Mapping Based Noise Reduction

for Robust Speech Recognition

Somayeh Hosseini

Master of Science in Electrical Engineering

with emphasis on Signal Processing

Blekinge Institute of Technology

Department of Signal Processing

July 2010

Thesis Advisors: **Dr. Rolf Bardeli**

Dr. Nedelko Grbic

Thomas Winkler

Examiner: **Dr. Nedelko Grbic**

Contents

List of Figures

List of Tables

Abstract

This project aims at proposing a new noise reduction technique for speech recognition purposes. The proposed method called mapping based noise reduction is performed on the feature vectors extracted from speech signals. In this work the dimensionality reduction functionality of algorithms such as Locally Linear Embedding and Principal Component Analysis is exploited to map the corrupted speech feature vectors to their corresponding noise-free feature vectors. The feature vectors are first mapped to the lower dimensional space and in this space the nearest clean vector to each noisy vector is found, mapped back again to the original space and given as the input to the speech recognition system. This approach is examined on the speech signals with artificially added wind noise with different signal to noise ratio values and articulated by two different speakers.

Chapter 1

Introduction and Related Work

A speech recognition system that converts the human speech to text is considered robust if it is able to achieve a high recognition accuracy even under improper conditions. These improper conditions might be caused by the low quality of input speech signal due to additive environmental noise or differences in acoustic features of testing and training speech signals that are recorded in different environments or articulated by different speakers.

The speech signals we are dealing with in this thesis are a part of the project MOVEON in Fraunhofer Institute, IAIS. The goal of the project MOVEON is to develop a communication system that is able to perform well even in challenging scenarios. A good instance of these challenging scenarios is when we have speech signals recorded on motorcycles where different sources of noises are available and we are going to accomplish robust speech recognition on those speech signals. Hence, in our experiments, we use the speech data that is corrupted by the same noise types appearing on speech data recorded from a moving motorcycle such as wind noise.

One way to achieve robustness in speech recognition system is to upgrade the corrupted speech signal by reducing the disturbance. Several speech enhancement methods have been tried to date. Spectral Subtraction [7] for instance tries to compensate the effects of noise by estimating the noise power spectrum and subtracting it from the power spectrum of the corrupted speech signal. Cepstral Mean Subtraction [7], Blind Equalization [7] and adaptive filters are another noise compensation methods for speech recognition. However, most of these methods are insufficient

when faced with difficult domains and consequently the need for an alternative noise reduction method arises.

1.1 Scope of the Thesis

The scope of this thesis is presenting a new approach for noise reduction called mapping based noise reduction, applied on the feature vectors of the speech signal. In this approach, the feature vectors extracted from noisy speech signals are projected to their nearest feature vectors extracted from clean speech signals. The projected feature vectors then serve as inputs to the speech recognition system to improve its accuracy and performance. This process is carried out in the lower dimensional subspace because with the complexity of higher dimensional space, the projection of feature vectors is almost not feasible. In other words, each noisy feature vector is replaced by its nearest clean feature vector found in the lower dimensional space. Principal Component Analysis and Locally Linear Embedding are the dimensionality reduction algorithms we have used for mapping the speech feature vectors to lower dimensional space.

1.2 Overview

In chapter 2 the basics of audio signal processing including time and frequency domain properties of audio signals are discussed. Chapter 3 provides an overview of a modern automatic speech recognition system and its units. Chapter 4 describes the mapping based method for noise reduction and the dimensionality reduction algorithms we have applied. In chapter 5 the results of experiments are presented and compared. Finally, the conclusion of this work is provided in chapter 6.

Chapter 2

Principles of Speech Signal Processing

A speech signal is the air pressure amplitude generated by forcing air through the vocal cords and radiated from a speaker's mouth or nose. From mathematics point of view, it is a function $x : \mathbb{R} \to \mathbb{C}$ which is an analog or continuous-time function. In order to represent such a signal in computers we have to digitalize it. The digitization process will be described further in the following sections.

In addition to time domain representation, a speech (audio) signal can be represented in frequency domain. The frequency domain representation of a signal shows us the frequency information of the signal.

2.1 Time Domain Representation of Speech Signals

Speech signals as information-bearing signals are functions of a single independent variable, time, and are analog in nature. This means that they are functions of continuous time. Such signals can be processed using analog processors. However, an alternative, and mostly better, method for processing is digital processing using a digital computer or microprocessor. In the latter case we have to first convert the so called analog signal to digital format, that is to convert it to a finite sequence of numbers, to perform the processing digitally. The analog to digital converter or A/D converter is used for this purpose. The process of analog to digital conversion or digitization consists of 3 steps: sampling, quantization and coding

which are briefly described in the following sections. The inverse process which converts the digital signal to analog, is called D/A conversion and is necessary after processing in many cases such as speech processing. However, in the speech recognition case, there is no need for D/A conversion since the output of a recognizer is text, not speech.

2.1.1 Sampling

By taking samples of a continuous-time signal at discrete time instants, it will be converted to a discrete-time signal. This is the first step to digitization. Thus, if $x_a(t)$ is the continuous-time signal and we take samples every T seconds, then the sampled signal is given by: $x(n) = x_a(nT)$, where T is called the sampling period. Moreover, the number of samples per second $F_s = \frac{1}{T}$ (Hz), is called sampling frequency. The relationship between continuous and discrete time variables t and n is given by :

$$t = nT = \frac{n}{F_s}. \tag{2.1}$$

Accordingly, there is a relationship between the continuous time and discrete time signal frequencies F and f. It is simply proved that this relationship is :

$$f = \frac{F}{F_s}, \tag{2.2}$$

where f is called normalized frequency and $\frac{-1}{2} \leq f \leq \frac{1}{2}$. Using the above relation, we will get a range for continuous-time signal frequency:

$$\frac{-F_s}{2} \leq F \leq \frac{F_s}{2} \tag{2.3}$$

which means the maximum frequency of an analog signal that can be distinguished after sampling at $F_s = \frac{1}{T}$, is $F_{max} = \frac{F_s}{2}$. In other words, only the frequencies which lie in this range can be reconstructed after sampling. If the selected sampling frequency does not fulfill the inequality in Equation (2.3), then a distortion called aliasing distortion occurs at frequencies inside and outside this range. For a better understanding of aliasing distortion consider an analog signal with fundamental frequency F_0. If this signal is sampled at a rate $F_s < 2F_0$, then all frequencies $F_k = F_0 + kF_s$ will produce identical sample values to those of F_0 and as a result the parts of the analog signal related to these frequency contents cannot be reconstructed from their sample values. The frequencies F_k are all called aliases of F_0. Figure 2.1 shows the aliasing distortion in time domain.

Thus, to find the appropriate sampling frequency, we should apply the sampling theorem [8]. According to the sampling theorem, if the highest

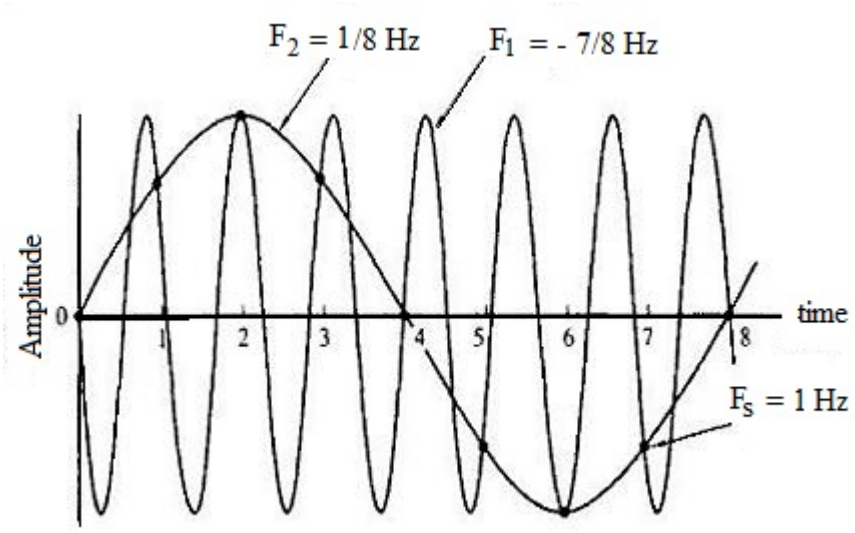

Figure 2.1: For two sinusoids with fundamental frequencies $F_1 = -\frac{7}{8}Hz$ and $F_2 = \frac{1}{8}Hz$, selecting a sampling rate of $F_s = 1Hz$ results in identical samples of two sinusoidal components; this is called aliasing distortion [8].

frequency content of an analog signal is $F_{max} = B$ (Hz) and the sampling frequency is $F_s > 2B$, then the analog signal can be accurately reconstructed from its sample values according to the following equation:

$$x_a(t) = \sum_{n=-\infty}^{\infty} x(n)\text{sinc}(2Bt - \frac{2Bn}{F_s}) \tag{2.4}$$

where

$$\text{sinc}(t) = \begin{cases} \frac{\sin(\pi t)}{\pi t} & \text{if } t \neq 0; \\ 1 & \text{if } t = 0. \end{cases}$$

The minimum allowable value of sampling frequency, $F_s = 2B$, is called nyquist rate.

Since a speech signal has frequency components up to 4 KHz, its sampling frequency is selected in the range of 8 to 16 KHz according to the sampling theorem.

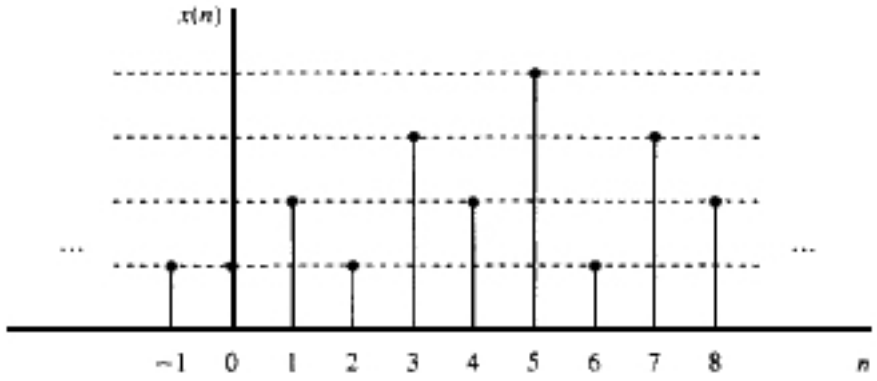

Figure 2.2: A digital signal $x(n)$ with four quantization levels [8]

2.1.2 Quantization and Coding

By sampling an analog signal it is only converted to a discrete-time signal, however, a digital signal must be both discrete in time and amplitude. Therefore, quantization is performed on a sampled signal to make it discrete-valued as well. This is done simply by converting continuous sample values to discrete levels and represent them as a finite sequence of numbers. Thus, each sample value is truncated or rounded to a predefined allowable level called quantization level. A rounding quantization is done by replacing each sample value by its nearest quantization level while truncation quantization replaces each sample value by the quantization level below it. The distance between successive quantization levels is called quantization resolution Δ. This value of resolution can simply be formulated as follows; if we have the maximum and minimum amplitude values x_{max}, x_{min}, of a sampled signal and the number of quantization levels L:

$$\Delta = \frac{x_{max} - x_{min}}{L - 1}, \tag{2.5}$$

where $x_{max} - x_{min}$, is called the dynamic range of the signal.

For the rounding quantization, the quantization error which is defined as the difference between the quantized signal and the sampled signal, $e_q(n) = x_q(n) - x(n)$, is bounded to the following range:

$$\frac{-\Delta}{2} \leq e_q(n) \leq \frac{\Delta}{2} \tag{2.6}$$

Therefore, by observing Equations (2.5), (2.6), one can simply conclude

that for a fixed value of dynamic range, increasing the number of quantization levels, increases the quantization resolution and consequently leads to a decrease in quantization error. Thus, the higher the number of quantization levels, the more accurate the quantization process.

Figure 2.2 illustrates a discrete-time quantized signal (digital signal).

Finally, coding is the representation of quantization levels by binary numbers of length b bits. 2^b binary numbers are produced by b bits and if L binary numbers are needed (L quantization levels), the following inequality must hold true to obtain the number of bits b:

$$2^b \geq L \Rightarrow b \geq \log_2 L. \tag{2.7}$$

For speech signals, usually 16 bit words are used.

2.2 Frequency domain Representation of Speech signal

In the frequency domain, a signal is represented by a sum of several sinusoidal components (frequency components). This is called the frequency analysis of the signal. For continuous-time signals, the Fourier series and the continuous Fourier transform represent the frequency components of a periodic and aperiodic signal, respectively.

A continuous-time periodic signal $x(t)$ that in Hilbert space [15] of $L^2([0,1])$ is represented in terms of its Fourier coefficients C_k, using the following expression for Fourier series:

$$x(t) = \sum_{k=-\infty}^{\infty} C_k e^{j2\pi k F_0 t} \tag{2.8}$$

where F_0 is the fundamental frequency (the lowest frequency or the first harmonic) and $T_0 = \frac{1}{F_0}$ is the period of the signal and $e^{j\theta} = \cos\theta + j\sin\theta$ is the Euler's formula ($j = \sqrt{-1}$). The Hilbert space (such as the Euclidean space) is the space of finite-energy signals and has the following definition:

$$L^2([0,1]) = \{x : [0,1] \to \mathbb{C} \mid ||x||_2 = \sqrt{\int_0^1 |x(t)|^2 dt} < \infty\}; \tag{2.9}$$

this means that the periodic signal has finite energy in one period (energy of a signal is defined as $\int_{\mathbb{R}} |x(t)|^2 dt$).

The Fourier coefficients can be obtained by the following expression:

$$C_k = \frac{1}{T_0} \int_{-\frac{T_0}{2}}^{\frac{T_0}{2}} x(t) e^{-j2\pi k F_0 t} dt. \tag{2.10}$$

Furthermore, the average power of a periodic signal $x(t)$, which has infinite energy but finite average power, is given by:

$$P = \frac{1}{T_0} \int_{-\frac{T_0}{2}}^{\frac{T_0}{2}} |x(t)|^2 dt = \sum_{k=-\infty}^{\infty} |C_k|^2; \tag{2.11}$$

this means that the total power of a periodic signal is the sum of the powers for each frequency component (harmonic) of it.

Figure 2.3 shows the power distribution among frequency components of a periodic signal $x(t)$ and it is called power density spectrum of the signal. One can observe from this figure that periodic signals have line spectra. However if the period of the signal tends toward infinity and the signal becomes aperiodic, the spectrum becomes continuous.

Thus, the Fourier transform of a finite energy (aperiodic) signal $x(t)$ is defined as:

$$X(F) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi F t} dt \tag{2.12}$$

which is a continuous function of frequency F.

The inverse Fourier transform is obtained by replacing the summation in Equation (2.8) by an integral over the frequency F:

$$x(t) = \int_{-\infty}^{\infty} X(F) e^{j2\pi F t} dF. \tag{2.13}$$

Finally, the total energy of $x(t)$, which is a finite-energy signal, is given by:

$$E = \int_{-\infty}^{\infty} |x(t)|^2 dt = \int_{-\infty}^{\infty} |X(F)|^2 dF, \tag{2.14}$$

while the energy density spectrum of $x(t)$ (the energy distribution of the signal as a function of frequency) is defined as:

$$S_{xx}(F) = |X(F)|^2; \tag{2.15}$$

therefore, the integral of S_{xx} over F is equal to total energy of signal.

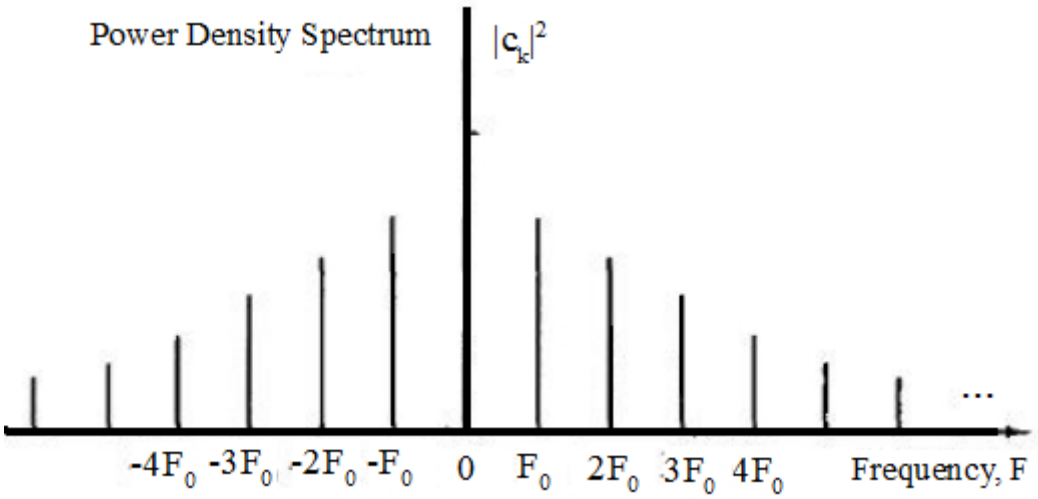

Figure 2.3: Power distribution of a continuous-time periodic signal with fundamental frequency F_0 [8].

2.2.1 Discrete-Time Fourier Transform, DTFT

In this section frequency analysis of discrete-time signals is discussed.

The Fourier transform of a continuous-time signal has an infinite number of frequency components in the range $-\infty < F < \infty$, while in the discrete-time case the frequency components are limited to the range $-\frac{1}{2} < f < \frac{1}{2}$ or equivalently $-\pi < \omega < \pi$, where $\omega = 2\pi f$ is the angular frequency. Therefore, the Fourier transform of a discrete-time signal is unique only in this range and the frequencies which lie out of this range are equivalent to the ones within the range. This means that the Fourier transform of a discrete-time signal or the discrete time Fourier transform is periodic with period 2π and it is defined as:

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}. \tag{2.16}$$

Figure 2.4 illustrates the DTFT of the following finite-length sequence:

$$x(n) = \begin{cases} 1 & 0 \leq n \leq L-1; \\ 0 & \text{otherwise.} \end{cases} \tag{2.17}$$

The inverse discrete time Fourier transform derived from the periodic

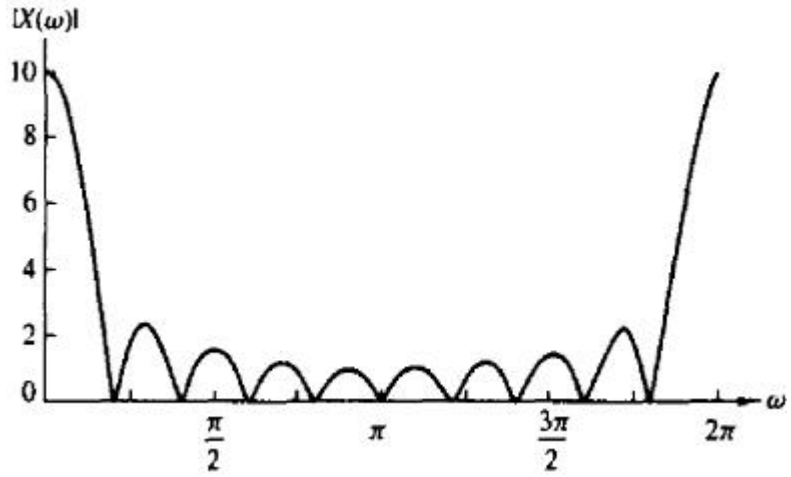

Figure 2.4: The magnitude of DTFT for the sequence in Equation (2.17) with length $L = 10$ [8].

characteristic of $X(\omega)$ is as follows:

$$x(n) = \frac{1}{2\pi} \int_{2\pi} X(\omega) e^{j\omega n} d\omega. \tag{2.18}$$

In addition, the energy of a discrete-time signal in terms of time and frequency can be simply formulated as:

$$E = \sum_{n=-\infty}^{\infty} |x(n)|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |X(\omega)|^2 d\omega, \tag{2.19}$$

and likewise the continuous-time case, energy density spectrum of the discrete-time signal is given by:

$$S_{xx}(\omega) = |X(\omega)|^2. \tag{2.20}$$

In the cases where our discrete-time signal, $x(n)$, is real, it is easily proved that $X(\omega)$ and $S_{xx}(\omega)$ will be symmetric functions of frequency. As a result,the frequency range of real discrete-time signals can be further restricted to the range $0 \leq \omega \leq \pi$.

2.2.2 Discrete Fourier Transform, DFT

As in the time-domain case, to present the spectrum of a discrete-time signal we need to digitize it first. By sampling the continuous spectrum

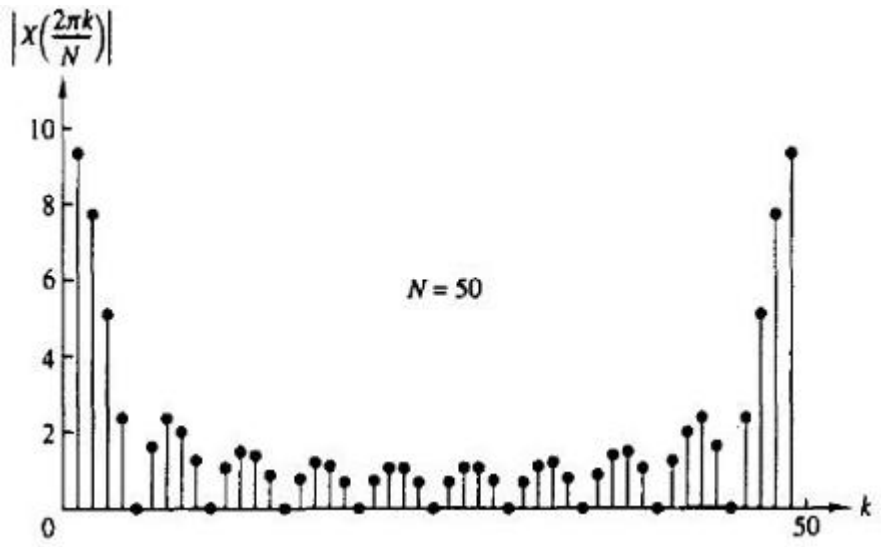

Figure 2.5: The magnitude of 50-point FFT for the sequence in Equation (2.17) with length $L = 10$ [8].

$X(\omega)$ of finite-energy sequence $x(n)$ at equally spaced frequencies, we can perform the frequency analysis of this sequence on a digital computer. This frequency domain representation of $x(n)$ is called Discrete Fourier Transform. The Discrete Fourier Transform (DFT) of a sequence $x(n)$ of length $L \leq N$ is given by:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}, k = 0, 1, ..., N-1, \tag{2.21}$$

where N is the number of frequency lines in the spectrum. You can observe in this equation that computation of the $N-$point DFT requires N^2 complex multiplications and $N(N-1)$ complex additions.

The relation for inverse DFT (IDFT) which reconstructs the sequence $x(n)$ from its frequency samples is as follows:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)e^{j2\pi kn/N}, n = 0, 1, ..., N-1. \tag{2.22}$$

In Figure 2.5 you can observe the frequency samples of the sequence in Equation (2.17).

Alternatively, viewing the Equations (2.21) and (2.22) as linear transformations of $x(n)$ and $X(k)$, leads to the matrix format representation of DFT and IDFT.

We define an $N \times N$ transformation matrix $\mathbf{W}$:

$$\mathbf{W} = \begin{pmatrix} 1 & 1 & 1 & ... & 1 \\ 1 & W & W^2 & ... & W^{N-1} \\ 1 & W^2 & W^4 & ... & W^{2(N-1)} \\ . & . & . & & . \\ . & . & . & ... & . \\ . & . & . & & . \\ 1 & W^{N-1} & W^{2(N-1)} & ... & W^{(N-1)(N-1)} \end{pmatrix}, \tag{2.23}$$

where $W = e^{-j2\pi/N}$, to obtain the following linear expressions for the $N-$point DFT:

$$\mathbf{X} = \mathbf{W}\mathbf{x}, \tag{2.24}$$

and IDFT:

$$\mathbf{x} = \frac{1}{N}\mathbf{W}^*\mathbf{X}. \tag{2.25}$$

The $N-$dimensional vectors $\mathbf{X}$ and $\mathbf{x}$ in the above equations are the vectors of signal sequence $x(n)$ and its frequency samples $X(k)$, respectively. The term $\mathbf{W}^*$ refers to the complex conjugate of the transformation matrix.

By applying symmetry and periodicity properties of the term $W = e^{-j2\pi/N}$, the complexity of DFT computation can noticeably be reduced. This leads us to a number of algorithms called Fast Fourier Transform (FFT) algorithms, that compute DFT efficiently. Assuming that N is not a prime number and can be factorized to a product of two integers ($N = LM$), one of FFT algorithms, for instance, divides the $N-$point DFT to a number of smaller DFTs ($M-$point DFTs) to reduce the computation cost. As a result, the computational complexity is reduced to $N(M+L+1)$ complex multiplications and $N(M+L-2)$ complex additions. In the cases where N is a power of 2, we will have an $N\log_{10}(N)$-complexity algorithm for computation of DFT.

Chapter 3

Automatic Speech Recognition, ASR

Speech technology includes several subfields; amongst them are Speaker Identification (extracting a person's identity from his/her voice), Speech Synthesis (converting text to voice in order to produce artificial human speech) and Speech Recognition (converting speech to text). In this section we try to introduce the newest techniques for Automatic Speech Recognition.

An automatic speech recognition system is used for labeling the human speech and it has a variety of applications such as voice dialling, call routing, data entry and automatic control of home appliances. Figure 3.1 gives you an overview of a speech recognition system and its units. A modern speech recognition system is composed of 3 main components:

- Feature Extraction: Extracting feature vectors from digitized speech signal.
- Training Hidden Markov Models: Training acoustic models for speech feature vectors using Hidden Markov Models.
- Recognition: Classifying the test speech vectors using the acoustic models obtained in previous step.

In the rest of this section we first introduce phonetics and then explain each of the above components.

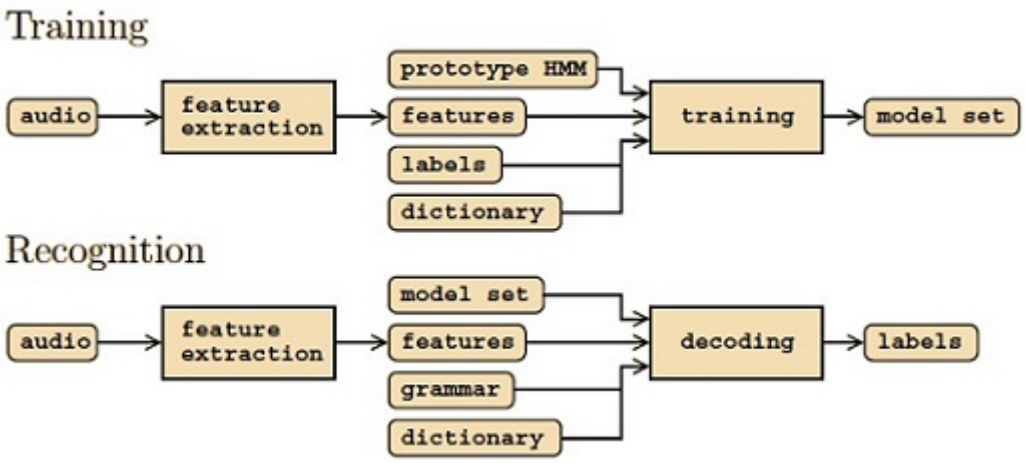

Figure 3.1: Components of a Speech Recognition System [10]

3.1 Speech Production

Study of human sound production and properties is called phonetics. In this section two major branches of phonetics, Articulatory Phonetics and Acoustic Phonetics are discussed. Articulatory Phonetics focuses on how human organs are involved in speech production while Acoustic Phonetics studies the acoustic properties of speech.

3.1.1 Articulatory Phonetics

Speech is composed of phonemes as its smallest units. There are two types of phonemes called vowels and consonants and a combination of them makes each spoken word. This section discusses the production of vowels and consonants as units of a spoken word. Figure 3.2 shows the human body organs that produce and modulate the speech (vocal tract). The sound source as you observe in this figure, is the airflow provided by lungs and diaphragm while breathing (Respiration phase). This airflow then passes through a structure called Larynx which plays an important part in sound production. The structure of Larynx as you can observe in Figure 3.3 contains two bands of muscle and tissue called vocal cords and the gap in between them called glottis. This phase of sound production with Larynx's contribution is defined as phonation. Depending on the position of vocal cords and the rate of airflow, different phonations are obtained. If the vocal cords are completely closed, a kind of consonantal

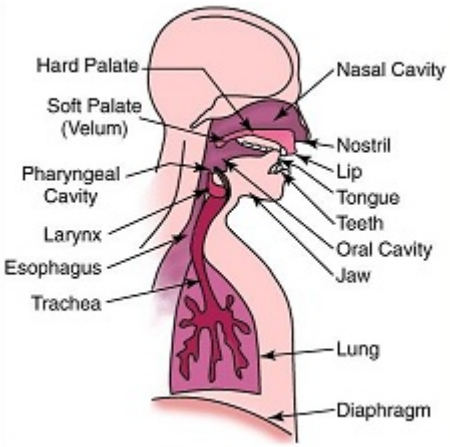

Figure 3.2: Human organs which make speech (Vocal tract) [9]

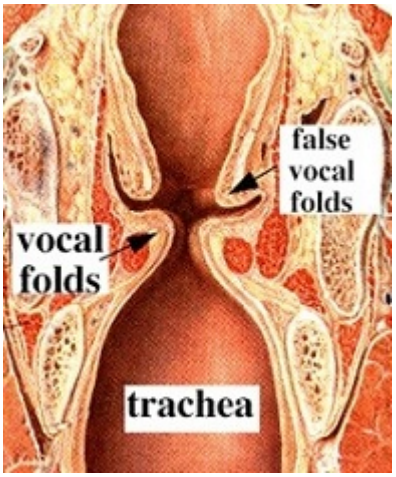

Figure 3.3: The structure of Larynx [3]

sound is produced named glottal stop. If the airflow causes a vibration in vocal cords while passing through them, the resulting noise in the air produces a voicing sound (vowels and voiced consonants) such as [b], [d], [v], [i], [r], [z]. On the other hand, if the vocal cords are completely apart (wide-open glottis), the air can freely pass through them and no noise will be produced in the air. The resulting sound in this case is a voiceless sound (voiceless consonants) such as [p], [t], [k], [s], [f], [sh], [th], [ch]. Moreover, if the air flows so rapidly through the wide-open glottis, a whisper-like sound such as [h] (and some whispered vowels) becomes the result.

The last phase of speech production is called articulation and it refers to

the places in the human vocal tract which are mostly constricted when a sound is produced (place of articulation) and how the organs such as tongue or lips contribute in producing the sounds (manner of articulation). Since the vowels are produced with no noise from constriction in the vocal tract, these two terms are only used for consonants. The articulatory features of vowels that classify the quality of vowels are as follows:

- Height: Vowel height determines the vertical position of the tongue. Based on this feature, vowels are classified to high vowels (the tongue located high in the mouth) such as [i] and [u] and low vowels (the tongue located low in the mouth) such as [a].
- Backness: Vowel backness refers to the position of the tongue relative to the back of the mouth. For this feature we will have front vowels (the tongue located forward in the mouth) such as [i] vs back vowels (the tongue located backward in the mouth) such as [u].
- Roundedness: It determines whether the lips are rounded or not during articulation and classifies the vowels to rounded or unrounded.
- Nasalization: It determines whether the air passes through the nose during articulation and classifies the vowels to nasal and oral.

A combination of these features determines the quality of each vowel sound.

The following are the most important places of articulation for classification of consonants:

- Labial (lips)
- Coronal (tip or blade of tongue)
- Dorsal (back of tongue)

Furthermore, the major manners of articulation for the consonants are as follows:

- Stop: All of articulators are closed and air cannot escape through the mouth.

- Oral stop (plosive): Air cannot escape through the nose ([p], [t], [k], [b], [d], [g]).
- Nasal stop: The oral cavity is closed and air escapes through the nose ([m], [n], [ng]).
- Fricatives: A noisy airflow exists at the place of articulation and a hissing sound is produced ([f], [v], [s], [z], [th], [dh]).
- Approximant: There is very little closure of articulators and consequently no noise is involved ([y], [r]).
- Lateral approximant: Airflow is obstructed along the center of oral tract and the sound is pronounced by sides of tongue ([l]).
- Tap or Flap: The oral cavity is closed temporarily by tongue ([dx]).
- Affricate: Oral stop that is followed by a fricative instantly ([ch], [jh]).

3.1.2 Acoustic Phonetics

This section describes the acoustic properties of speech signals. The amplitude (loudness) of a speech signal is defined as the air pressure in different time instants. A complex speech signal is resulted from adding several simple sinusoids with different frequencies. A spectrum shows each frequency content (pitch) of the speech signal and its corresponding amplitude. In a spectrum diagram the horizontal axis represents the frequency (in Hz) and the vertical axis represents the amplitude (in dB). The spectrum of a sound wave acts like a prism that breaks the light into colors; similarly, it breaks a sound wave to its frequency components. This helps us to find out the frequency patterns of different vowels and consonants (or phonemes in general).

The human speech production system can be simulated by a system consisting of excitation and articulation units. In the excitation (source) unit a switch is used to decide between a tone generator for the voiced sounds and a noise generator for the unvoiced sounds (simulating the vocal cords task). Then in the articulation (filter) unit a variable filter is used to articulate the sounds and make the final speech (simulating the mouth and nose tasks). Thus, the spectrum of a speech waveform is composed of the actual frequencies produced by the source mounted on an envelope that is the frequency response of the filter. The peaks of the filter's (vocal tract's) frequency response (resonance frequencies of the filter) are called formants

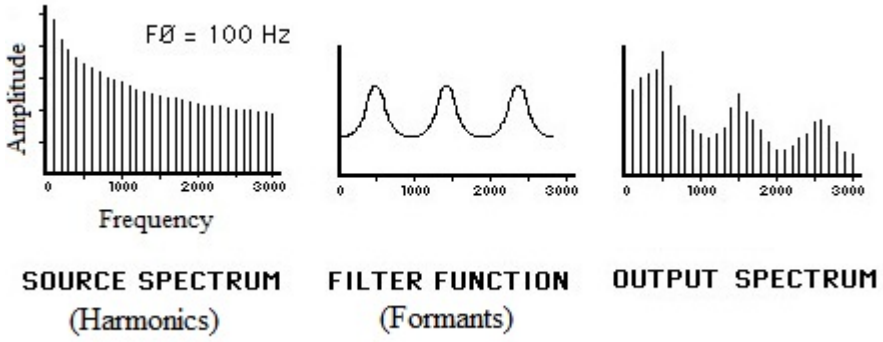

Figure 3.4: The spectrum of a speech waveform is a combination of harmonics and formants [14]

[6]. The formants as demonstrated in Figure 3.4 make the overall shape (envelope) of the spectrum while the harmonics are the frequency contents of it. Different vowels (with different formants) can have the same harmonics and same vowels (with similar formants) can be spoken with different harmonics.

3.2 Feature Extraction

Feature extraction is a kind of dimensionality reduction method used in pattern recognition tasks to reduce the amount of data (in order to avoid redundancy), strengthen the variable parts that improve discrimination of patterns and attenuate the variable parts that worsen discrimination of patterns. As for speech recognition, the variable parts that may disorder the speech recognition outcome are speaker variabilities and acoustic and channel distortions. Through this process the digitized speech signal is transformed to a sequence of equally spaced (in time) feature vectors. The speech signal can be considered stationary in the intervals covered by each feature vector (10 ms intervals).

The features that we use for speech recognition are called Mel Frequency Cepstral Coefficients (MFCC features). In this section we explain how MFCC feature vectors are derived from the digitized speech signal.

The first step in MFCC feature extraction is to perform short-time Fourier transform (STFT) on the digitized speech signal. The short-time Fourier

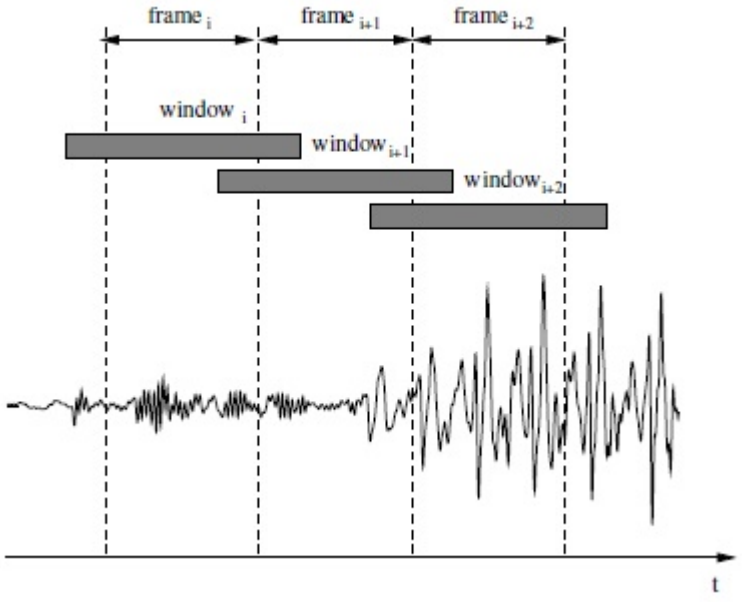

Figure 3.5: Overview of short-time Fourier transform

transform determines the temporal frequency contents (time localization of frequency contents) of a signal which is favorable in our case and it is a linear transform. Since we are dealing with digital speech signals, we only describe the discrete STFT. Figure 3.5 shows the process of calculating discrete STFT. First, the digitized speech signal $x(n)$ is split up into finite-length frames by multiplying the speech signal by a discrete finite-length window $w(n)$. For reducing the boundary noise, we should extract overlapping frames. Then FFT is calculated for each overlapping frame according to the following equation:

$$\mathbf{STFT}\{x(n)\} = X_i(k) = \sum_{n=-\infty}^{\infty} x(n)w(n-i)e^{-j2\pi\frac{k}{N}n}, \tag{3.1}$$

where $k = 0, 1, ..., N-1$, i denotes the frame index and N is the number of frequency bins for each frame that is equal to the frame length.

The squared magnitude of the resulting complex sequence is defined as the spectrogram of the signal $x(n)$ and it is obtained by the following equation:

$$\text{spectrogram}\{x(n)\} = |X_i(k)|^2. \tag{3.2}$$

The plot of spectrogram (with horizontal and vertical axes representing time and frequency respectively) shows how the frequency contents of a

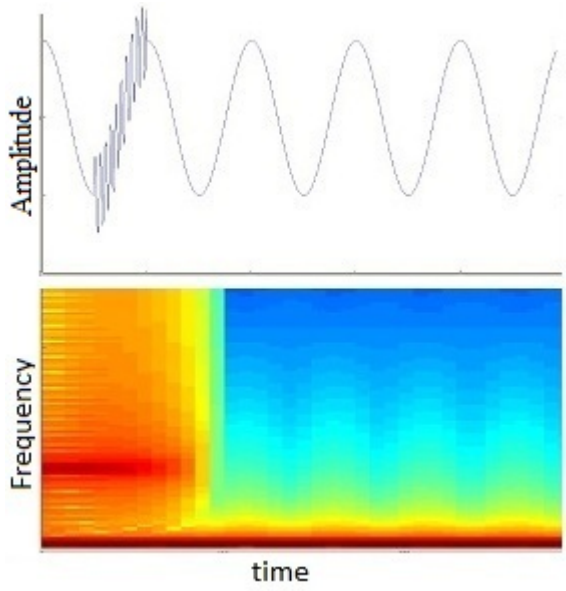

Figure 3.6: A digital signal and its spectrogram

signal change during the time (time-frequency graph). As an example, in Figure 3.6 the spectrogram of a signal that has varying frequency contents in different time intervals is given. One can observe from this figure that the spectrogram shows more energy of the signal in the interval where it has higher frequencies.

There are three factors that have effect on the quality of STFT: window type, window length and step size value. In Figure 3.7 you can observe two commonly used window types for frame extraction and the spectrograms that are obtained using these window functions. Hamming window is the most appropriate one since it prevents the discontinuities at window boundaries. The equation for hamming window is given by:

$$w(n) = \begin{cases} 0.54 - 0.46\cos\frac{2\pi n}{N-1} & 0 \leq n \leq N-1; \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

The window length controls the time-frequency resolution of the spectrogram. Long windows lead to high frequency resolution (for detection of formants with short spectral distance) and low temporal resolution while short windows result in low frequency resolution and high temporal resolution (for detection of fast variations such as short plosive). Therefore, having high frequency resolution and temporal resolution at the same time is not possible and even not desirable. However, for au-

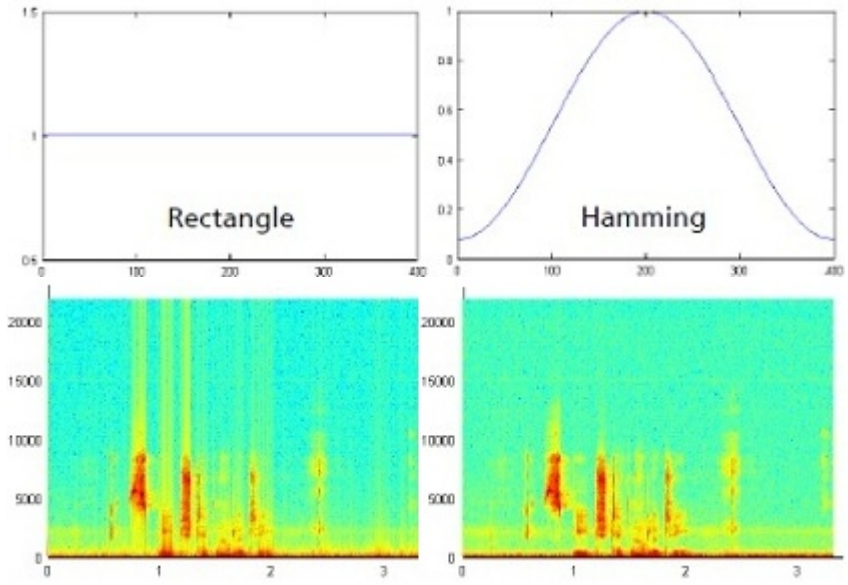

Figure 3.7: Common window functions used for STFT calculation and their effect on spectrogram

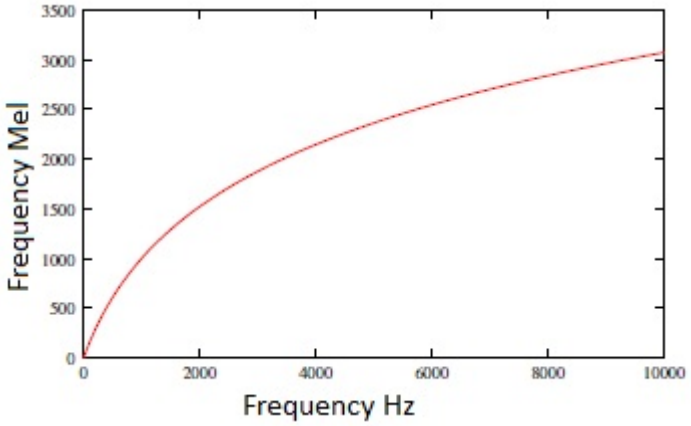

Figure 3.8: Mel scale versus Hertz scale [16]

tomatic speech recognition the window length of 400 samples (at 16 kHz sampling rate) which leads to frequency resolution of $\Delta f = 40$ Hz and temporal resolution of $\Delta t = 25$ ms along with the window shift value of 10 ms, works well.

So far we have obtained the temporal spectrum of the digital speech signal

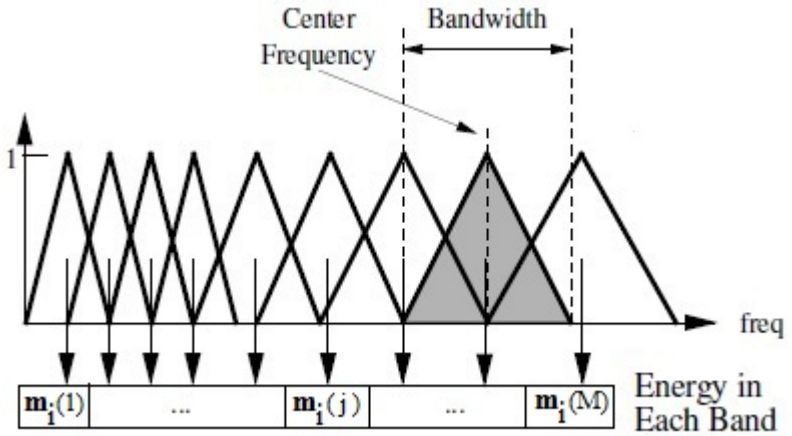

Figure 3.9: Mel-frequency triangular filter banks [17]

and now we should transform the frequency scale to the mel scale and take the logarithm of squared amplitude of spectrum. The mel is a non-linear frequency scale for sound pitches that are perceived by the listeners to have equal distances from each other. It models the human auditory system due to the fact that the human ear does not perceive the pitches of sound linearly but logarithmically. This is to our advantage because it improves the human ear sensitivity in lower frequencies and smooths the spectrum. Moreover, experiments reveal that mapping the frequencies of speech signal to mel scale improves the speech recognition performance. The following equation is used for converting the Hertz frequency to its corresponding Mel scale:

$$mel = 2595 \log_{10}(1 + \frac{f}{700}), \tag{3.4}$$

and Figure 3.8 demonstrates this relation. As you see in this figure, the high frequencies are compacted in the mel scale while the low frequencies are expanded. The reference point between the Hertz and Mel scales is 1000 Hz = 1000 Mel and the relation is approximately linear below 1 KHz and logarithmic above 1 KHz. For converting the frequency scale to Mel, we can apply Mel-frequency triangular filter banks [11] as exhibited in Figure 3.9. The filters are bandpass and they model auditory spectral bands. The center frequencies of triangles are calculated according to the Mel scale (Equation (3.4)) and as a result the bandwidth and distance of triangles increase with frequency. For each of the triangular filters, the squared magnitude of each frame's spectrum is multiplied by the filter gain and the results are summed up. The logarithm of the resulting values present the scaled power spectrum of each frame. The following formula

expresses this step:

$$X_i'(j) = \log \sum_{k=0}^{N-1} H(j,k)|X_i(k)|^2, \{j = 1, ..., M\}. \tag{3.5}$$

In Equation (3.5), M represents the number of filter banks ($M < N$), $X(k)$ is the spectrum of a frame (short-time spectrum) and $H(j,k)$ is the jth filter bank. The reason for taking the logarithm of squared amplitude Mel-spectrum is that the loudness of a sound is approximately perceived in a logarithmic manner.

The last step of MFCC feature extraction is to calculate the Discrete Cosine Transform (DCT) of the Mel log power spectrum in order to decorrelate the Mel-spectrum vectors which are highly correlated. The DCT is nearly similar to DFT; the only difference between them is that DFT uses both sine and cosine functions (complex exponentials) to calculate Fourier transform whereas DCT uses only cosine function. The following equation formulates the last step of MFCC feature extraction:

$$m_i(l) = \sum_{j=1}^{M} X_i'(j) \cos\left(l\frac{\pi}{M}(j - \frac{1}{2})\right), \{l = 1, ..., M\}, \tag{3.6}$$

and it can be expressed in matrix form as follows:

$$\mathbf{m_i} = \text{DCT}(\mathbf{X_i'}), \tag{3.7}$$

where $\mathbf{m_i}$ is the final M-dimensional feature vector. The resulting vector is called Mel power Cepstrum since we have taken the Fourier transform of log-Mel power spectrum. The word Cepstrum refers to the Fourier transform of log spectrum and it is derived from reversing some letters in "spectrum". The independent variable of the cepstrum is called queferency. Figure 3.10 compares the plots of Hertz spectrum, Mel spectrum and Mel cepstrum.

The feature vectors $\mathbf{m_i}$ are 13-dimensional vectors called static features. The dynamic features are gained by taking the first and second order derivatives of short-time energy of the signal resulting in two more 13-dimensional vectors. Thus, each frame of speech signal is finally represented by a 39-dimensional feature vector; although, we only need the static features (the first 13 MFCCs) in our experiments. Figure 3.11 presents a summary of the whole process of MFCC feature extraction. The MFCC feature vectors are extracted for both training and testing data in our experiments.

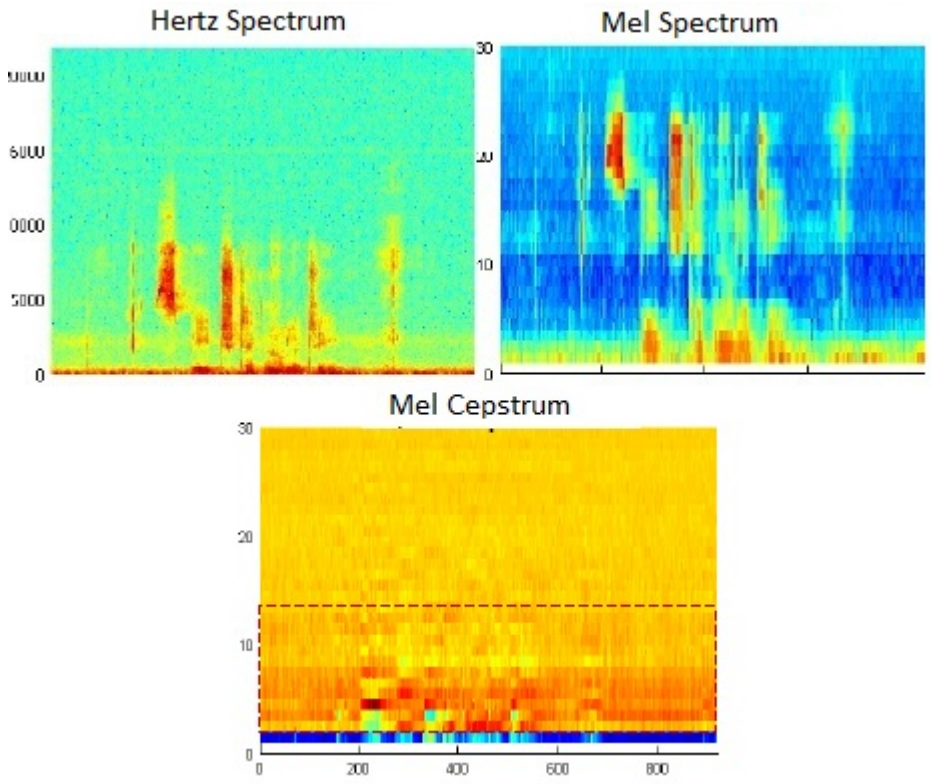

Figure 3.10: A graph example of Hertz spectrum, Mel spectrum and Mel cepstrum

3.3 Training

In the modern speech recognition systems, the acoustic models are trained by Hidden Markov Models [12]. In this section we briefly describe HMMs and its application in speech recognition. First, we introduce the HMM and its elements and 3 basic problems for it. Then we focus on the functionality of HMM for training acoustic models and labeling the test speech signals.

3.3.1 An Introduction to Hidden Markov Models

HMM is a statistical model of sequences of events which is an extension of Markov chains. In the ordinary Markov chains the states are visible while in a HMM only the outputs (a set of observation sequences) are visible

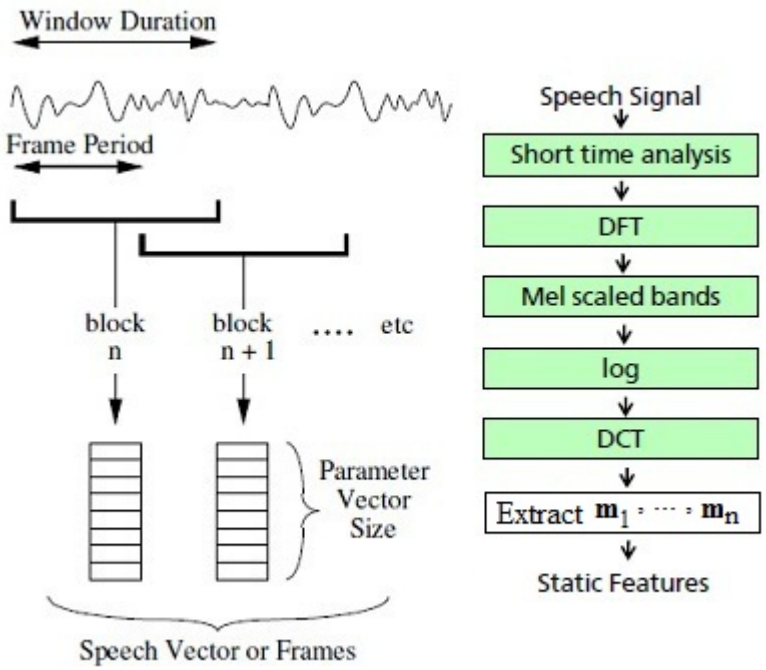

Figure 3.11: A summary of MFCC feature extraction [17]

and the states are hidden. In a first order Markov Model we assume that the probability of an event depends on its previous event only. HMMs are widely used for recognition of time-varying pattern sequences such as speech feature vectors. The elements of a first order HMM illustrated in Figure 3.12 are as follows:

- A set of N states: $S = \{s_1, s_2, ..., s_N\}$ that are not visible (denoted by circles in Figure 3.12). Only one state is active at each time instant.
- A set of K possible observation vectors: $\mathbf{V} = \{\mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_K}\}$; the observation vectors are selected from set $\mathbf{V}$ ($\mathbf{o_t} \in \mathbf{V}$).
- The probabilities of state transitions (meaning change from one state to another at equally spaced discrete times and denoted by arcs in Figure 3.12) which is a $N \times N$ matrix $\mathbf{A}$ with $a_{ij} = P(q_t = s_j | q_{t-1} = s_i)$, where $q_t \in S$ represents the state at time t and $\sum_{j=1}^{N} a_{ij} = 1$.
- The probabilities of observation vectors represented by a $N \times K$ matrix $\mathbf{B}$ with $b_{jk} = P(\mathbf{o_t} = \mathbf{v_k} | q_t = s_j)$, where $\mathbf{o_t} \in \mathbf{V}$ represents the observation at time t and $\sum_{k=1}^{K} b_{jk} = 1$.

- The initial state probability represented by a $N-$dimensional vector $\boldsymbol{\pi}$ with $\pi_i = P(q_1 = Si)$ where $\sum_{i=1}^{N} \pi_i = 1$.

A HMM denoted by $\boldsymbol{\lambda}$ is represented by its known parameters: $\boldsymbol{\lambda}(\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$.

The three basic questions we may face in a HMM are:

- **Evaluation Problem:** What is the probability that an observation sequence $\mathbf{O} = \mathbf{o_1}, \mathbf{o_2}, ..., \mathbf{o_T}$ is produced by a given model $\boldsymbol{\lambda}$ or mathematically what is the probability $P(\mathbf{O}|\boldsymbol{\lambda})$ given $\mathbf{O}$ and $\boldsymbol{\lambda}$?

  Solution: Forward algorithm [12]

- **Uncovering Problem:** What is the most likely state sequence $Q = q_1, q_2, ..., q_T$ that can produce an observation sequence $\mathbf{O} = \mathbf{o_1}, \mathbf{o_2}, ..., \mathbf{o_T}$ by a given model $\boldsymbol{\lambda}$?

  Solution: Backward algorithm [12]

- **Learning Problem:** For a given observation sequence $\mathbf{O} = \mathbf{o_1}, \mathbf{o_2}, ..., \mathbf{o_T}$ of length T, how should we set the parameters of the model $\boldsymbol{\lambda}(\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$ to maximize the probability $P(\mathbf{O}|\boldsymbol{\lambda})$?

  Solution: Viterbi algorithm for training [12]

The learning and evaluation problems are similar to the training and recognition phases of an automatic speech recognition system.

3.3.2 Training Acoustic Models

The goal of a speech recognition system is to do a mapping between a sequence of speech vectors and its corresponding symbol. The speech signal is either related to a single symbol (word) selected from a fixed vocabulary or a sequence of symbols (sentences). In the former case, we will have isolated word (word level) recognition while the latter case is called continuous speech (phoneme level) recognition. In the isolated word recognition, we have to generate a model for each word in the vocabulary using enough training samples. Consequently, words that do not appear on the training set cannot be recognized. Although isolated word recognition sounds artificial, it has a variety of practical applications.

If we represent the input speech waveform as a sequence of observation vectors $\mathbf{O} = \mathbf{o_1}, \mathbf{o_2}, ..., \mathbf{o_T}$ (where $\mathbf{o_t}$ is the observation vector at time t), then computing the following probability will solve the isolated word recognition problem:

$$\arg\max_i \{P(w_i|\mathbf{O})\}, \tag{3.8}$$

where w_i is the $i-$th word in the vocabulary. The probability in Equation (3.8) cannot be computed directly. Hence, by using the Bayes' rule we turn it to the following computable probability:

$$P(w_i|\mathbf{O}) = \frac{P(\mathbf{O}|w_i)P(w_i)}{P(\mathbf{O})}, \tag{3.9}$$

that means if the prior probabilities $P(w_i)$ are known, the most likely word corresponding to the observation sequence $\mathbf{O}$ is only related to the probability $P(\mathbf{O}|w_i)$. Therefore, solving the isolated word recognition problem is simplified to computation of the probability $P(\mathbf{o_1}, \mathbf{o_2}, ...|w_i)$. By generating a Markov Model for each word using spoken examples of it, we can replace the computation of probability $P(\mathbf{O}|w_i)$ by the estimation of the Markov Model parameters. In other words, if $\boldsymbol{\lambda_i}$ represents the Markov model of the word w_i, the following relation will hold true:

$$P(\mathbf{O}|w_i) = P(\mathbf{O}|\boldsymbol{\lambda_i}), \tag{3.10}$$

meaning that the isolated word recognition problem is solved by finding the parameters of the Markov models for each word w_i. As indicated in Figure 3.12, the sequence of observation vectors $\mathbf{O}$ corresponding to each word is assumed to be produced by a Markov model. At each time unit the state of the model is changed and when the state j is entered at time t, a vector $\mathbf{o_t}$ is produced by the probability density $b_j(\mathbf{o_t})$. Sometimes the system stays at the same state for several time units and as a result more than one vectors are produced at the same state and by the same probability density function. The transition between states is also a discrete probability denoted by a_{ij}. The state sequence that generates vectors $\mathbf{o_1}$ to $\mathbf{o_6}$ in Figure 3.12 is $Q = q_1, q_2, q_2, q_3, q_4, q_4, q_5, q_6$. Given this state sequence, the probability $P(\mathbf{o_1}, ..., \mathbf{o_6}, Q|\boldsymbol{\lambda})$ can be computed simply by multiplying the transition probabilities a_{ij} and the output probabilities $b_j(\mathbf{o_t})$. In a hidden Markov model, however, the states are hidden and we have no information of the state sequence Q. Therefore, the probability $P(\mathbf{O}|\boldsymbol{\lambda})$ is calculated by finding the most probable state sequence and multiplying its corresponding transition and output probabilities assuming that we know the parameters of the model a_{ij} and $b_j(\mathbf{o_t})$.

The training phase of a speech recognition system consists of finding the HMM parameters for each word. The models for each vocabulary word

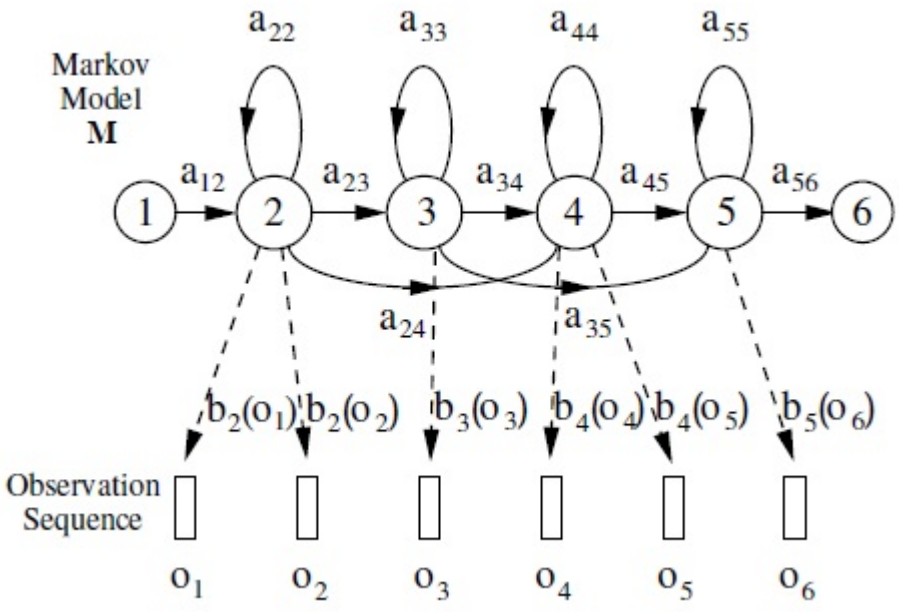

Figure 3.12: An example of a Markov Model $\boldsymbol{\lambda}$ [17]

is produced using enough spoken examples of that word. The parameters of the models are estimated by the Baum-Welch Re-Estimation algorithm [17]. These parameters are the transition probabilities a_{ij} and the means μ_j and variances Σ_j for each of the output probabilities $b_j(\mathbf{o_t})$. The distribution functions of output probabilities are continuous Gaussian Mixture densities.

In continuous speech recognition our models would be trained based on phonemes rather than words; therefore, even the words that are not inside the training set can be recognized. Moreover, since we have only a limited number of phonemes, we would have enough training samples for our models. However in continuous speech recognition, it is so difficult to specify the boundaries between symbols from the speech signal. In this case the HMMs are connected to form the continuous speech models and 3 states are considered for each phoneme.

3.4 Recognition

The recognition phase of an ASR system is to estimate the maximum value of $P(\mathbf{O}|\boldsymbol{\lambda_i})$ after training the HMMs for each word (in isolated word recognition case) / sentence (in continuous speech recognition). The maximum value of these probabilities (as mentioned in section 3.3.2) is calculated based on the most likely state sequence using Viterbi decoding algorithm [17]. An extension of the Viterbi algorithm is used for continuous recognition as well. The word (sentence) w_i corresponding to the model which produces the maximum likelihood $P(\mathbf{O}|\boldsymbol{\lambda_i})$ will be then the recognized word (sentence).

Chapter 4

Mapping Based Noise Reduction

This chapter mainly focuses on linear and nonlinear dimensionality reduction techniques for the purpose of noise reduction. In many speech recognition problems, environmental and natural noise corrupt the speech data. In order to perform robust speech recognition in presence of such noise, one has to first reduce the noise. Several noise reduction methods have been used so far to improve the speech recognition in noisy environments, amongst them are Spectral Subtraction, Cepstral Mean Subtraction and Blind Equalization. Although, in difficult domains these methods prove to be insufficient.

Another method, however, is the mapping based method which maps the noisy feature vectors extracted from a corrupted speech signal to corresponding clean feature vectors. In other words, it substitutes each sample from a noisy data set (noisy feature vectors) $\mathbf{X_n}$ by a sample from a clean data set (clean feature vectors) $\mathbf{X_c}$ which best describes its features.The mapping, however, is done in a low dimensional space due to the fact that in the lower dimensional space, the distances between vectors are calculated more accurately and consequently the most accurate substitution of the noisy feature vector with highest similarity to it, is found. For a better understanding of this, you can observe Figure 4.1. One can observe from this figure that the nonlinearity of high dimensional space, leads to non accurate measurements of distance between vectors and as a result the substitution result is not as expected, while, mapping to low dimensional space, results in a submanifold with more linear characteristics and as a result higher precision in distance measurement. Moreover, the hope

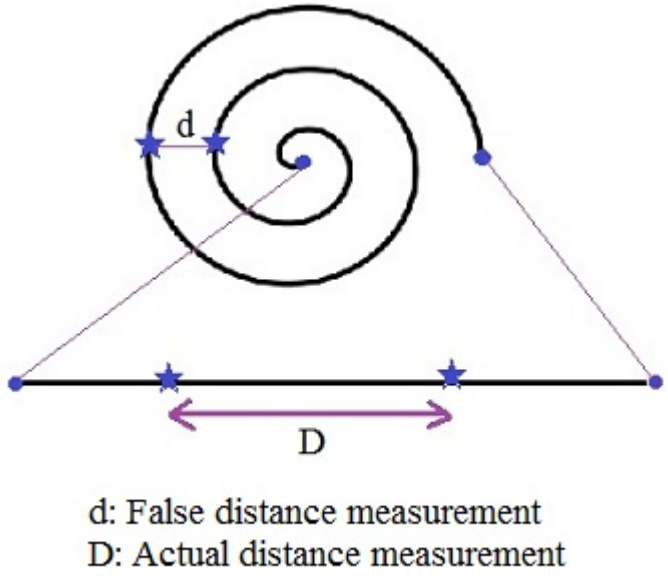

Figure 4.1: An example that shows how mapping to lower dimensional space leads to more accurate distance measurement and as a result more precise vector replacement

is that some dimensions representing noise are dropped in the dimensionality reduction. This nonlinear dropping of noise dimensions makes the process conceptually and mathematically different from techniques like multi-conditional training.

In the following sections two dimensionality reduction methods are described in detail: Principal Component Analysis as a linear method and Locally Linear Embedding as a nonlinear method. Then, a standard $k-$means clustering algorithm is discussed and finally the whole noise reduction procedure is presented in last section.

4.1 Locally Linear Embedding, LLE Algorithm

Locally Linear Embedding [5] as an unsupervised manifold learning algorithm, performs a nonlinear mapping from high-dimensional to low-dimensional space. Some of the applications of LLE are Speech Analysis, Feature Extraction for use in Speech Recognition, Speech data visualization, Phone Classification, and Noise Reduction.

The idea behind LLE is that the neighbouring points in high dimensional space, remain adjacent in low dimensional space as well. In other words, mapping through LLE preserves the local neighbourhood in the low dimensional space. Therefore, LLE aims at finding a low dimensional embedding which has this property that if two points are neighbors or co-located to each other in high dimensional space, they will definitely be neighbours in low dimensional space as well. This means that LLE maintains the local neighbourhood of high dimensional space in low dimensional space.

Suppose $\mathbf{X} : \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}\}$, $\mathbf{x_i} \in R^D$ is the high-dimensional data set lying on a nonlinear manifold. LLE maps this set of data to a low dimensional data set $\mathbf{Y} : \{\mathbf{y_1}, \mathbf{y_2}, ..., \mathbf{y_n}\}$, $\mathbf{y_i} \in R^d, d < D$ in two steps : firstly, finding K nearest neighbours of each sample $\mathbf{x_i}$ and calculating the reconstruction weights of each sample based on it's nearest neighbours and secondly, finding the low dimensional mappings $\mathbf{y_i}$ for each $\mathbf{x_i}$ which minimizes a function of the reconstruction weights.

To obtain the reconstruction weights based on nearest neighbours of $\mathbf{x_i}$, the following function is minimized:

$$\epsilon_1(\mathbf{W}) = \sum_{i=1}^{n} |\mathbf{x_i} - \tilde{\mathbf{x}_i}|^2 = \sum_{i=1}^{n} |\mathbf{x_i} - \sum_{j=1}^{K} w_j^{(i)} \mathbf{x_{N(j)}}|^2, \tag{4.1}$$

$$\tilde{\mathbf{x}_i} = \sum_{j=1}^{K} w_j^{(i)} \mathbf{x_{N(j)}},$$

where $\mathbf{x_{N(j)}}$ represents the $j-$th nearest neighbour of $\mathbf{x_i}$. The function ϵ_1 shows the difference between original and reconstructed values of $\mathbf{x_i}$ and thus the error of reconstruction. With the optimal value of weights (minimum value of ϵ_1), $\mathbf{x_i}$ is reconstructed most accurately by it's K nearest neighbours $\mathbf{x_{N(1)}}...\mathbf{x_{N(K)}}$. Then for only one vector $\mathbf{x_i}$, above equation is:

$$\epsilon_1^{(i)} = \sum_{j=1}^{K} \sum_{m=1}^{K} w_j^{(i)} w_m^{(i)} Q_{jm}^{(i)}, \tag{4.2}$$

$$Q_{jm}^{(i)} = (x_i - x_{N(j)})^T (x_i - x_{N(m)}),$$

where $\sum_{j=1}^{K} w_j^{(i)} = 1$ and $\mathbf{Q}^{(i)}$ contains the inner products of neighborhood vectors for $\mathbf{x_i}$. Finally by solving Equation (4.2) and setting $\mathbf{R}^{(i)} = (\mathbf{Q}^{(i)})^{-1}$, the optimal reconstruction weights are calculated:

$$w_j^{(i)} = \frac{\sum_{m=1}^{K} R_{jm}^{(i)}}{\sum_{p=1}^{K} \sum_{q=1}^{K} R_{pq}^{(i)}}. \tag{4.3}$$

However, since $rank(\mathbf{Q}^{(i)}) = min(D, K)$, for $K > D$, $\mathbf{Q}^{(i)}$ is not invertible and as a result a regularization parameter r should be added before inversion of $\mathbf{Q}^{(i)}$:
$\mathbf{R}^{(i)} = (\mathbf{Q}^{(i)} + r\mathbf{I})^{-1}$ ($\mathbf{I}$ is the unity matrix with diagonal elements equal to 1 and other elements equal to 0), otherwise the eigenanalysis algorithm for finding the inverse of $\mathbf{Q}^{(i)}$ does not converge.

Now that the reconstruction weights are calculated, low-dimensional mappings $\mathbf{y_i}$ can be calculated by minimizing the following function of these weights:

$$\epsilon_2(\mathbf{Y}) = \sum_{i=1}^{n} |\mathbf{y_i} - \sum_{j=1}^{K} w_j^{(i)} \mathbf{y_{N(j)}}|^2. \tag{4.4}$$

This equation can be expressed in matrix form as follows:

$$\epsilon_2 = \sum_{i=1}^{n} \sum_{j=1}^{n} M_{ij} \mathbf{y_i}^T \mathbf{y_j} = trace(\mathbf{YMY^T}), \tag{4.5}$$

where $\mathbf{W}$ is an $n \times n$ sparse matrix of reconstruction weights with $W_{i,N(j)} = w_j^{(i)}$ and $\mathbf{M}$ is an $n \times n$ matrix wich is defined as $\mathbf{M} = (\mathbf{I} - \mathbf{W})^\mathbf{T}(\mathbf{I} - \mathbf{W})$ and finally $\mathbf{Y}$ is the matrix containing low dimensional vectors $\mathbf{y}_i$ in its columns. In order to find the optimal solution, Lagrange Multipliers method is used. The Lagrange equation is obtained by a combination of Equation (4.5) and a constraint applied to the covariance matrix of $\mathbf{Y}$ to make it equal to identity: $\frac{1}{n}\mathbf{YY^T} = \mathbf{I}$. Thus, the derivative of this Lagrange equation is set to zero to lead us to the final equation: $(\mathbf{M} - \mathbf{\Lambda})\mathbf{Y}^T = \mathbf{0}$ where $\mathbf{\Lambda}$ is a diagonal matrix which contains Lagrange multipliers as diagonal elements. The solution to this equation obviously consists of the eigenvectors of $\mathbf{M}$. However, not all of these eigenvectors but only the ones which correspond to the $(d + 1)$ smallest eigenvalues except for the first one which is discarded, minimize ϵ_2. The eigenvector corresponding to the smallest eigenvalue is equal to the mean of vectors $\mathbf{y_i}$ and by discarding it we make $\sum_{i=1}^{n} \mathbf{y_i} = \mathbf{0}$. Figure 4.2 shows a summary of how LLE works. According to this figure, mapping a vector $\mathbf{x_i}$ to its corresponding vector in lower dimensional space $\mathbf{y_i}$ through LLE can be summarized in 3 steps:

1. The nearest neighbours to $\mathbf{x_i}$ are selected.

2. The reconstruction weights w_{ij} are calculated based on how well $\mathbf{x_i}$ can be recreated by its nearest neighbours.

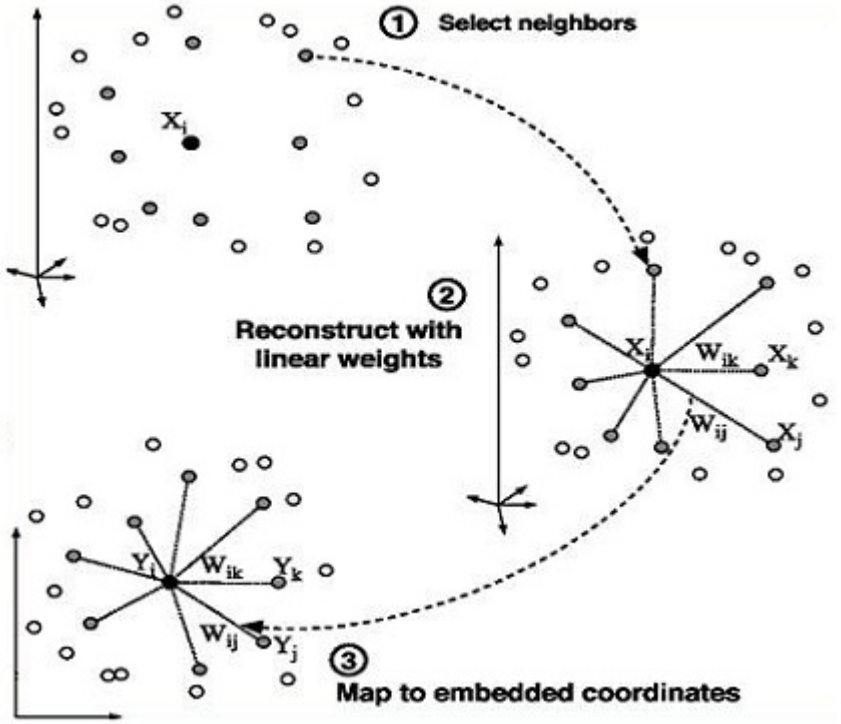

Figure 4.2: A summary of LLE [1]

3. The same reconstruction weights w_{ij} are used to find the mapped vector $\mathbf{y_i}$ based on how well $\mathbf{y_i}$ can be recreated by mappings of $\mathbf{x_i}$'s nearest neighbours weighted to w_ij.

To map a new sample $\mathbf{x}$ which is not included in data set $\mathbf{X}$, its K nearest neighbours $\mathbf{x_{N(1)}}...\mathbf{x_{N(K)}}$ in $\mathbf{X}$ are found to calculate the new reconstruction weights w_j according to Equation (4.3). Then, it is mapped by using the new weights and corresponding vectors in $\mathbf{Y}$ to the nearest neighbours. Therefore, the mapping equation is:

$$\mathbf{y} = \sum_{j=1}^{K} w_j \mathbf{y_{N(j)}}. \tag{4.6}$$

According to above mapping procedure, there are a couple of parameters which affect the results. To have a desired mapping result, these 3 parameters should be set properly:

1. Regularization parameter r: needed for the cases where $K > D$ to

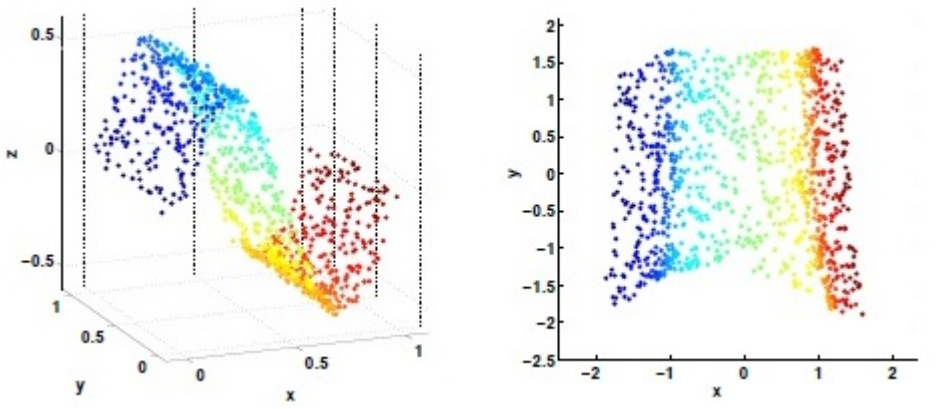

Figure 4.3: A 3D data set mapped to 2D space using LLE [4]

help the eigenvalue computation algorithm converge. For my experiments, $r = 10^{-3}$ results in a good convergence.

2. Number of neighbours K: Low values of K cause a mapping with no global properties. On the other hand, with high values of K, the whole data is considered as neighborhood and as a result, the non-linearity of the mapping process is lost. Additionally, high values of K make the algorithm slower due to high amount of computations.

3. Dimensionality to map to d: Low values of d result in several data samples mapped to the same new point and consequently data is lost and high values for d lead to a mapping with higher amount of noise involved.

Figure 4.3 shows an example of a 3-dimensional data set mapped to 2-dimensional space by LLE.

4.2 Principal Component Analysis

Principal Component Analysis [2] is an unsupervised learning algorithm which reduces the dimensionality of data points by projecting each point of a data set linearly on a number of vectors called principal components. These principal components are ordered in the direction of maximum variance of data and they are orthogonal to each other consecutively. To put it in other words, PCA projects the data to a lower dimensional space

(principal subspace) orthogonally and in a way that leads to maximum variance of the projected data. In addition to dimensionality reduction, PCA has other applications such as data compression, feature extraction and data visualization.

Let $\mathbf{x_n} : n = 1, 2, ..., N$, $\mathbf{x_n} \in R^D$ represent a set of vectors with dimensionality D. Using PCA, we intend to project this set of vectors to a new set of vectors $\mathbf{y_n} : n = 1, 2, ..., N$, $\mathbf{y_n} \in R^d$ with a lower dimensionality $d < D$.

We start the formulation by projecting the data to a one dimensional space or by setting $d = 1$, and then generalize it to all possible values for d. We define the principal component of this one dimensional space by a vector $\mathbf{u_1}$ of dimensionality D. Since we only need the direction of $\mathbf{u_1}$ for data projection, we assume that it is a unit vector and formulate this assumption as: $\mathbf{u_1}^T\mathbf{u_1} = 1$. Using the principal component $\mathbf{u_1}$, the input vectors $\mathbf{x_n}$ are projected to scalar values $y_n = \mathbf{u_1}^T\mathbf{x_n}$.

To proceed, we need the relations for mean and covariance of the input vectors $\mathbf{x_n}$. The following relations give the mean vector $\mathbf{m}$ and covariance matrix $\mathbf{S}$ of input vectors $\mathbf{x_n}$, respectively:

$$\mathbf{m} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x_n}, \tag{4.7}$$

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x_n} - \mathbf{m})(\mathbf{x_n} - \mathbf{m})^T. \tag{4.8}$$

Now, we can proceed by finding a relation for the variance of projected data y_n, and maximizing it in terms of $\mathbf{u}_1$. Considering that the mean of projected data is $\mathbf{u_1}^T\mathbf{m}$, the following relation is the variance of projected data:

$$Var(y_n) = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{u_1}^T\mathbf{x_n} - \mathbf{u_1}^T\mathbf{m})^2 = \mathbf{u_1}^T\mathbf{S}\mathbf{u_1}. \tag{4.9}$$

The goal is now to solve the following maximization problem:

$$\begin{aligned} &\arg\max_{\mathbf{u_1}} \mathbf{u_1}^T\mathbf{S}\mathbf{u_1} \\ &\text{subject to: } \mathbf{u_1}^T\mathbf{u_1} = 1. \end{aligned} \tag{4.10}$$

This is a constrained maximization problem which uses the unity characteristic of $\mathbf{u_1}$ as the constraint to avoid that the magnitude of $\mathbf{u_1}$ tends to

infinity. By using a Lagrange multiplier denoted by λ_1, this constraint is imposed to the maximization problem and the following unconstrained maximization problem is obtained:

$$\arg\max_{\mathbf{u_1}}\{\mathbf{u_1}^T\mathbf{S}\mathbf{u_1} + \lambda_1(1 - \mathbf{u_1}^T\mathbf{u_1})\}. \tag{4.11}$$

To solve this problem we should simply set the derivative in terms of $\mathbf{u_1}$ equal to zero which results in the following equation:

$$\mathbf{S}\mathbf{u_1} = \lambda_1\mathbf{u_1}. \tag{4.12}$$

Equation (4.12) apparently implies that the vector $\mathbf{u_1}$ is an eigenvector of the covariance matrix $\mathbf{S}$. We can left multiply both sides of this equation by $\mathbf{u_1}^T$ as follows:

$$\mathbf{u_1}^T\mathbf{S}\mathbf{u_1} = \lambda_1, \tag{4.13}$$

to make one further conclusion. We observe that the left side of Equation (4.13) is the variance of projected data which is equal to λ_1. This means that we will have the maximum variance of projected data if $\mathbf{u_1}$ is selected as an eigenvector corresponding to the largest eigenvalue of the covariance matrix $\mathbf{S}$.

So far, we have found the first principal component which is the eigenvector related to the largest eigenvalue of $\mathbf{S}$. To find the other principal components we should similarly find the eigenvectors corresponding to the second, third, ... largest eigenvalues of $\mathbf{S}$. The orthogonality requirement of successive principal components is fulfilled since the eigenvectors are orthogonal to each other.

Finally, we generalize the algorithm for any value of d ($d-$dimensional projection). The best $d-$dimensional projection which maximizes the variance of projected data is obtained by finding the d eigenvectors $\mathbf{u_1}, ..., \mathbf{u_d}$ (d principal components) corresponding to the d largest eigenvalues $\lambda_1, ..., \lambda_d$ of the covariance matrix $\mathbf{S}$.

As an example, Figure 4.4 illustrates the realization of a data set with 3 dependent coordinates and its principal components.

4.3 k-Means Clustering Algorithm

$k-$means is an unsupervised clustering algorithm which classifies the data set to a number of clusters ($k-$clusters) such that each data point is associated to the nearest cluster centroid. Mathematically, $k-$means algorithm

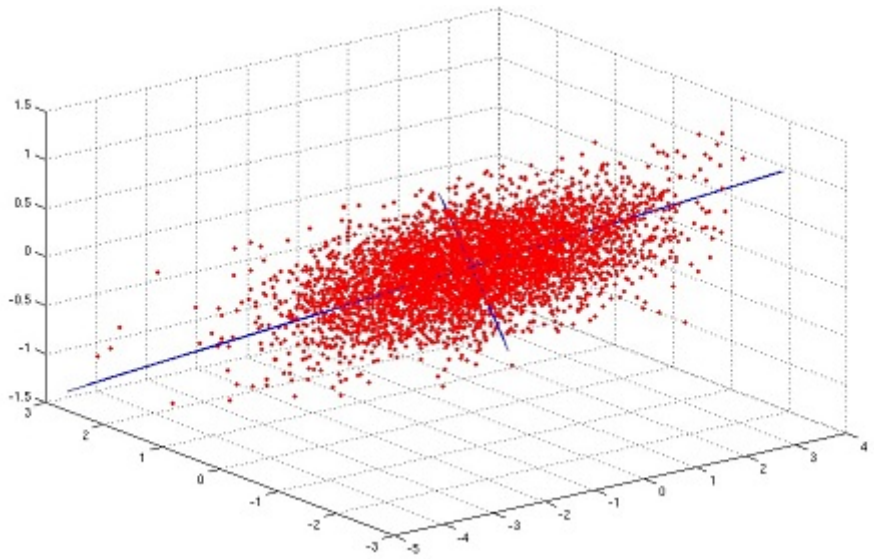

Figure 4.4: Realization of a 3D data set $\mathbf{x}_n$ (red) and Principal components of the data set $\mathbf{x}_n$ (blue)

classifies a data set $\mathbf{X} : \{\mathbf{x_1}, ..., \mathbf{x_n}\}$, where $\mathbf{x_i} \in R^d$, into k disjoint clusters $\mathbf{S} : \{S_1, S_2, ..., S_k\}$ by minimizing the sum of Euclidean distances between the cluster centroid and the data inside a cluster. This can be formulated as:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x_j} \in S_i} ||\mathbf{x_j} - \mathbf{c_i}||^2, \tag{4.14}$$

where $\mathbf{c_i}$ denotes the centroid of cluster S_i. Here, the cluster centroid is the mean of data points inside the cluster. The standard $k-$means algorithm works as follows:

1. The first step is to select a value for the number of clusters k.

2. The second step is to set the initial values for the k cluster centroids $\mathbf{c_1}, \mathbf{c_2}, ..., \mathbf{c_k}$. One way is to assign the data point randomly to the k clusters and calculate the centroids and the other way is to assign the k first data points to each of k clusters and assign the rest of data to the cluster with nearest centroid and recalculate the centroid after each assignment.

3. In the third step calculate the Euclidean distance between each data point and each of the cluster centroids and again assign the data points to the clusters with nearest centroid. Recalculate the cluster

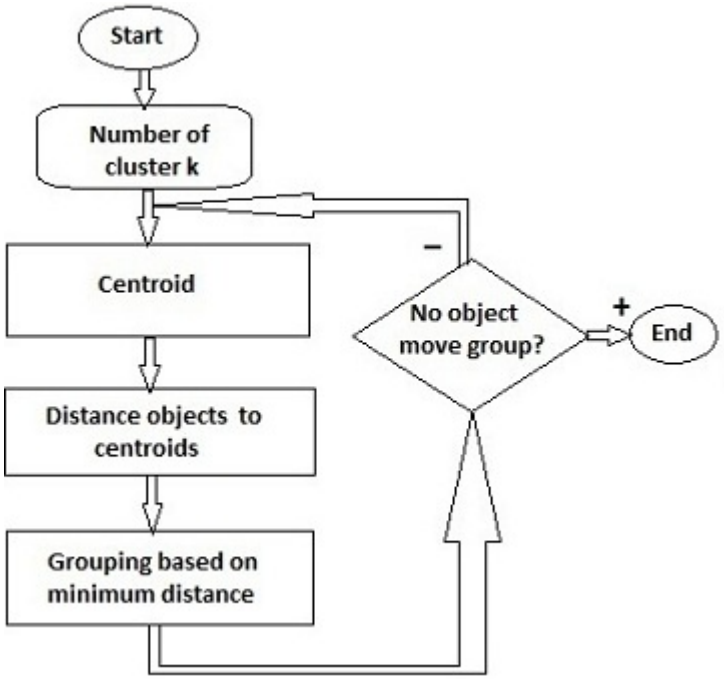

Figure 4.5: Block Diagram of standard $k-$means algorithm [13]

centroid for the clusters which have gained or lost data points. This step can be formulated as follows:

$$S_i^{(t)} = \{\mathbf{x_j} : ||\mathbf{x_j} - \mathbf{c_i}^{(t)}|| \leq ||\mathbf{x_j} - \mathbf{c}_\ell{}^{(t)}||, \forall \ell = 1, ..., k\}, \tag{4.15}$$

$$\mathbf{c_i}^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\mathbf{x_j} \in S_i^{(t)}} \mathbf{x_j}. \tag{4.16}$$

4. Iterate until no change occurs in the location of centroids and the algorithm converges. The algorithm will always converge if after each iteration the sum of Euclidean distances between each cluster centroid and the points in that cluster decreases.

Figure 4.5 shows a block diagram of the algorithm and Figure 4.6 shows an example of how the cluster centroids (denoted by red stars) move during iterations and finally all the data points (pictured by blue squares) are classified to two disjoint clusters.

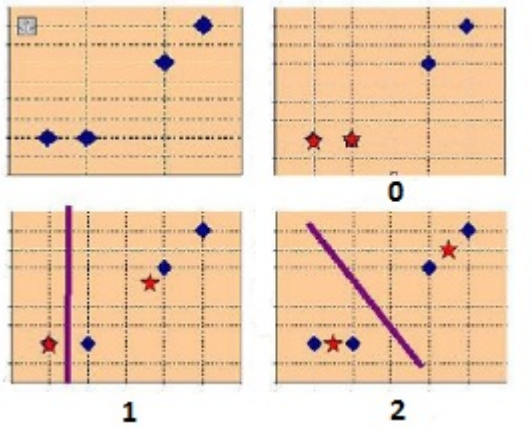

Figure 4.6: An example of how cluster centroids (stars) move after each iteration in the $k-$means algorithm [13]

4.4 Mapping Procedure

The process of mapping the noisy feature vectors to clean ones is performed in several steps which are presented in this section. The clean vectors are the MFCC feature vectors extracted from clean speech signals recorded for the purpose of speech recognition and the noisy vectors are the MFCC feature vectors extracted from the same speech signals and by the same speakers but recorded in a noisy environment like on motorcycles where different sources of noise (wind, construction, other vehicles, etc.) are available.

The following is the noise reduction process:

1. Let $\mathbf{X}$ be the set of clean feature vectors. First, this set is mapped to a lower dimensional set using a dimensionality reduction algorithm (LLE / PCA).

2. Let $\mathbf{x_n}$ be the noisy vector which should also be mapped to low dimensional space. Since it is considered a new previously unseen sample, the mapping can be done using the weights obtained from nearest neighbours of the noisy vector in clean vectors and find the mapping of noisy vector based on that. The process is as follows :

   First of all, the K nearest clean vectors to the noisy vector are found in the original manifold. Using these nearest neighbours, reconstruction weights w_i are calculated so that the following relation is ful-

filled:

$$\mathbf{x_n} \approx \sum_{i=1}^{K} w_i \mathbf{x_i}. \tag{4.17}$$

To get a mapping of the noisy data $\mathbf{z}$, use the same weights as follows:

$$\mathbf{z} \approx \sum_{i=1}^{K} w_i \mathbf{y_i}, \tag{4.18}$$

where $\mathbf{Y}$ is the image of $\mathbf{X}$ mapped to lower dimensional space. Equations (4.17) and (4.18) are developed based on the minimum approximate ($\approx$) error of reconstruction (of signal using its nearest neighbours) in original and lower dimensional spaces.

3. After mapping both clean and noisy vectors to lower dimensional spaces, we find the nearest mapped clean vector $\mathbf{y_i}$ to the mapped noisy vector $\mathbf{z}$.

4. Finally, the nearest neighbour to $\mathbf{z}$ is mapped back again to original space. In other words, $\mathbf{x_i}$ related to $\mathbf{y_i}$ is found and given as the input to the recognizer instead of $\mathbf{z}$.

Lastly, to avoid the fact that the mapped vector $\mathbf{x_i}$ may lie far away from the original vector $\mathbf{x_n}$, a linear interpolation between these two vectors can be used. Therefore, a λ parameter determines what percentage of $\mathbf{x_n}$ and $\mathbf{x_i}$ is considered in output: $\mathbf{o} = (1 - \lambda)\mathbf{x_i} + \lambda \mathbf{x_n}$.

In order to improve the current mapping based noise reduction algorithm, the available clean and noisy data can be classified prior to mapping process. This means that a $k-$means clustering is performed on the clean data set to find several cluster centroids $\mathbf{c_i}, i = 1, ..., k$. Then the nearest cluster $\mathbf{c_i}$ to each noisy vector $\mathbf{x_{n_i}}$ is found by calculating Euclidean distances between that noisy vector and the cluster centroids. Then considering that $\mathbf{x_{n_i}}$ belongs to cluster $\mathbf{c_i}$, the mapping process for $\mathbf{x_{n_i}}$ is done using the clean data from the same cluster $\mathbf{X_{c_i}}$.

A brief description of this algorithm is as follows:

1. Let $\mathbf{X}$ be the clean data set; using a $k-$means clustering algorithm, it is classified to k clusters $\mathbf{X_{c_1}}, ..., \mathbf{X_{c_k}}$, where $\mathbf{c_1}, ..., \mathbf{c_k}$ are cluster centroids. Then each cluster of clean data is mapped to low dimensional space separately: $\mathbf{Y_{c_1}}, ..., \mathbf{Y_{c_k}}$ are clean data sets in low dimensional space.

2. Let $\mathbf{x_n}$ be the noisy vector which can also be classified by calculating the Euclidean distances between $\mathbf{x_n}$ and all cluster centroids to find its nearest cluster $\mathbf{c_i}$. Assuming that the nearest centroid to $\mathbf{x_n}$ is $\mathbf{c_i}$, we can say $\mathbf{x_n}$ belongs to cluster $\mathbf{X_{c_i}}$.

3. To find the best clean vector substitution for $\mathbf{x_n}$, do the same mapping process as before for the data in cluster $\mathbf{c_i}$ or in other words do the mapping for $\mathbf{x_n}$ using $\mathbf{X_{c_i}}$, $\mathbf{Y_{c_i}}$.

Chapter 5

Implementation and Results

In this chapter, different approaches for implementation and the obtained results are presented. For each approach there are a number of parameters which affect the performance and the results are compared so that we finally find the optimum values. Moreover, the results are verified for different speakers and for different levels of artificially added noise.

5.1 Data Selection

The speech data used in this experiment includes clean speech signals from 2 male speakers and the noisy signals obtained by adding wind noise artificially to the same clean signals and with different signal to noise ratio values. The feature vectors are then extracted for clean and distorted speech signals. As an example of how different signal to noise ratio values affect the MFCC feature vectors, you can observe in Figure 5.1 the spectrogram of distorted MFCC vectors with 25 and 0 dB SNR values.

*Corr	Clean	10 dB	25 dB
Speaker 1	82.58	40.91	75.00
Speaker 2	87.88	60.61	86.36

Table 5.1: The percentage of correctly recognized phonemes for our clean and noisy feature vectors

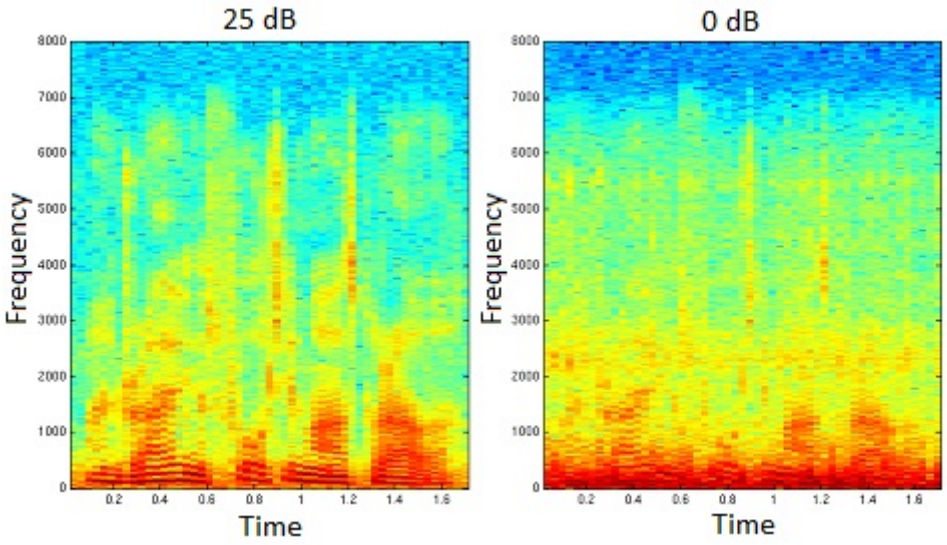

Figure 5.1: The spectrogram of two noisy feature vectors with SNR values of 25 and 0 dB

We have used the Hidden Markov Model Toolkit (HTK) [17] for the continuous (phoneme level) speech recognition and QtOctave for implementation of our noise reduction algorithm. In Table 5.1 you can observe the recognition results of clean vectors, noisy vectors with SNR=10 dB and noisy vectors with SNR=25 dB, for two speakers. The value of "Corr" in this and all other tables represents the percentage of correctly recognized phonemes.

In the following sections the recognition results of the distorted speech signals after processing by our noise reduction methods are presented.

5.2 Implementation by LLE

We implemented our noise reduction procedure described in section 4.4 first by using LLE as the dimensionality reduction algorithm. The noisy feature vectors were originally 13 dimensional vectors. Through the noise reduction process this dimensionality was reduced to the values d lower than 13. The other parameters in our experiment were the number of nearest neighbours K and the interpolation parameter λ described in section 4.4.

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 5$	21.21	35.61	41.67	40.91	42.42
$K = 15$	18.94	37.12	40.15	37.12	40.91
$K = 20$	26.52	43.94	43.18	43.94	43.18

Table 5.2: Speaker 1, SNR = 10 dB, $d = 12$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 5$	18.94	31.82	33.33	43.18	40.91

Table 5.3: Speaker 1, SNR = 10 dB, $d = 11$

The implementation results we obtained using LLE for the dimensionality reduction are given in Tables 5.2 to 5.5 for different values of d, K, and λ. In addition, Figures 5.2 and 5.3 show a graph of how these results change by λ for different values of K and λ. One can observe from Table 5.2 a slight improvement in speech recognition for $K = 20$. However, we found out by checking our results that there is some randomness in the output of LLE that makes our results unstable. In other words, we obtain a new result every time we run the algorithm. This randomness might be caused by the eigen analysis algorithm used in LLE. Moreover, the use of LLE makes the implementation of our method so slow to be run. Hence, we decided to replace LLE by another dimensionality reduction algorithm which does not have the mentioned drawbacks.

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 10$	32.58	52.27	66.67	70.45	74.24
$K = 15$	50.76	67.42	70.45	69.70	75.00

Table 5.4: Speaker 1, SNR = 25 dB, $d = 12$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 10$	15.15	29.55	40.91	56.06	59.85

Table 5.5: Speaker 2, SNR=10 dB, $d = 11$

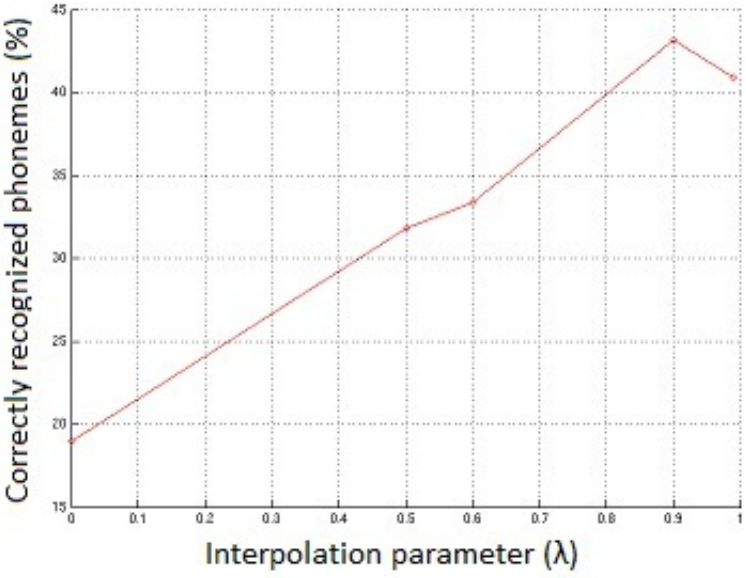

Figure 5.2: LLE, SNR = 10 dB, $d = 11$

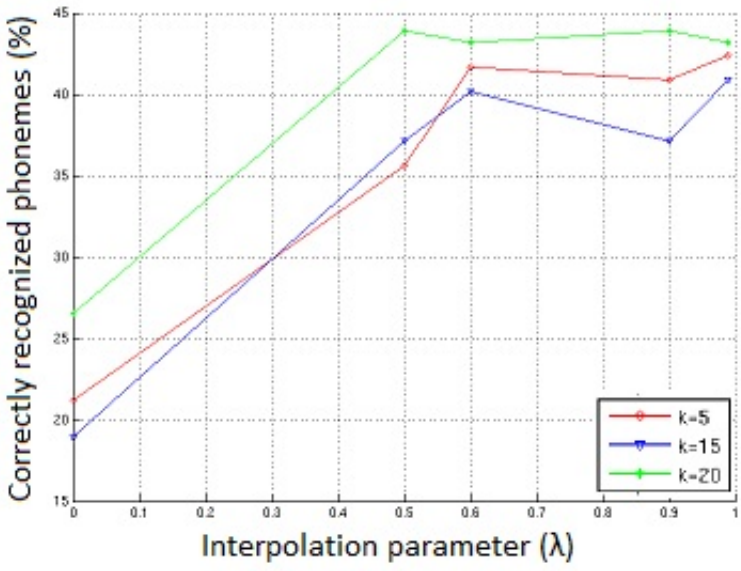

Figure 5.3: LLE, SNR = 10 dB, $d = 12$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 5$	26.52	40.91	42.42	46.97	46.97
$K = 10$	23.48	42.42	43.18	46.21	46.21
$K = 15$	37.88	43.94	42.42	44.70	44.70
$K = 20$	27.27	38.64	40.91	45.45	45.45

Table 5.6: Speaker 1, SNR = 10 dB, $d = 12$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 5$	26.52	40.91	40.15	45.45	45.45
$K = 10$	26.52	40.15	43.94	45.45	45.45
$K = 15$	25.00	40.15	44.70	43.94	43.94
$K = 20$	28.03	41.67	40.91	44.70	44.70

Table 5.7: Speaker 1, SNR = 10 dB, $d = 11$

5.3 Implementation by PCA

The dimensionality reduction algorithm we used instead of LLE was PCA. Using PCA led to the faster implementation of our algorithm. Besides, the randomness caused by LLE did not exist in PCA and consequently using PCA resulted in stable outputs. Tables 5.6 to 5.10 illustrate the results obtained by using PCA instead of LLE. One can observe from Table 5.6, for instance, that using PCA in our algorithm leads to a considerable improvement of 6 percent in the speech recognition. Figures 5.4 and 5.5 illustrate graphically how the results change with different values of λ. The best value of λ and K according to these graphs are $\lambda = 0.9$ and $K = 5, 10$ leading to the highest percentage of recognition. Furthermore, $d = 12$ is the best value of the dimensionality to map to.

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 5$	68.18	74.24	74.24	75.00	75.76
$K = 10$	67.42	72.73	72.73	72.73	75.76
$K = 15$	67.42	71.21	73.48	75.76	75.00
$K = 20$	66.67	70.45	71.97	75.76	75.00

Table 5.8: Speaker 1, SNR = 25 dB, $d = 12$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 10$	23.48	42.42	47.73	60.61	60.61

Table 5.9: Speaker 2, SNR = 10 dB, $d = 11$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 5$	72.73	84.85	84.85	84.85	86.36
$K = 10$	72.73	80.30	80.30	85.61	87.12

Table 5.10: Speaker 2, SNR = 25 dB, $d = 12$

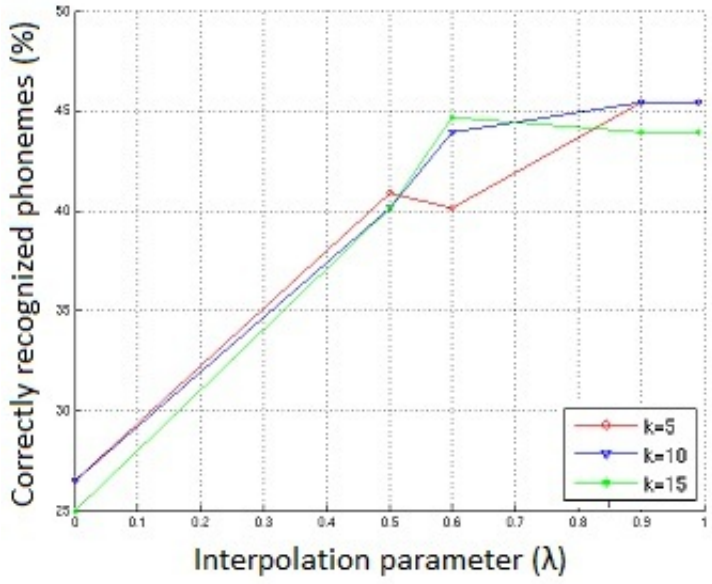

Figure 5.4: PCA, SNR = 10 dB, $d = 11$

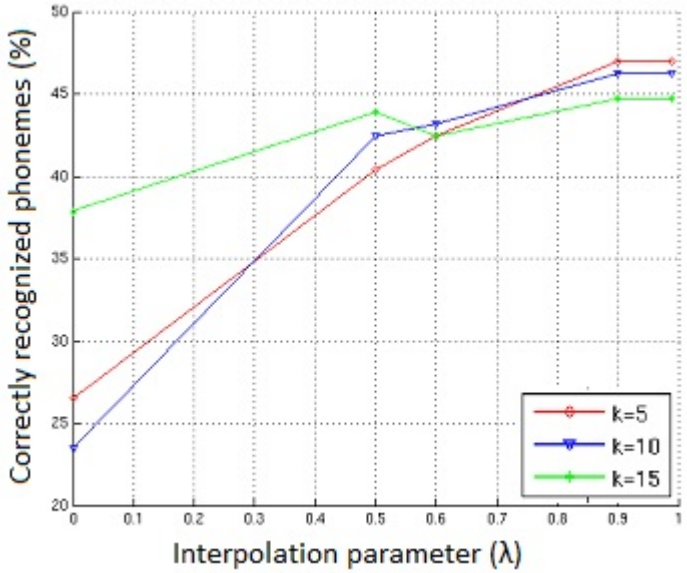

Figure 5.5: PCA, SNR = 10 dB, $d = 12$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 15$	8.33	5.30	6.06	36.36	41.67

Table 5.11: 2 clusters, LLE, Speaker1, SNR = 10 dB, $d = 11$

5.4 Implementation by Clustered Vectors

This section consists of the results obtained by using the clustered feature vectors and doing the noise reduction on each cluster seperately. In this part of our work, we classified both clean and noisy feature vectors to 2, 3, 4, and 5 clusters before we map them to lower dimensional space. Then for each class of vectors we did the same process of noise reduction using both LLE and PCA as dimensionality reduction algorithms. With higher numbers of clusters the noise reduction algorithm was executed faster. However, the obtained results given in Tables 5.11 to 5.30 show no significant improvement in speech recognition. The reason for having undesirable results after clustering the feature vectors might be that we did not have enough training (clean) feature vetors to be used in our algorithm.

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 10$	3.79	12.12	27.27	64.39	75.76

Table 5.12: 2 clusters, LLE, Speaker 1, SNR = 25 dB, $d = 12$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 10$	6.82	8.33	8.33	38.64	38.64

Table 5.13: 2 clusters, PCA, Speaker 1, SNR = 10 dB, $d = 12$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 5$	12.12	11.36	23.48	61.36	75.76
$K = 10$	12.12	12.88	28.03	63.64	71.21

Table 5.14: 2 clusters, PCA, Speaker 1, SNR = 25 dB, $d = 12$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 10$	6.06	6.06	10.61	51.52	51.52

Table 5.15: 2 clusters, PCA, Speaker 2, SNR = 10 dB, $d = 11$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 10$	19.70	16.67	71.21	83.33	87.12

Table 5.16: 2 clusters, PCA, Speaker 2, SNR = 25 dB, $d = 11$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 10$	3.79	2.27	4.55	37.12	37.12

Table 5.17: 3 clusters, LLE, Speaker 1, SNR = 10 dB, $d = 12$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 5$	4.55	13.64	25.00	62.12	75.76

Table 5.18: 3 clusters, LLE, Speaker 1, SNR = 25 dB, $d = 12$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 10$	6.82	3.03	6.06	34.85	40.91

Table 5.19: 3 clusters, PCA, Speaker 1, SNR = 10 dB, $d = 12$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 10$	9.09	8.33	27.27	61.36	75.76

Table 5.20: 3 clusters, PCA, Speaker 1, SNR = 25 dB, $d = 12$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 10$	7.58	6.82	9.85	53.03	53.03

Table 5.21: 3 clusters, PCA, Speaker 2, SNR = 10 dB, $d = 11$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 5$	15.91	17.42	65.15	81.82	86.36

Table 5.22: 3 clusters, PCA, Speaker 2, SNR = 25 dB, $d = 12$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 10$	4.55	4.55	8.33	31.82	40.91

Table 5.23: 4 clusters, LLE, Speaker 1, SNR = 10 dB, $d = 12$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$d = 6$	5.30	6.82	21.97	58.33	75.76
$d = 12$	8.33	9.09	20.45	61.36	75.76

Table 5.24: 4 clusters, LLE, Speaker 1, SNR = 25 dB, $K = 10$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$d = 5$	7.58	4.55	6.82	34.85	$34.85*$
$d = 12$	6.06	5.30	6.06	34.85	$34.85*$

Table 5.25: 4 clusters, PCA, Speaker 1, SNR = 10 dB, $K = 10$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$d = 4$	6.82	10.61	23.48	58.33	76.52
$d = 12$	12.88	9.09	17.42	59.09	75.76

Table 5.26: 4 clusters, PCA, Speaker 1, SNR = 25 dB, $K = 15$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 10$	4.55	4.55	6.06	35.61	38.64

Table 5.27: 5 clusters, LLE, Speaker 1, SNR = 10 dB, $d = 12$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 10$	5.30	8.33	8.33	72.73	71.97

Table 5.28: 5 clusters, LLE, Speaker 1, SNR = 25 dB, $d = 12$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 10$	7.58	3.79	3.79	36.36	36.36

Table 5.29: 5 clusters, PCA, Speaker 1, SNR = 10 dB, $d = 12$

$*Corr$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 0.6$	$\lambda = 0.9$	$\lambda = 0.99$
$K = 5$	7.58	8.33	8.33	71.21	71.21

Table 5.30: 5 clusters, PCA, Speaker 1, SNR = 25 dB, $d = 11$

Chapter 6

Summary and future work

6.1 Conclusions

In this work, we have presented a new approach towards noise reduction for speech recognition named mapping based noise reduction. In this technique, feature vectors which are extracted from noisy speech signals are mapped from a higher dimensional feature space to a lower dimensional subspace in which the clean speech feature vectors are also available. Afterwards, the noisy features are projected to their nearest clean feature vectors in lower dimensional space. The results obtained from our experiments implemented by Octave show that the recognition percentage can be raised up to 6% for some speech signals when we use PCA to find the lower dimensional features. Nonetheless, it should be noted that in some cases, this technique does not lead to a significant improvement in the outcome of the speech recognition system. One of the reasons for such undesirable results might be lack of training (clean) feature vectors. Another reason might be that the feature vectors turn into non smooth vectors after the mapping process. Consequently, this method does not prove yet to yield promising outcomes in all cases.

6.2 Future work

Considering the results of our experiments, the mapping based noise reduction is efficient when enough clean models in lower dimensional space can be found. Accordingly the current work can be improved to some

extent by using more clean feature vectors for training the submanifold model of the clean speech vectors.

Bibliography

[1] LLE algorithm. Locally linear embedding. http://www.cs.nyu.edu/~roweis/lle/.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] John Coleman. Phonetics course. http://www.phon.ox.ac.uk/jcoleman/mst_mphil_phonetics_course_index.html.

[4] Dick de Ridder and Robert P.W. Duin. Locally linear embedding for classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002(January):1–13, 2002.

[5] Andrew Errity and John McKenna. An investigation of manifold learning for speech analysis. *School of Computing Dublin City University*, 2009(January):1–4, 2009.

[6] Sadaoki Furui. *Digital Speech Processing, Synthesis and Recognition (2nd Edition)*. CRC Press, 2000.

[7] Christopher Kermorvant. A comparison of noise reduction techniques for robust speech recognition. 1999:1–16, 1999.

[8] John G. Proakis and Dimitris K Manolakis. *Digital Signal Processing: Principles, Algorithms and Applications (3rd Edition)*. Prentice Hall, 1995.

[9] Rabiner and Juang. Introduction to the physiology of speech and hearing. http://cobweb.ecn.purdue.edu/~ee649/notes/physiology.html.

[10] Giampiero Salvi. Htk tutorial. http://www.speech.kth.se/~matsb/speech_speaker_rec_course_2007/htk_tutorial.pdf.

[11] Sigurdur Sigurdsson, Kaare Brandt Petersen, and Tue Lehn-Schioler. Mel frequency cepstral coefficients: An evaluation of robustness of mp3 encoded music. *Proceedings of the Seventh International Conference on Music Information Retrieval (ISMIR)*, 2006(January):1–4, 2006.

[12] Mark Stamp. A revealing introduction to hidden markov models. (January):1–20, 2004.

[13] Kardi Teknomo. K-mean clustering tutorials. `http://people.revoledu.com/kardi/tutorial/kMean/`.

[14] Yale University. The source-filter model. `http://www.haskins.yale.edu/featured/heads/mmsp/acoustic.html`.

[15] Wikipedia. Hilbert space. `http://en.wikipedia.org/wiki/Hilbert_space`.

[16] Wikipedia. Mel scale. `http://en.wikipedia.org/wiki/Mel_scale`.

[17] Steve Young, Dan Kershaw, Julian Odell, Dave Ollason, Valtcho Valtchev, and Phil Woodland. *The HTK Book (for HTK Version 3.4)*. Cambridge University, 2009.

Typeset August 31, 2010