# Discriminant Feature Space Transformations for Automatic Speech Recognition

*Vikrant Tomar*

*McGill ID: 260394445*


Department of Electrical & Computer Engineering
McGill University
Montreal, Canada

February 20, 2012

# Preface

The report will cover analysis, implementation, and test of various techniques in Automatic Speech Recognition. Primary metric of the algorithm analysis, discussed herein, include the ability to successfully recognize a series of spoken digits in terms of %- word accuracy. The speech corpus in the focus is the noise-mixed TI-connected digit database (also known as ETSI Aurora-2 database). The implementation of the algorithm is done on MATLAB R2010b in a 64-bit environment.

In the report, the natural algorithm is denoted with log; otherwise the base is stated. Throughout the report, the whole sequence of a signal will be denoted by a vector written with bold letters, *i.e.* $\boldsymbol{x}$, whereas $x(n)$ will correspond to an element in that sequence. If both time and vector indices are used the variable $\boldsymbol{h}_j(k)$ is the $j$'th coefficient at time index $k$, where the bold face represents a vector. Literature references are represented with numbers in IEEE format, e.g. [number], and a the full list of references is found in the References section.

The report also includes the MATLAB implementation of the algorithm.

# Synopsis

One of the most widely used techniques for Automatic Speech Recognition (ASR) is to use Mel-Frequency Cepstrum Coefficients (MFCC) features. Then recognition is performed by training a Hidden Markov Model (HMM) on features of speech vectors. MFCC features gives a fairly accurate static information of speech, however, it is not a good technique to capture time-evolution of speech spectrum. For the same, the first and second difference coefficients, known as the $\Delta$-Cepstrum and $\Delta\Delta$ or acceleration-Cepstrum are appended to feature vectors. However, components of these super vectors are strongly correlated. Furthermore, it is not very clear if capturing the time evolution of speech frames in this manner is optimal for ASR.

In recent years, new ways of combining the static and dynamic features of speech have been proposed, which focus on techniques of discriminant analysis. One such method is to utilize Linear Discriminant Analysis (LDA). This report focuses on investigating the effectiveness, and limitations (if any) of ASR using LDA.

# Contents

# List of Figures

# Listings

# Chapter 1

# Introduction

Speech is the fundamental mode of communication among humans. In addition, it also posses information about the language, gender, emotional state *etc.,* of the speaker. Over the last few decades, there have been a lot of emphasis in developing applications related to speech technology such as Automatic Speech Recognition (ASR), speech synthesis, speech understanding, *etc.* The ultimate goal of all these applications is to achieve successful human-machine interaction through speech. In this chapter, basics of ASR and challenges are described very briefly so that the reader could develop the necessary background for the rest of the report.

## 1.1 Feature Extraction

One of the primary building blocks of an ASR system is signal analysis, which refers to parametrize the speech signal to best suit the ASR model. Since in an ASR system, *what* is spoken is more important than *who* spoke it, the parametrization step should be aimed at capturing the relevant information, while discarding the irrelevant information. One crucial and formidable aspect of this parametrization step is to eliminate any variability in the speech due to speaker and/or environment.

The signal analysis of the present day ASR systems is based on short term spectral analysis [1], usually Fourier analysis. For further processing and smoothing, Mel frequency cepstral coefficients (MFCC) filtering is widely used [2]. Such a filter bank is referred to Mel-filter bank. Various signal processing steps in the computation of MFCC features are illustrated in Fig. 1.1.

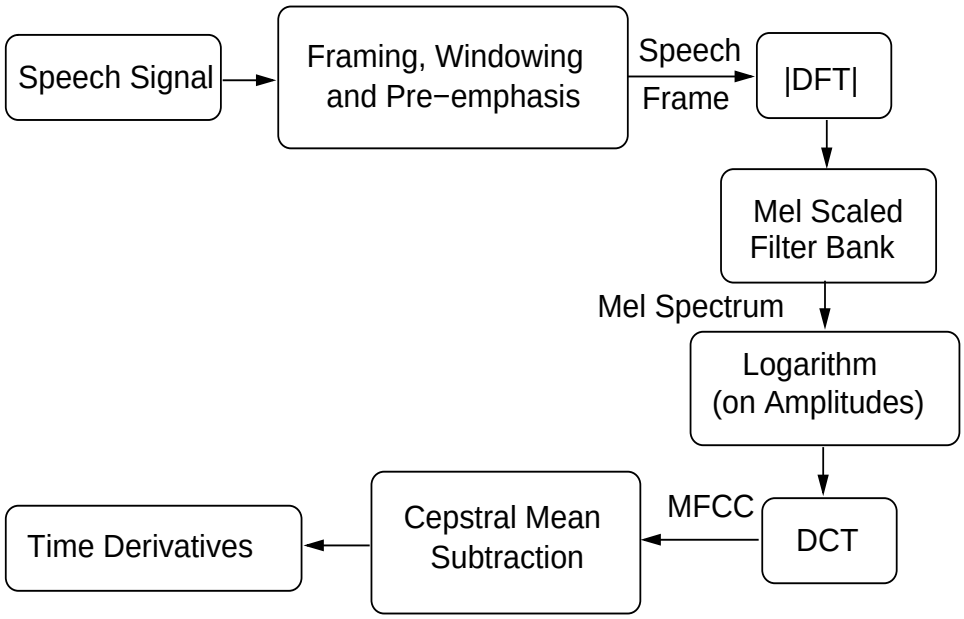



**Fig. 1.1** MFCC Feature Extraction

The most important step of the procedure is the Mel filter bank processing. This filter converts DFT spectrum to Mel frequency spectrum. The DFT coefficients are binned into required number of channels by multiplying them by weights corresponding to Mel-scaled triangular filter mask. The relation between linear frequency (Hz), and Mel frequency is given by

$$\begin{aligned} f_{Mel} &= 1127 \log\left(1 + \frac{f_{Hz}}{700}\right) \\ \text{or,} \quad \mathrm{f_{Mel}} &= 2595 \log_{10}\left(1 + \frac{\mathrm{f_{Hz}}}{700}\right) \end{aligned} \tag{1.1}$$

where $f_{Mel}$ and $f_{Hz}$ correspond to the frequencies in Mel and linear domains respectively. The structure of a typical Mel filter bank is shown in Fig. 1.2. It should be noted that the filter has higher resolution at lower frequencies and lower resolution at higher frequencies, which is motivated from the processing in the human ear, [1]. The output of the Mel filter bank is then log compressed and converted into Cepstral coefficients by applying discrete cosine transform (DCT), and long term mean (typically that of a sentence) is subtracted from the coefficients in order to remove the effect of time-invariant distortions introduced

by the transmission channel and the recording device, and lip-radiation.

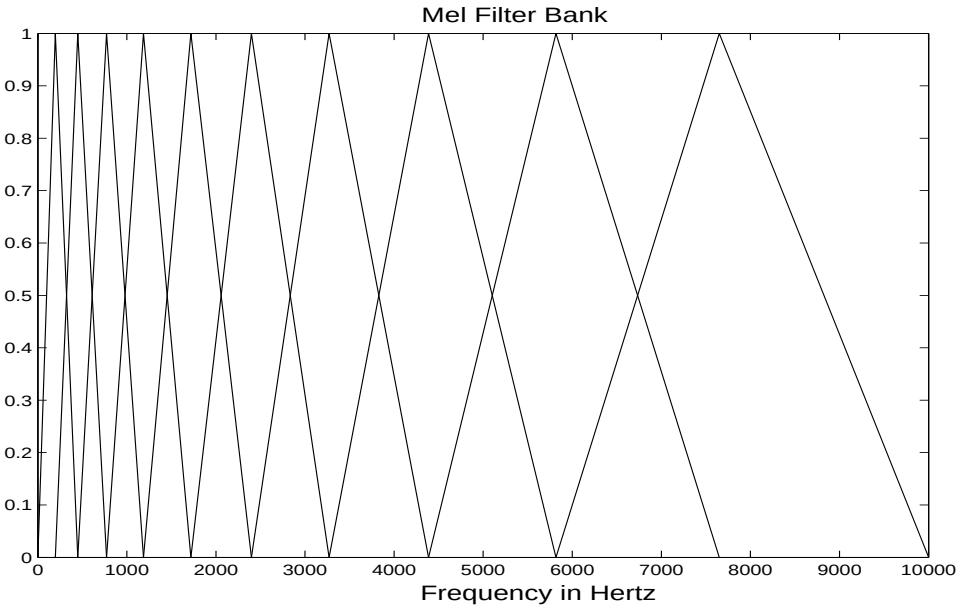



**Fig. 1.2** A Typical Mel filter bank

The output vectors of this process are known as MFCC features of the speech signal, and constitutes the basis of modern-day speech recognition systems.

## 1.2 Challenges in ASR

ASR systems can be broadly classified as *speaker dependent* and *speaker independent*. A recognizer trained using speech data from a particular speaker is referred to as speaker-dependent (SD) recognition system, and it may not be accessible to other speakers. In contrast, speaker-independent (SI) recognition system in principle can be accessed by any speaker. Therefore, for improved performance it is important to minimize the effects of the individual characteristics and only consider the linguistic information that is common to all speakers uttering the same language. The major challenges in ASR arise due to the variability in the speech signal and can be accounted to the varying acoustic characteristics of speakers, the background noise as well as based on the size of the task, vocabulary and grammar.

The acoustics of the speech signal varies even for the same spoken text utterance,

on the account of physiological differences in the speech production organs, as well as the differences in linguistic style and habits, among others. The difficulty in a speech recognition problem also depends on the nature of the speech being recognized. The gamut of speech recognition systems can vary from an SD isolated word to a large SI vocabulary continuous speech recognition system, where co-articulation and other continuity effects increase the level of difficulty of the task. For instance, the read speech is much easier to transcribe than conversational speech.

One of the most important challenge in optimal ASR is noise (environmental or otherwise). The presence of ambient noise is inevitable in real-world applications. Furthermore, the problem is made even more formidable by the varying characteristics of noise from one situation to another, because of which it is not feasible to use a static model of noise. A mere presence of noise can severely degrade the performance of an ASR system. Because of this, noise robustness is a very important and highly active area of research in speech recognition. Although, the direct treatment of this problem is beyond the scope of this report, we shall be discussing, in brief, various proposed techniques of noise reduction in ASR.

Having introduced the basics of ASR in this chapter, we would look at ASR from a Machine Learning perspective in the next chapter.

# Chapter 2

# A Machine Learning viewpoint of ASR

Most of the present day state-of-the-art ASR systems are based on statistical approaches, where typical ASR problem translates to find the optimal sequence of words $\hat{\boldsymbol{W}}$ from all possible sequences of words $\boldsymbol{W}$ that yields the highest probability for the given acoustic feature sequence $\boldsymbol{X}$.

$$\hat{\boldsymbol{W}} = \arg\max_{W} p(\boldsymbol{W}|\boldsymbol{X}) \tag{2.1}$$

The posterior probability can further be decomposed using Bayes's rule:

$$p(\boldsymbol{W}|\boldsymbol{X}) = \frac{p(\boldsymbol{X}|\boldsymbol{W})p\boldsymbol{W}}{p(\boldsymbol{X})} \tag{2.2}$$

where

- $p(\boldsymbol{X}|\boldsymbol{W})$: probability of observing $\boldsymbol{X}$ under the assumption that $\boldsymbol{W}$ is a true utterance.
- $p(\boldsymbol{W})$: probability that word sequence $\boldsymbol{W}$ is uttered.
- $p(\boldsymbol{X})$: average probability that $\boldsymbol{X}$ will be observed.

Note that the denominator does not depend on the word sequence $\boldsymbol{W}$, and $p(\mathbf{X})$ is same for all candidate word sequences. Therefore, it does not affect the outcome of finding the

optimal sequence, and we may safely ignore it and write:

$$\hat{\mathbf{W}} = \arg\max_{W} \left\{ \overbrace{p(\boldsymbol{X}|\boldsymbol{W})}^{acoustic\ model\ probability} \times \underbrace{p(\boldsymbol{W})}_{language\ model\ probability} \right\} \tag{2.3}$$

As denoted in the expression 2.3, the entire decision model thus can be decomposed into acoustic and language model properties.

## 2.1 Acoustic Modelling of Speech

The acoustic model provides the probability of observing an acoustic feature vector for a given phone or word given that a particular word-sequence is spoken. Thus the main objective of an acoustic model is to capture the information present in the acoustic feature vectors. Furthermore, the model should also taken into account the time varying statistics of speech signal in order to take care of the co-articulation effects and other effects caused by continuity constraints. Researchers have explored a number of approaches for characterizing speech, such as Dynamic Time Warping (DTW) [3, 4] or Artificial Neural Networks (ANN) [5], however, most of the modern days state-of- the-art recognition systems use Hidden Markov Models (HMM) [6, 7, 8]. Following is a brief description of HMM for speech recognition.

Let $\boldsymbol{X_1}, \boldsymbol{X_2}, \cdots \boldsymbol{X_n}$ be a sequence of random vectors representing the acoustic features. The sequence is said to be *first order Markov* if

$$p(\boldsymbol{X}_i|\boldsymbol{X}_{i-1}, \cdots, \boldsymbol{X}_1) = p(\boldsymbol{X}_i|\boldsymbol{X}_{i-1})$$

That is, the future of the event, given present, is independent of the past. Thus, HMM allows the acoustic data to be probabilistic outputs of a (hidden) Markov chain, in which each state $i$ corresponds to a ***segment*** of a word or phone with an observation probability $p(\boldsymbol{x}|state = i)$. Therefore, the model can take advantage of the mathematical convenience of Markov assumption in determining the *unknown* phone string from the *known* outputs of the process.

An HMM can be characterized by following variables:

1. State of the chain at time $t$, $q_t$, and total number of states, $N$.

2. The 1-step state-transition probability matrix $\boldsymbol{A}$, where,

$$\boldsymbol{A}_{ij} = p(q_t = j | q_{t-1} = i)$$

3. Observation (output) probability distribution for each state[1].

$$\boldsymbol{B} = \{b_j(\boldsymbol{x})\}, \qquad b_j(\boldsymbol{x}) = p(\boldsymbol{x} | q_t = j)$$

4. Initial State Distribution, $\boldsymbol{\Pi} = \{\pi_i\}, \quad \pi_i = p(q_1 = i)$

Thus, an HMM ASR model can be defined as:

$$\Lambda = \{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\Pi}\} \tag{2.4}$$

The objective of HMM-based acoustic modelling is to accurately estimate the parameters of the HMM from a training dataset and efficiently compute the output (acoustic model) probability $p(\boldsymbol{X}|\boldsymbol{W})$. Here the word $\boldsymbol{W}$ is represented by the model $\Lambda$. As mentioned before, an HMM state could represent ***segment*** of a word, phone, or other units of speech. For a word-based HMM, an HMM for each word is created with $N$ states that represents the segments of the word (e.g. phones)[2]. Hence, in the work leading to this report, each spoken digit (word) has a corresponding HMM model with $N = 16$ states. Furthermore, The left-to-right HMM model, in which the only transitions allowed are the transition to the same state or to the next right state, is most suitable for speech recognition as the left-to-right structure is intuitively compatible with the temporal nature of the speech signals [6].

### 2.1.1 Gaussian Mixture Models for Observation Probabilities

As mentioned in the Section 2.1, each state $i$ corresponds to a ***segment*** of a word or phone with an observation probability $p(\boldsymbol{x}|state = i)$. These observation probabilities can be represented by a parametric probability density function (pdf), which facilitates the

[1]Here $\boldsymbol{x} \in \Re^d$ indicates an arbitrary feature vector belonging to state $j$. Thus, $b_j(\boldsymbol{x})$ is the probability of observing $\boldsymbol{x}$, given my Markov chain is in state $j$.

[2]For phonetic HMMs, each phone is represented by an HMM that consists of states corresponding to segments of the phone (e.g. onset, steady and release portion of vowels).

process of training HMMs. A suitable and commonly used distribution for representing the output feature vector $\boldsymbol{x}$ is the multi-variate Gaussian distribution. Thus, the probability of observing a $d$ dimensional feature vector $\boldsymbol{x}$ in state $i$ for an HMM model $\Lambda$ (corresponding to $\boldsymbol{x}$) is estimated as given below:

$$p(\boldsymbol{x}|\Lambda, q=i) = N(\mathbf{x}; \boldsymbol{\mu}_j, \Sigma_j) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right] \tag{2.5}$$

where, $\boldsymbol{\mu}_i \in \Re^d$ and $\Sigma_i \in \Re^{d\times d}$ denote the mean vector and covariance matrix of the $i^{\text{th}}$ state, respectively.

A Gaussian distribution, however, provides a limited representation of the actual observation probabilities. A more flexible and comprehensive distribution, Gaussian mixture models (GMM), can be obtained by combining multiple weighted Gaussian densities.

$$b_i(\mathbf{x}) = \sum_{m=1}^{M} \lambda_{i,m} N(\mathbf{x}, \boldsymbol{\mu}_{i,m}, \Sigma_{i,m}), \tag{2.6}$$

where M indicates the total number of Gaussian mixtures, each having its own mean vector $\boldsymbol{\mu}_{i,m} \in \Re^d$ and covariance structure $\Sigma_{i,m} \in \Re^{d\times d}$. $\lambda_{i,m}$ denotes the mixture weight corresponding to the $m^{\text{th}}$ Gaussian of $i^{\text{th}}$ state. Furthermore, the mixture weights are constrained to lie between 0 and 1, $0 \leq \lambda_{i,m} \leq 1$, and their sum is 1, $\sum_{m=1}^{M} \lambda_{i,m} = 1$[3].

GMMs are a powerful tool for modelling distributions whose underlying structure is unknown, as often is the case in acoustic modelling. The distribution itself is parameterized by a balance between the number of mixtures, M, and estimation of observation probability distributions. *A large value of M can lead to over-fitting, whereas, a smaller value may not produce a good representation of data.* It is noteworthy here that modelling different classes with different number of mixture components may be even more beneficial. The parameter estimation is usually done using an iterative procedure until the model parameters are optimized with respect to some criterion. The GMM parameters are typically estimated using the Baum-Welch Expectation Maximization (EM) algorithm for HMM, details of which can be found in [9]. The acoustic likelihood is calculated using the forward-backward

[3]In this project work, six Gaussian mixture components per class are used, *i.e.,* M = 6.

algorithm [7, 9], *i.e.*,

$$p(\boldsymbol{W}|\boldsymbol{X}) = \sum_{\forall Q} p(\boldsymbol{X}, Q|\boldsymbol{W}) \tag{2.7}$$

where, $Q$ represents all possible paths in $\boldsymbol{W}$.

## 2.2 Language Modelling of Speech

The language model probability is an *a-priori* probability $p(\boldsymbol{W})$ of a word sequence, $\boldsymbol{W} = w_1, w_2, \cdots, w_N$. Generally complex statistical models, which takes into account the syntax, semantics, and pragmatics of the language, are used for this purpose.

One such widely accepted model is the $l$-gram language model, which assumes the language to obey an $(l\text{-}1)^{\text{th}}$ order Markov process. Thus, the probability of the $l^{\text{th}}$ word depends on the ($l$-1) predecessors, *i.e.*,

$$p(\boldsymbol{W}) = \prod_{i=1}^{l} p(w_i|w_{i-1}, \cdots, w_{i-l+1}) \tag{2.8}$$

## 2.3 Global Search

The most crucial aspect of an ASR system is to combine the acoustic model probabilities $p(\boldsymbol{X}|\boldsymbol{W})$ and the language model probabilities $p(\boldsymbol{W})$ in order to find the optimum word sequence that best represents the underlying acoustic sequence $p(\boldsymbol{W}|\boldsymbol{X})$. Further from Eq. (2.7), the approximate likelihood rule is given as:

$$p^*(\boldsymbol{X}|\boldsymbol{W}) = \max_{\forall Q} p(\boldsymbol{X}, Q|\boldsymbol{W}) \tag{2.9}$$

where * signifies approximation. This equation is usually used while performing recognition or search for optimum word sequence; it is referred to as the Viterbi approximation.

Having established the basics of recognition and search in ASR, and analyzed the most frequently used algorithms from a Machine Learning perspective, we will discuss the Linear Discriminant Analysis (LDA) based contemporary methods of feature space transformation and discrimination in the next chapter.

# Chapter 3

# Discriminant Feature-space Transformations

The primary objective of feature extraction is to filter-out, from the high amount of information contained in the speech waveforms, any irrelevant and redundant information. Thus, the output feature vectors are essentially a minimal representation of the most relevant information to the *current* ASR task. The MFCCs give a fairly accurate measure of static features of speech, however, the time evolution of the speech spectrum is also of great interest from an optimal ASR perspective, especially when a number of theories suggest that such information plays a crucial role for human speech perception [1, 10],

In this chapter, we will discuss the ways to incorporate the information pertaining to time-evolution in the MFCC features, followed by alternative suggestions, which suggests modelling ASR as a classification problem.

## 3.1 Capturing the Dynamics of Speech

One of the most common feature analysis techniques involves concatenating a set of static and dynamic feature vectors for each given speech frame. As explained in Section 1.1, the static features are often computed using Cepstral analysis, whereas, the dynamic features are computed by the first and second order difference/derivatives of the static features[11]. These difference vectors are then concatenated to the original static feature vector to produce a super-vector.

Despite of being one of the most widely used techniques for dynamic feature extraction,

the aforementioned technique is not necessarily the most parsimonious way of capturing dynamics of speech. Furthermore, the super-vectors created this way, are known to have high degree of correlation among their components, which violates major ASR assumptions [12]. Therefore, a number of efforts have been made to perform this task in a more mathematically sound framework. One important line of thought is to consider ASR as a type of classification problem, however, it is not clear what the best definition of classes should be; various choices such as words and phones have been considered in the literature [13, 12]. Regardless of this choice, the fundamental problem thus becomes to discriminate between such classes.

## 3.2 ASR as a Classification Problem

In the new light of considering ASR as a classification problem, a number of new techniques to capture time-evolution of the speech spectrum has been proposed. In these techniques, instead of performing the simple concatenation of static and dynamic features, the extracted features for a given frame are concatenated with features from few preceding and succeeding frames. In addition, a discriminant feature-space transformation is used to reduce the dimensionality of the resulting super-vectors while maximizing class discrimination[1]. Consequently, not only we have used a proper mathematical basis for extracting information about the time evolution of speech frames, but also we can expect higher performance in a new space where classes, however defined, have been maximally separated.

Having motivated the use of discriminant analysis as a preprocessing step for ASR, we will now proceed with more details regarding one of the most widely used techniques: Linear discriminant analysis (LDA).

[1] In statistical pattern recognition, one approach for tackling the issues of high dimensionality is to use a linear transformation to lower the dimensionality of the data. Classically, these approaches are divided into two categories: techniques based on principal component analysis (PCA) and techniques based on Fisher's linear discriminant. While PCA based techniques aims at finding the projection that best represents the data in a least-squares sense, the techniques based on Fisher's linear discriminant are aimed at finding the projection that maximizes the separability of the data [14, 15]. As a result, these latter techniques have been widely used in the feature analysis stage of ASR systems.

## 3.3 LDA for ASR based classification problem

LDA is fast becoming a standard method for class-discrimination based ASR [16]. In the transformed space, the feature vectors, while containing information about the time evolution of the spectral content of the signal in the current frame, are optimal in the sense that they allow for maximum class discrimination.

### 3.3.1 Algorithm

Consider the set $\boldsymbol{X} = \{\boldsymbol{x}_1, \cdots \boldsymbol{x}_n\}$ of vectors in $\Re^d$ , each of which belongs to only one class $c \in \{c_1, c_2, \cdots, c_k\}$, where $k$ is the number of classes. We refer to this set $\boldsymbol{X}$, along with the corresponding class labels, as our training data. We are interested in a classification problem where, using our training data, we would like to be able to determine the class label of any given vector $\boldsymbol{x} \in \Re^d$. The goal of discriminant analysis is to estimate the parameters of a projection matrix $\boldsymbol{P} \in \Re^{m \times d}$, with $m \leq d$, to transform vectors from a $d$-dimensional feature-space onto an $m$ dimensional feature-space, where class discrimination is maximized. The transformation is performed according to

$$\boldsymbol{y}_i = \boldsymbol{P}\boldsymbol{x}_i \qquad \forall i = 1, 2, \cdots, n \tag{3.1}$$

where $\boldsymbol{x}_i$ is an arbitrary vector in the source space, and $\boldsymbol{y}_i$ is the corresponding transformed vector in the new feature-space.

Furthermore, lets assume that each class $c_i$ contains $N_i$ vectors and is characterized by its mean vector $\boldsymbol{\mu}_i$, and the covariance matrix $\Sigma_i$. We define the following two scatter metrics [13]:
Within-class scatter:

$$S_W = \frac{1}{N} \sum_{i=1}^{k} N_i \Sigma_i \tag{3.2}$$

Between-class scatter:

$$S_B = \frac{1}{N} \sum_{i=1}^{k} \left( N_i \boldsymbol{\mu}_i \boldsymbol{\mu}_i^T - \boldsymbol{\mu}\boldsymbol{\mu} \right) \tag{3.3}$$

where $N = \sum_i N_i$, and $\boldsymbol{\mu} = \sum_i x_i$. We can see that the within-class scatter is a measure of the average variance of the data within each class, while the between-class scatter represents the average distance between the means of the data in each class and the global

mean. In other words, the within-class scatter metric indicates how closely situated, in the transformed space, the features belonging to the same class are, and the between-class scatter indicates the same for feature-vectors NOT belonging to same class.

Our aim is to maximize the between-class scatter, while at the same time minimize the within-class scatter. Given a transformation matrix $\boldsymbol{P}$, we can define the following indicator of class separability.

$$I_L(\boldsymbol{P}) = \frac{|\boldsymbol{P}S_B\boldsymbol{P}^T|}{|\boldsymbol{P}S_W\boldsymbol{P}^T|} \tag{3.4}$$

where, we have normalized a measure of the average distance between the centroids of each class, by a measure of the average within-class variance. Therefore, the highest separability is attained where this ratio is maximized.

Fortunately, finding the transformation $\boldsymbol{P}_{lda}$ that maximizes $I_L(\boldsymbol{P})$ has a closed-form solution. The columns of the matrix are given by the generalized eigenvectors corresponding to the largest eigenvalues in the following equation [14].

$$S_B\boldsymbol{P}^j_{lda} = \lambda_j S_W\boldsymbol{P}^j_{lda} \tag{3.5}$$

where $\boldsymbol{P}^j_{lda}$ indicates the columns of matrix $\boldsymbol{P}_{lda}$.

## 3.4 Semi-tied Covariance Matrices and Transform

LDA works well under the assumption that each class of data is generated by an exponential distribution with a shared covariance structure and separate means. This is a large improvement over the previously described methods since an attempt is made to separate the classes, however, the assumption that data for each class is generated by a single exponential with shared covariance structure is still very strong.

In fact, LDA produces a projected space whose dimensions, in general, are highly correlated. Therefore, there is a need to decorrelate the resulting space, however, it is hard to find a single transform which diagonalize the covariance structures for all states. For this problem, Gales [17, 18] proposed a Semitied Covariance Transform (STC). The discussion of the STC is beyond the scope of this report, and interesting reader is advised to refer to one of the aforementioned references.

# Chapter 4

# Experimental Setup, Results and Conclusion

The work discussed in this report has evaluated the effectiveness of discriminant feature-space transformation methods with respect to traditional $\Delta$, and $\Delta\Delta$ (acceleration) based methods. In particular, LDA based method of discriminant feature-space transformation was considered.

## 4.1 Database, Tools, and Procedure

The European Telecommunications Standards Institute (ETSI) Aurora 2 database was used in the work leading to this report. The database –containing a total of 8440 utterances from 55 males and 55 females– has been created by adding various simulated noise samples to connected digit utterances from the TI spoken-digits (TIGITS) database. Eight different noise environments are simulated in Aurora 2: subway, speech babble, automobile, exhibition hall, restaurant, street, airport, and train station. To reduce the mismatch between the model and the noisy test speech, the HMM model was trained on the multi-conditioned noise mixed dataset. The testing was done for 3 separate testsets (A, B, and C), each of which consists of 4004 utterances with different kind of noise conditions (discussed further with results). Furthermore, there were different SNR levels of each noise type, *viz.* clean signal, 20dB SNR, 15dB SNR, 10dB SNR, and 5dB SNR.

For the implementation of GMM and HMM, the mainstream speech recognition toolkit HTK (HMM Tool Kit) version 3.4.1 was used [19]. For baseline results, the MFCC features

(calculated using HTK) were fed into the HTK system, which executed the selected scripts, and produced recognition results[1].

For LDA based ASR, a simple program, written in Python (available on request), was used to concatenate a total of 9 (4 preceding + current + 4 succeeding) speech frames (feature dimension: 13) to create a 117-dimension super-feature-vector. The LDA projection matrix $\boldsymbol{P}_{lda}$ was then calculated using the multi-conditioned training set. The same $\boldsymbol{P}_{lda}$ was then used to perform discriminant feature-space transformation on the testsets and project the features to a 39 dimensional space. These resulted features (both training and testing) were then fed into HTK system with a new set of configuration. Some implementations like LDA were done in MATLAB R2010b in a 64-bit environment. The MATLAB codes are given in the Appendix A.

For easy understanding, a flow chart is given in Figure. 4.1.

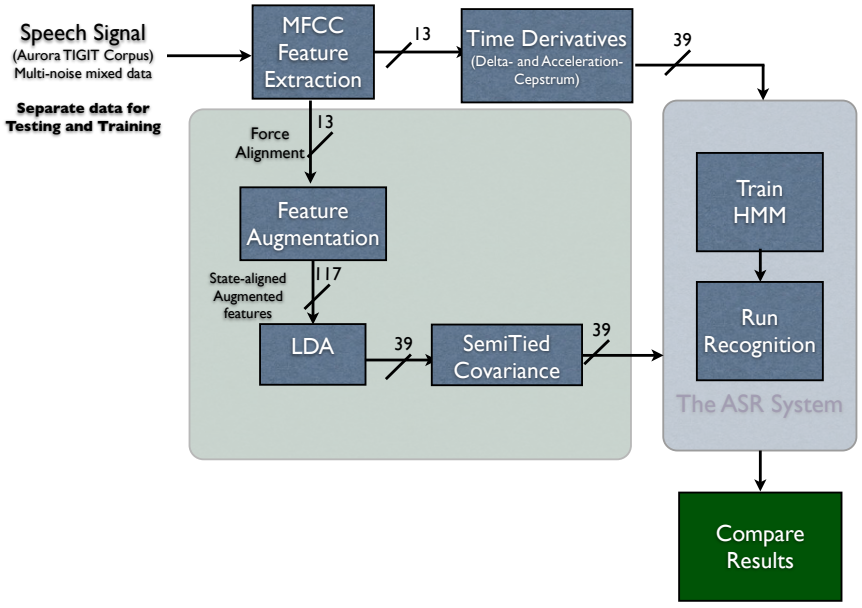



**Fig. 4.1** Flowchart of the followed procedures.

[1]HTK specific details have been omitted here. One may refer to The HTK Handbook for details [19].

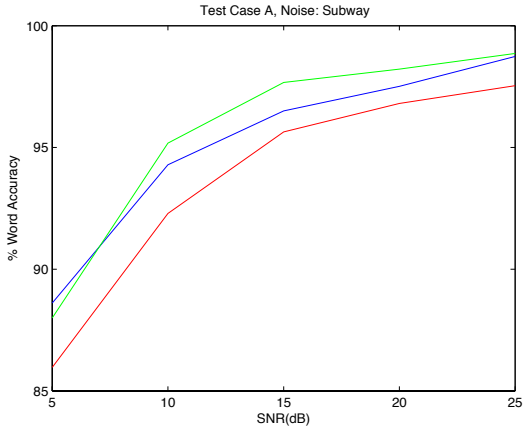
## 4.2 Results and Conclusion

The results of various tests run on various subsets and noisy conditions are discussed in this section. The results are illustrated in figures 4.2, 4.3, and 4.4. For each testset, different noise conditions were tested separately. Furthermore, the results are compared for the baseline results (traditional MFCC and $\Delta$, and acceleration based feature vectors), and LDA without Semitied Covariance Transform (LDA in the figures), and LDA with Semitied Covariance Transform (STC in the figures). Primary metric of performance is taken to be the algorithm's ability to successfully recognize a series of spoken digits in terms of %- word accuracy, *i.e.*, higher is better.

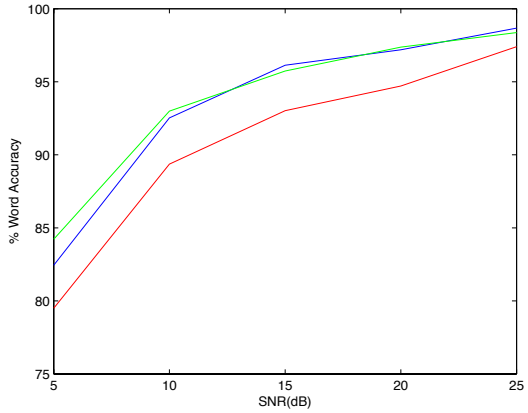
In all these figures, the x-axis represents the SNR level, whereas the y-axis represents the %-word accuracy. Please note that the point 25dB on x-axis refers to the clean speech.

It is evident from the figures that LDA with STC, in general, produces better results than the baseline case. It is also important to note here that performance of recognition degrades when LDA alone is used to create the feature vectors, which is in-line with our discussion in Section 3.4.

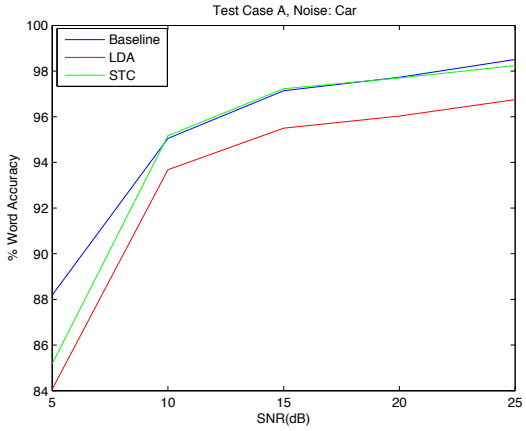
One of the issues concerned with the use of this speech corpus is the fact there are limitations on how well speech in a noisy environment can be simulated by adding noise samples to clean speech. For example, referred to as the Lombard effect, speakers usually attempt to speak more effectively and therefore differently (in terms of loudness, speed, emphasis, *etc.*), in noisy environments [20]. As all the utterances in Aurora 2 were recorded in a quiet environment, such effects are absent from this corpus. This poses some limitations on how realistic the obtained test results are.



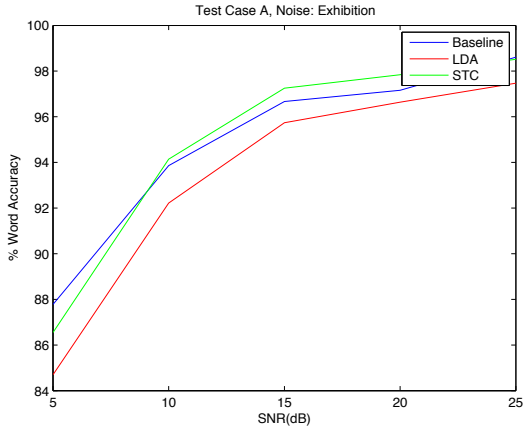

(a) Noise 1 of Testcase A




(b) Noise 2 of Testcase A

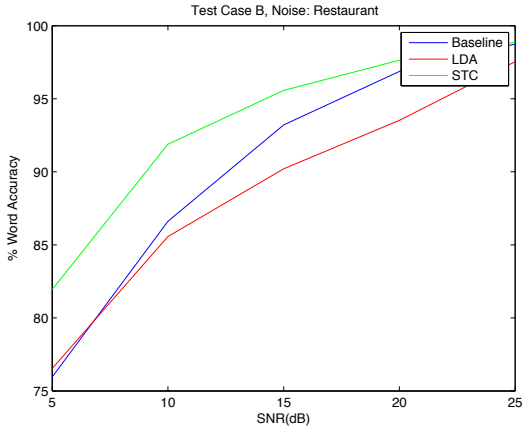



(c) Noise 3 of Testcase A

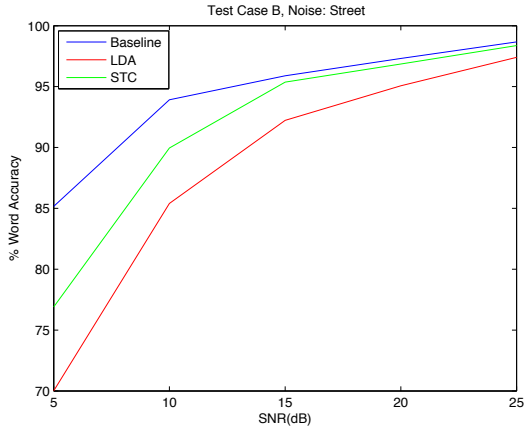



(d) Noise 4 of Testcase A

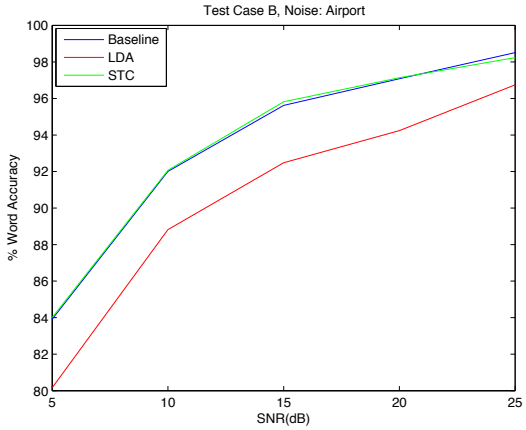
**Fig. 4.2** Recognition Results for Test subset A



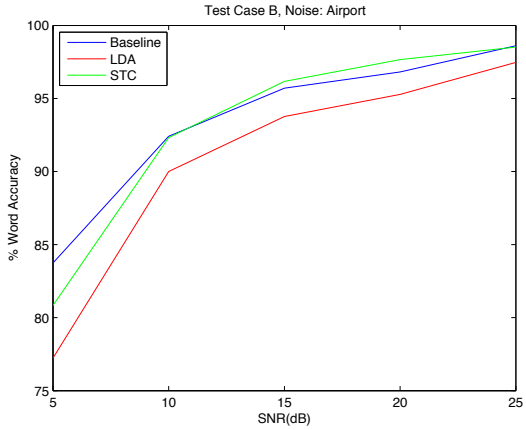

(a) Noise 1 of Testcase B




(b) Noise 2 of Testcase B

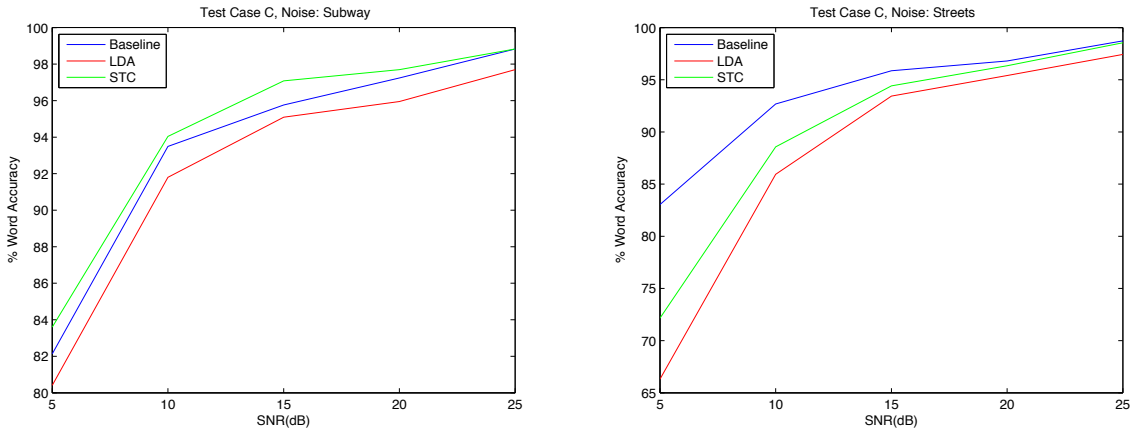



(c) Noise 3 of Testcase B




(d) Noise 4 of Testcase B

**Fig. 4.3** Recognition Results for Test subset B




(a) Noise 1 of Testcase C

(b) Noise 2 of Testcase C

**Fig. 4.4** Recognition Results for Test subset C

# References


[1] D. O'Shaughnessy, *Speech Communication: Human and Machine*. New York, NY, USA: IEEE, 2002.

[2] S. B. Davis and P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuous Spoken Sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, pp. 357 – 366, Aug 1980.

[3] H. Sakoe and S. Chiba, "Dynamic programming optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, pp. 43–49, 1978.

[4] G. M. White and R. B. Neely, "Speech recognition experiments with linear prediction, bandpass filtering, and dynamic programming," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, pp. 183–188, 1976.

[5] R. P. Lippmann, *Readings in Speech Recognition*, ch. Review of Neural Networks for Speech Recognition. Morgan Kaufmann Publishers, Inc., 1991.

[6] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, pp. 257–286, 1989.

[7] L. R. Rabiner and R. W. Schafer, *Introduction to Digital Speech Processing*, vol. 1 of *Foundations and Trends in Signal Processing*. Now, 2007.

[8] F. Jelinek, *Statistical Methods for Speech Recognition*. The MIT Press: London, 1999.

[9] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing*. Prentice Hall PTR, 2001.

[10] S. Furui, "Speaker independent isolated word recognition using dynamic features of speech spectrum," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pp. 52 – 59, Feb 1986.

[11] J. Hernando, "Maximum likelihood weighting of dynamic speech features for CDHMM speech recognition," in *ICASSP: IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 1267–1270, 1997.

[12] Y. Tang and R. Rose, "A study of using locality preserving projections for feature extraction in speech recognition," in *ICASSP: IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2008.

[13] G. Saon, M. Padmanabhan, R. Gopinath, and S. Chen, "Maximum likelihood discriminant feature spaces," tech. rep., IBM T. J. Watson Research Center, 2000.

[14] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley Interscience, 2nd ed., 2000.

[15] C. M. Bishop, *Pattern Recognition and Macine Learning*. Springer, 2006.

[16] P. F. Brown, *The Acoustic-Modeling Problem in Automatic Speech Recognition*. PhD thesis, Carnegie Mellon University, May 1987.

[17] M. J. F. Gales, "Semi-tied covariance matrices for hidden markov models," *IEEE Transactions on Speech and Audio Processing*, vol. 7, pp. 272 – 281, May 1999.

[18] M. J. F. Gales, "Semi-tied covariance matrices for hidden markov models," tech. rep., Cambridge University, Engineering Department, Trumpington Street, Cambridge, CB2 1PZ, England, Feb 1998.

[19] University of Cambridge and Microsoft Research, "HTK Speech Recognition Toolkit, http://htk.eng.cam.ac.uk/."

[20] P. Lau, "The lombard effect as a communicative phenomenon," tech. rep., UC Berkeley Phonology Lab, 2008.

# Appendix A

# Matlab Codes

This section enlists various MATLAB codes used in this project.

## LDA

Here the algorithm to perform LDA and produce final mapping (transformation matrix $\boldsymbol{P}_{lda}$) is given.

```
function [mapping] = lda(X, labels, no_dims)
%LDA Perform the LDA algorithm
%
% The function runs LDA on a set of datapoints X. The variable
% no_dims sets the number of dimensions of the feature points in the
% embedded feature space (no_dims >= 1, default = 2). The maximum number
% for no_dims is the number of classes in the data minus 1.
% The function returns the final mapping (P) in the cell mapping.
% Furthermore, the mapped data can be computed as:
% mappedX = X * P;

    if ~exist('no_dims', 'var') || isempty(no_dims)
        no_dims = 2;
    end

  % Make sure data is zero mean
    mapping.mean = mean(X, 1);
  X = bsxfun(@minus, X, mapping.mean);

```

```
    % Make sure labels are nice
21  [classes, bar, labels] = unique(labels);
      nc = length(classes);
23
    % Intialize Sw
25  Sw = zeros(size(X, 2), size(X, 2));

27    % Compute total covariance matrix
      St = cov(X);
29
    % Sum over classes
31  for i=1:nc

33        % Get all instances with class i
          cur_X = X(labels == i,:);
35
      % Update within-class scatter
37    C = cov(cur_X);
      p = size(cur_X, 1) / (length(labels) - 1);
39    Sw = Sw + (p * C);
      end
41
      % Compute between class scatter
43    Sb        = St - Sw;
      Sb(isnan(Sb)) = 0; Sw(isnan(Sw)) = 0;
45  Sb(isinf(Sb)) = 0; Sw(isinf(Sw)) = 0;

47    % Make sure not to embed in too high dimension
      if nc <= no_dims
49        no_dims = nc - 1;
          warning(['Target dimensionality reduced to ' num2str(no_dims) '.']);
51    end

53  % Perform eigendecomposition of inv(Sw)*Sb
      [P, lambda] = eig(Sb, Sw);
55
      % Sort eigenvalues and eigenvectors in descending order
57    lambda(isnan(lambda)) = 0;
    [lambda, ind] = sort(diag(lambda), 'descend');
```

```
59    P = P(:,ind(1:min([no_dims size(P, 2)])));

61    % Store mapping for the out-of-sample extension
      mapping.P = P;
63    mapping.val = lambda;
```

Listing A.1 Matlab code for LDA based feature space transformation