorial on Cepstrum and LPCCs May 2nd, 2015 [Online]. Available: http://www.practicalcryptography.com/miscellaneous/machine-learning/tutorial-cepstrum-and-lpccs/

[14] Sadaoki Furui (2008) Speaker recognition. Scholarpedia, 3(4):3715.

[15] Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Friedman, J., & Tibshirani, R. (2009). The elements of statistical learning (Vol. 2, No. 1). New York: springer.

[16] Wan, V., & Campbell, W. M. (2000, December). Support vector machines for speaker verification and identification. In *NEURAL NETWORKS SIGNAL PROCESS PROC IEEE* (Vol. 2, pp. 775-784).

[17] Chao, Y. H., Tsai, W. H., & Wang, H. M. (2010, November). Speaker verification using support vector machine with LLR-based sequence Kernels. In *Chinese Spoken Language Processing (ISCSLP), 2010 7th International Symposium on* (pp. 182-185). IEEE.

[18] Cummins, F., Grimaldi, M., Leonard, T., & Simko, J. (2006, June). The chains speech corpus: Characterizing individual speakers. In *Proc. Int. Conf. on Speech and Computer (SPECOM)* (pp. 1-6).