Chapter 6

Feature Extraction

6.1 INTRODUCTION

The speech signal contains a large number of information which reflects the emotional characteristics, gender classification and the speaker’s identity. Every speech and speaker has special individual characteristics which are embedded in their speech utterances. Feature extraction from the speech signal is the signal processing front end process which converts the human speech into some useful parametric representation. Feature extraction plays a crucial role in the overall performance of a speech recognition as well as speaker recognition system. A good feature extraction technique must capture the important characteristics of the signal also should discard some irrelevant attributes. Feature extraction is the process of keeping useful information from the signal while discarding redundant and unwanted information [31]. Feature parameters play a significant role to distinguish speeches as well as speakers from each other. These parameters are useful for analysis in various speech allied applications such as speech recognition, speaker recognition, speech synthesis and speech coding. Therefore, we tried to extract some prominent characteristics from the speech to satisfy the performance of our proposed system. The extracted features must satisfy some criteria while dealing with the speech signal such as:
The features should represent speech characteristics, including vocal tract characteristics of the speakers and sense of hearing characteristics.

- Easy to compute extracted speech features
- Not be susceptible to mimicry
- Perfect in describing environment variation
- Stability over time

In general, the speech signals are slow varying time signals which are also called as quasi stationary. For this variability in the speech signal, it is better to perform feature extraction in short term interval which could reduce these variability. Therefore, speech signals are examined over a short period of time (generally 10-30 ms), also called frames, where the characteristics of speech signal becomes stationary. We can say that feature extraction is to transform the input speech waveform in to a sequence of acoustic features vectors, each vector representing the information in a small time window of the signal. Feature extraction is a method which transforms high dimensional input signals into lower dimensional vectors [35]. That is, the huge number of voice samples, in our case 16,000 samples per second, is translated into a small number of features that somehow represent the speech/speaker.

The speaker-specific characteristics of speech can be categorized into physical and learned. The physical characteristics are the shapes and sizes of the speech production organs, like vocal folds and vocal tract. The learned characteristics include rhythm, intonation style, accent, choice of vocabulary and so on.

For recognition, good features should have large between-speaker variability and small within-speaker variability, and be robust against noise and distortion. Also the dimension of features should be low, because otherwise the computation cost would be high. Figure 6.1 shows a summary of features from viewpoint of their physical interpretation.

The features for speaker recognition can be divided into:

- Short-term spectral features
- Voice source features
- Spectral-temporal features
- Prosodic features
- High-level features

![Figure 6.1: Physical interpretation view points of features.](image)

The short-term spectral features are the simplest, and most discriminative, so is most commonly used in speech and speaker recognition. State-of-the-art recognition systems often combine these features, attempting to achieve more accurate recognition results. The short-term spectral features convey information of the spectral envelope. The spectral envelope contain information of the speaker’s vocal tract characteristics, like the location and magnitude of the peaks (formants) in the spectrum, hence is commonly used for speaker recognition.

Nowadays, the most common short-term features used in speech and speaker recognition are:
• Short-time Energy (STE)
• Short-time Zero Crossing Rate (ZCR)
• Short-time autocorrelation function
• Linear Predictive Coding (LPC)
• Linear Predictive Cepstrum Coefficients (LPCCs)
• Log Frequency Power Coefficients (LFPCs)
• Perceptual Linear Prediction (PLP)
• Sub-band energy
• Wavelet Transform (WT)
• Formant
• Pitch
• Mel-Frequency Cepstral Coefficients (MFCCs)

The preference of parameters is very important as it involves with the complexity of the neural network design and the accuracy of the speech and speaker recognition. So different recognition systems have different chosen features. In my proposed work, the features short time Zero Crossing rate (ZCR), Short Time Energy (STE), Linear Predictive Coding (LPC) and Mel-frequency Cepstral Coefficient (MFCC).

Before extracting the features from the speech signal, it is necessary first to capture the speech signal followed by the pre-processing of the speech signal. Generally, the pre-processing of the speech signal consists of pre-emphasis, framing and windowing.

The pre-processing of speech signal can be depicted by the block diagram shown in Figure 6.2.
6.2 CAPTURING THE SPEECH SIGNAL

In the processing part, the first action is to capture the signal that we require. The signal from a human is in analog form. To store the speech signal into a computer, this analog signal must be converted into its digitized form. So, the first constituent of speech processing is the measurement of speech signal. When people used to talk into a microphone, firstly the analog signal pressurizes the air, and then the analog electric signals go to the microphone. There are two steps in analog speech digitization process. They are: Sampling and Quantization. The processing sequence can be depicted by the Figure 6.3.

---

**Figure 6.2: Speech signal pre-processing diagram.**

**Figure 6.3: Speech signal digitization procedures.**
6.3 PRE-PROCESSING

6.3.1 Silence Removal

Generally, the speech signals contain many areas of silence or noise parts. The silence parts of the signal are useless for recognition, because they consist of no information. And if we keep the silence part of the signal, it will make the processing signal too large and take more time and memory when extracting information or features from the speech signal. Therefore, only the signal parts that contain the actual speech are useful for recognition. There are many ways to cut out the silence parts from the speech signals. They are:

- **Physical way:** The silence part is removed manually which overwrites the original signal. Using this method, if the signals or the processing database have a small size, it is feasible and also time saving. But in case of a large database, this is not a perfect choice.

- **Processing way:** In this choice we try to get some features of the signals and estimate thresholds for the features, and lastly according to the thresholds remove or keep the frames. The silence removal feature works by taking away everything at the beginning and end of a signal which falls under a certain level or threshold. Mainly, the signal energy and the spectral centroid are used as features in this method.

In this work, as the training of neural network takes a lot of time, a physical or manual method is applied to do the silence removal. The Audacity software is used to record the speech in **16,000 Hz** in **mono format**. The silence removal from the speech can also be done through MATLAB by the steps as follows: **first read the speech, and plot it**. After that check the beginning sample and the end sample of the signal, then cut the silence part keeping only the usable signal. Finally, rewrite the new signal to the old signal with sampling frequency **16,000 Hz**, **16 nbits bits** parameters.
Figure 6.4: Original speech signal ‘• অসমীয়া’ (IPA: /ɔʃmija/) with silence.

The Figure 6.4 shows the original speech of the word ‘• অসমীয়া’ uttered by a male speaker which is plotted in the time domain. The silence of the speech can be removed by the following Matlab code:

```matlab
1. [signal, fs, nbits] = wavread(‘speaker_number_speech_number.wav’);
2. time = (1:length(signal))*1/fs;
3. plot(time, signal, ‘r’);
4. signal = signal(start sample : end sample);
5. wavwrite(signal, 16000, 16, ‘speaker_number_speech_number.wav’);
6. time = (1:length(signal))*1/fs;
7. plot(time, signal, ‘r’)
```

After this procedure, the result is shown in the Figure 6.5 given below:
6.3.2 Normalization

Normalization is a technique to adjust the volume of audio files to a standard level, as different recording levels can cause the volume to differ greatly from word to word [31]. The recording sound samples with different volumes and possibly some DC offset should naturally not influence the detection system. A simple way to get rid of this is to normalize the signal in some way, e.g. scaling, and offsetting the signal so that it can be put down between the levels -1 and 1. Therefore, normalization is needed and can be applied before any other processing. The feature that we will extract later on, e.g. the Mel-Frequency transform, is depended on the power of the signals. This causes that speaking loudly will be seen differently than quietly. By applying normalization in the recording signal, this effect can be reduced.
In my computation, the normalization is done by the Matlab function `mapminmax()`.

The format type of `mapminmax()` function is depicted below.

```matlab
signal=mapminmax(signal, ymin, ymax);
```

Where, $y_{\text{min}} = \text{minimum value for each row of the vector signal (default is -1).}$

and $y_{\text{max}} = \text{maximum value for each row of the vector signal (default is +1).}$

### 6.3.3 Pre-emphasis

Generally speech signal is pre-emphasized before doing any further processing. If we observe the spectrum of a voiced speech segment, we can see that the energies belonging in the voiced samples distributed in the lower frequencies than in the higher frequencies [33].
Therefore, in order to boost the amount of energies in the high frequencies, the motive of pre-emphasis is to compensate the high frequency section which was suppressed during the sound production mechanism of humans. Speech that comes out from the mouth will have to decay of 6dB per octave, a pre-emphasis filter is applied to remove the -6dB octave decay of the spectral energy [37].

The Figure 6.7 depicts the frequency domain representation (amplitude spectrum) of the utterance of the word ‘আসমীয়া’ where it is noticed that the energy in the voiced segments allocates more in the lower frequencies than in the higher frequencies.

Pre-emphasis is performed by a first-order high-pass filter, which can be expressed as a difference Equation (6.1) in time domain as shown below:

\[ \text{EmphasisSignal} = \text{Signal}(n) - a \times \text{Signal}(n-1) \quad \text{--- (6.1)} \]

Or, it can be expressed as a transfer function in z-domain by the Equation (6.2):
The variable $a$ is a filter coefficient whose value lies between 0.9 and 1.0. For my computation, the value of $a$ is set to 0.9375 which is a general setting value.

In Matlab, the pre-emphasis is computed by the function `filter()` depicted as follows.

```
Emphasis_signal=filter([1 -0.9375], 1, signal);
```

After the pre-emphasis, the speech sound is turned into sharper than the original speech signal with a smaller volume. The Figure 6.8 depicts the pre-emphasis of the signal of the speech ‘• অসমীয়া’ in terms of frequency domain.

![Figure 6.8: Speech signal ‘• অসমীয়া’ (IPA: /ɔʃmija/) in frequency domain after pre-emphasis.](image-url)
From the Figure 6.8, it is observed that the volume or amplitude in the speech signal is too much smaller than the original speech signal. That is why we can conclude that after the pre-emphasis the speech sounds sharper too much with a smaller volume.

6.3.4 Windowing

Speech signals generally unstable i.e. the statistical properties of the speech signals across the time are not constant. But in a short interval of time, generally 10 -30 ms, speech signal can be regarded as stationary [31]. This is carried out by multiplying the speech samples with a windowing function to cut out a short segment from the speech signal. The time for which the signal is considered for processing is called a window. The data belonging to the window is called a frame. The speech features are extracted once every \( P \) msec, which is termed as frame rate while the duration of the window is \( Q \) msec. Generally, \( Q \) is bigger than the \( P \). This leads to cause overlap between two consecutive frames. These overlapping segments are used for speech analysis. It is necessary to choose the frame length and frame shift length. This procedure can be described by the Figure 6.9.

![Figure 6.9: Framing.](image)

Page | 146
Normally, the frame overlap is considered to be \( \frac{1}{2} \text{th} \) or \( \frac{3}{4} \text{th} \) of its frame length [35]. In this thesis, the frame length is 256 samples and frame shift is 128 samples where the sampling frequency is 16,000 Hz. In time domain, the frame length is equal to 16 msec and frame shift is 8 msec.

Many windowing functions can be used:

- **Rectangular window**
- **Hanning window**
- **Hamming window**

The rectangular window is very smooth but it causes many problems during the Fourier analysis where it rapidly cuts of the signal at its boundaries because it has a high side lobe that causes frequency spectrum lost, and also the high frequency parts. The **Hanning window** crosses too fast and also the low-pass characteristics are not smooth. In a good window function, it consists of a narrow main lobe and low side lobe levels in their transfer functions, which shrinks the values of the signal toward zero at the window boundaries, avoiding discontinues.

Out of these, the most widely used window is **Hamming window** because of its smoothness nature in low pass and very low side lobe.

### 6.4 FEATURES SELECTION

Selecting features is very important for a recognition system. The complexity of the neural network is depended on the extracted features. So, different recognition systems normally use different features. In this thesis, four features are chosen out of different features which are used frequently in the recognition systems nowadays. The four features that are considered in the feature extraction phase are **Zero Crossing Rate (ZCR)**, **Short-Time Energy (STE)**, **Linear Predictive Coding (LPC)** and **Mel-Frequency Cepstral Coefficients (MFCCs)**.
6.4.1 Zero Crossing Rate (ZCR)

Zero Crossing Rate determines the information about the number of zero crossings present in a given signal. The concept behind zero crossing is to calculate how many times the signal waveform crosses the zero amplitude line by transition from a positive to negative or vice versa in a specific time [33]. In mathematical terms, a ‘Zero Crossing’ is a point where the sign of a function changes (e.g. from positive to negative), represented by a crossing of the axis (zero value) in the graph of the function. Spontaneously if the numbers of zero crossings are more in a given signal, the signal will be changed rapidly which implies that the signal contains the high frequency information. Like the similar way, if the numbers of zero crossings are less, the signal will be changed slowly denoting that the signal contains low frequency information. The zero crossing of a signal can be depicted by the Figure 6.10.

Figure 6.10: Zero crossings of a speech signal.

Zero Crossing Rates (ZCR) is a voice activity detection method to determine whether a speech segment is voiced or unvoiced or silent. The ZCR is also an important parameter for End Point Detection (EPD). The zero crossing is also a technique which can be used to estimate the fundamental frequency of speech. The number of zero crossings per second is equal to twice the frequency of the signal. Therefore, we can say that ZCR gives indirect information about the frequency of the signal.

The zero crossing rates are relatively high in unvoiced sounds compared to the zero crossing rates in the voiced sounds. The zero crossing rate value for the silence region should be zero.
Unfortunately, very few sound samples are recorded in the perfect clean speech i.e. there contains some level of background noise which poses a very high zero crossing rate value.

The zero crossing rate of a stationary signal can be defined by the Equation (6.3).

\[
ZCR = \sum_{n=-\infty}^{\infty} |\text{sgn}(s(n)) - \text{sgn}(s(n-1))| \quad --- (6.3)
\]

Where \(\text{sgn}()\) is a signum function and is defined as by the Equation (6.4).

\[
\text{sgn}(s(n)) = \begin{cases} 
1 & \text{if } s(n) \geq 0 \\
-1 & \text{if } s(n)<0 
\end{cases} \quad --- (6.4)
\]

The Equation (6.3) can be modified for non-stationary signal like speech which is known as short term ZCR by the Equation (6.5) that computes ZCR for the \(i^{th}\) analysis frame of length N of the speech signal.

\[
ZCR(n) = \frac{1}{2N} \sum_{m=0}^{N-1} s(m)w(n-m) \quad --- (6.5)
\]

The factor 2 (by symmetric feature of speech signal) comes to take care from the fact that one cycle of a signal gives two zero crossings values.

The \texttt{zcr()} function of a speech segment can be defined in a programming language as follows.
To determine the ZCR values from a given speech signal the following specifications are considered in my present study.

- Frame length = 256 samples (16 msec)
- Frame overlap = 128 samples (8 msec)
- Sampling Frequency = 16,000 Hz
- Window type = Hamming (256 samples)

Generally, the speech signal changes with time over few msec. Therefore, different Assamese vowel phonemes, consonant phonemes, words and sentences are considered to compute how the zero crossing rates are changed with respect to short period of time. During the ZCR computation of a speech signal, every frame of the speech signal returns one ZCR value.

If we pass the speech of Assamese word "অসমীয়া" to compute the zero crossing rates through the ZCR computation function, we get the feature vector \([\text{zcr}]\) for the speech signal like depicted below:

```matlab
function [zcr]=zcr(segment)
z=0;
for i = 1:length(segment)-1
    if segment(i) * segment(i+1) > 0
        z=z+1;
    else
        z=z+0;
    end
zcr=z/length(segment);
end
```
In the next step, these feature vectors, computed from different Assamese speech utterances from different speakers, are clustered/compressed using \textit{k-means} algorithm followed by the neural network design phase. We can observe from the following figures how ZCR values differ from frame to frame of different speech signal waves like speech voiced, unvoiced and silence. Most practical speech recognition systems depend deeply on vowel recognition to achieve high performance [31]. Figure 6.11 to Figure 6.18 we try to depict visually how zcr values vary from considered vowel phonemes, consonant phonemes, words, and a sentence.
Figure 6.11: ZCR of Assamese vowel phoneme ‘ि’ (/i/) uttered by a speaker.

Figure 6.12: ZCR of Assamese vowel phoneme ‘ु’ (/u/) uttered by a speaker.
Figure 6.13: ZCR of Assamese consonant phoneme ‘খ’ (/kh/) uttered by a speaker.

Figure 6.14: ZCR of Assamese consonant phoneme ‘শ’ (/ɳ/) uttered by a speaker.
Figure 6.15: ZCR of Assamese speech ‘অসমীয়া’ (/ɔɔmija/) uttered by a speaker.

Figure 6.16: ZCR comparison of three utterances of Assamese word ‘বহাগ’ (/bɔhag/) uttered by a speaker (only first 60 frame are considered to display).
Figure 6.17: ZCR comparison of three utterances of Assamese word ‘ ( )’/ɔhag/ uttered by three different speakers (first 60 frame are considered).

Speech signal varies such as from voiced to unvoiced or unvoiced to voice with time over few msec (10 to 30 msec). From the Figure 6.11 to Figure 6.18, we can observe how the zero
crossing rates behaves in different Assamese vowel phonemes, consonant phonemes, words and sentences with frame time 16 msec. It is observed that zero crossing rates are quite high of unvoiced sounds (like /k/, /kh/, /r/, /ɳ/ etc.) compared to voiced sounds (like /a/, /e/, /i/, /o/ etc.).

It is thus seen from the above ZCR characteristics that zero crossing rate is an important parameter for voiced/unvoiced classification in case of both the male and female speakers. The zero crossing count is also seen as an indicator of the frequency of the signal at which the energy is concentrated in the speech signal. That is why, this feature vector is taken for our recognition system.

6.4.2 Short Time Energy (STE)

Energy associated with the speech signal is time varying in nature. The loudness of a speech signal is the most prominent characteristics according to human aural perception. There are several interchangeable terms like volume, energy, intensity etc. which are commonly used to describe the loudness of speech signals. That is why short time energy is computed from a speech signal for describing the loudness. Short time energy is an acoustic feature that is correlated to the samples’ amplitudes within a frame i.e. short period of time. Hence, the importance for any automatic processing of speech is to determine how the energy is varying with time i.e. to be more clearly, energy associated with short term region of speech [75, 76].

The relation to determine the short time energy in a speech signal can be derived from the total energy computation relation defined in signal processing. The total energy of a signal is defined by the Equation (6.6).

\[
E_T = \sum_{n=-\infty}^{\infty} s^2(n)
\]  

--- (6.6)

In case of short time energy computation, the speech signal is considered to be length of 10-30 msec. Let us, considered the frame length of the speech signal is \(N\) where the samples of the speech are given by “\(n=0 \text{ to } n=N-1\)”. Then outside the frame the energy computation amplitudes of the speech samples will be zero. Now we can replace the Equation (6.6) by the Equation (6.7).
\[
E_T = \sum_{n=-\infty}^{-1} s^2(n) + \sum_{n=0}^{N-1} s^2(n) + \sum_{n=N}^{\infty} s^2(n) \\
= \sum_{n=0}^{N-1} s^2(n) 
\]

--- (6.7)

The Equation (6.7) gives the total energy present in a frame of speech signal i.e. from \( n=0 \) to \( N-1 \).

To represent a speech frame more specifically we can depict it by the Equation (6.8) as follows.

\[ s_w(n) = s(m)w(n-m) \]

--- (6.8)

Where \( w(n) \) represent the windowing function and \( n \) is the shift/rate in number of samples at which we want to determine the short time energy.

Now, we can to find the energy of a speech signal by the Equation (6.9) as follows.

\[ E(n) = \sum_{m=-\infty}^{\infty} (s(m)w(n-m))^2 \]

--- (6.9)

To determine the short time energy from a given speech signal the following specifications are considered.

- Frame length= 256 samples (16 msec)
- Frame overlap= 128 samples (8 msec)
- Sampling Frequency= 16,000 Hz
- Window type= Hamming (256 samples)

To compute the short time energy a function \( \text{STE}() \) is defined. If we consider the Assamese speech "অসমীয়া" from our database to calculate the short time energy through the defined function \( \text{STE}() \), we get the following output returning total number of frames in the speech
signal (total 95 frames) and also the short time energy frame by frame of the speech to built the feature vector [STE].

In the next step, these feature vectors, extracted from different Assamese speech signals from different speakers, are clustered followed by the neural network design phase.

We can observe from the following Figure 6.19 to Figure 6.25 how the STE values differ frame by frame of different speech signals like: speech voiced, unvoiced and silence parts.
Figure 6.19: STE of Assamese vowel phoneme ‘ই’ (/i/) uttered by a speaker.

Figure 6.20: STE of Assamese vowel phoneme ‘উ’ (/u/) uttered by a speaker.
Figure 6.21: STE of Assamese consonant phoneme ‘ ’ (/kh/) uttered by a speaker.

Figure 6.22: STE of Assamese word ‘ ’ (/ram/) uttered by a speaker.
Figure 6.23: STE comparison of vowel phoneme ‘এ’ (/e/) uttered by both male and female speakers.

Figure 6.24: STE comparison of word ‘সময়া’ (/ɔjɔmija/) uttered by both male and female speakers.
From the Figures, we can conclude that the energy associated with **voiced region** is high compared to **unvoiced region**. The silence region shows negligible energy content. Thus, short time energy can be used for voiced, unvoiced and silence classification of speech. Like zero crossing rate, the short time energy is also used for the end point detection of speech signal in the present study. For these reasons, short time energy is considered as one of the features in the proposed recognized system.

One of the most basic but difficult aspects of speech processing is to detect when a speech utterance starts and ends which is termed as **end-point detection (EPD)** mechanism. The goal of end-point detection is to identify the important part of a speech segment. EPD plays an essential part because it is the first step in audio signal processing and recognition. In case of unvoiced sounds taking place at the beginning or end of the utterance, it is difficult to detect
accurately the speech signal from the background noise signal [60]. In this work, end-point detection is carried out to separate silence and speech signal. Both zero crossing rate (ZCR) and short time energy (STE) can be used for end-point detection (EPD) of speech signal like depicted in the Figure 6.26.

Figure 6.26: End-point detection computation of Assamese sentence speech “আ এ ম প থ আ ব হ আ ল ব আ ছ” uttered by a speaker.

This example uses the toolbox SAP for end-point detection. In the above Figure 6.26 the green and red lines indicate the beginning and end of utterance respectively.
6.4.3 Linear Predictive Coding (LPC)

Speech signal is formed by the convolution of excitation source and time varying vocal tract components. These excitations and vocal tract components are to be separated from the existing speech signal to study these components independently. LPC is a method of separating out the effects of source and filter from a speech signal [88]. Methods depend on homomorphic analysis like cepstral analyses are evaluated to de-convolute the given speech into excitation and vocal tract system components. The cepstral analysis gives the deconvolution of speech into source and system components by traversing through frequency domain which makes the deconvolution task computational intensive process. LPC is a tool used mostly in audio signal processing and speech processing for representing the spectral envelop of a digital signal of speech in compressed form, using the information of a linear predictive model.

![Spectral envelop and its formants.](image)

We are computed to obtain an estimate of the spectral envelop as depicted in **Figure 6.27** of the frequency magnitude spectrum in order to find the formant locations and thus classify speakers based on their unique speech patterns. **Linear predictive coding (LPC)** offers a powerful, yet simple method to provide exactly this type of information. The way in which LPC is applied to analysis the speech signal directs a reasonable source-vocal tract separation. As a result, a cost-conscious representation of vocal tract characteristics makes possible. The spectral envelop of a
digital signal of speech in compressed form is represented in terms of LPC model [90]. It is a speech analysis technique for encoding quality speech sound at a low bit rate that gives a way for estimation of speech parameters like cepstral coefficients, formant frequencies and pitch etc. Basically, the LPC algorithm produces a vector of coefficients that represent a smooth spectral envelop of the DFT magnitude of a temporal input signal. The principle of LPC is that speech consists of an impulse that is passed through an all-pole filter, which corresponds to the vocal tract transfer function, and it analyses the given segment of speech to find the coefficients of the filter that represents the vocal tract.

6.4.3.1 LPC analysis

In LP analysis of speech, an all pole model is assumed for the system producing speech signal \( s(n) \). The basic concept behind the LPC model is that a given speech sample at time \( n \), \( s(n) \), can be approximated as a linear combination of the past \( p \) speech samples. The predicted sample can be represented as by the Equation (6.10)

\[
s(n) = \sum_{i=1}^{p} a_i s(n-i) + Gu(n)
\]

--- (6.10)

Where \( a_i \) \((i=1, 2, 3, \ldots, p)\) are co-efficient assumed to be constant over the speech analysis frame. The \( u(n) \) is the normalized excitation and \( G \) is the gain of excitation. If \( \hat{s}(n) \) is the estimate value of \( s(n) \), calculated from the linear combination of past \( p \)-samples, then we can get the Equation (6.11).

\[
\hat{s}(n) = \sum_{k=1}^{p} a_k s(n-k)
\]

--- (6.11)

Now, the predictor error can be defined as by the Equation (6.12) given below.

\[
e(n) = s(n) - \hat{s}(n)
\]
\[ s(n) - \sum_{k=1}^{p} a_k s(n-k) \quad \text{--- (6.12)} \]

This error is nothing but the LP residual of the given speech signal.

For a speech frame of size \( m \) samples, the mean square of prediction error over the whole frame is given by

\[
E = \sum_{m} e^2(m) = \sum_{m} \left[ s(m) - \sum_{k=1}^{p} a_k s(m-k) \right]^2 \quad \text{--- (6.13)}
\]

Optimal predictor coefficients will minimize this mean square error. The minimum MSE criterion of \( E \) is

\[
\frac{\partial E}{\partial a_k} = 0, \quad k = 1, 2, \ldots, p \quad \text{--- (6.14)}
\]

Differentiating the Equation (6.13), we get

\[ Ra = r \quad \text{--- (6.15)} \]

Where, \( a = [a_1, a_2, \ldots, a_p]^T \), \( r = [r(1), r(2), \ldots, r(p)]^T \), and \( R \) is a Toeplitz symmetric autocorrelation matrix given by,
Equation (6.15) can be solved for prediction coefficients using Durbin’s algorithm as follows:

\[ E^{(0)} = r(0) \] \hspace{1cm} \text{(--- (6.17))}

\[ k_i = \frac{r(i) - \sum_{j=1}^{i-1} \alpha_j^{i-1} \cdot r(|i - j|)}{E^{(i-1)}}, \quad 1 \leq i \leq p \] \hspace{1cm} \text{(--- (6.18))}

\[ \alpha_i^{(i)} = k_i \] \hspace{1cm} \text{(--- (6.19))}

\[ \alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)} \] \hspace{1cm} \text{(--- (6.20))}

\[ E^{(i)} = (1 - k_i^2) E^{(i-1)} \] \hspace{1cm} \text{(--- (6.21))}

The above set of Equations can be solved recursively for \( i = 1, 2, \ldots, p \). The final solution is given by the Equation (6.22)

\[ a_m = \alpha_m^{(p)}, \quad 1 \leq m \leq p \] \hspace{1cm} \text{(--- (6.22))}
Where, $a_m$’s are linear predictive co-efficient (LPCs).

### 6.4.3.2 LPC Processor for Speech and Speaker Recognition

The LPC front-end processor has been widely used in speech and speaker recognition system. The block diagram of LPC processor can be depicted by the Figure 6.28 given below.

![Figure 6.28: The block diagram of LPC processor.](image)

The basic steps in the processing include the following:

- **Pre-emphasis**

The speech waveform which is digitized has a high dynamic range and it suffers from additive noise. So pre-emphasis is used to spectrally flatten the signal so as to make it less susceptible to finite precision effects in the processing of speech. The most widely used pre-emphasis is the fixed first-order system. The calculation of pre-emphasis is shown as follows.

$$H(z) = 1 - az^{-1} \quad 0.9 \leq a \leq 1.0 \quad \text{--- (6.23)}$$

The most common value for $a$ is 0.95 (Deller et al; 1993). For fixed point implementation value of $a$ is 0.9375). A Pre-Emphasis can be expressed as

$$\hat{s}(n) = s(n) - 0.95s(n-1) \quad \text{--- (6.24)}$$
• **Frame Blocking**

The speech signal is said to be stationary when it is examined over a short period of time. In order to analyze the speech signal, it has to be blocked into frames of $N$ samples, with adjacent frames being separated by $M$ samples. If $M \leq N$, then LPC estimates from frame to frame will be quite smooth. On the other hand, if $M > N$ there will be no overlap between adjacent frames.

If we denote the $l^{th}$ frame of speech by $x_l(n)$, and there are $L$ frames within the entire speech signal, then

$$x_l(n) = \tilde{s}(Ml + n), \quad n = 0, 1, ..., N - 1, \quad l = 0, 1, ..., L - 1 \quad --- (6.25)$$

• **Windowing**

Each frame is windowed in order to minimize the signal discontinuities or the signal is lessened to zero at the starting and ending of each frame. If window is defined as $w(n)$, then the windowed signal is

$$\tilde{x}(n) = x(n)w(n), 0 \leq n \leq N - 1 \quad --- (6.26)$$

A typical window used is the Hamming window, which has the form

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2n}{N - 1}\right), 0 \leq n \leq N - 1 \quad --- (6.27)$$

• **Autocorrelation Analysis**

Each frame of windowed signal is next auto-correlated to give

$$R(n) = \sum_{m=0}^{N-1-n} \tilde{x}(n)\tilde{x}(n + m), m = 0, 1, 2, \ldots, p \quad --- (6.28)$$
Where, the highest autocorrelation value, \( P \) is the order of the LPC analysis. The selection of \( p \) depends primarily on the sampling rate.

- **LPC Analysis**

The next processing step is the LPC analysis which converts each frame of autocorrelation coefficients \( R \) into the LPC parameters. The LPC parameters can be the LPC coefficients. This method of converting autocorrelation coefficients to LPC coefficients is known as Durbin’s method. **Levinson-Durbin recursive** algorithm is used for LPC analysis which is already mentioned in the LPC analysis part.

In the present study, **12 order lpc** coefficients are computed. The specifications for the parameters are as listed below:

- **Frame length**=256 samples
- **Frame overlap**= 128 samples
- **Sampling frequency**= 16,000 Hz.
- **Window type**=Hamming (256 samples)

From **12 order lpc** computation of a particular speech frame, we get **13 lpc coefficients** values. In our analysis, the first LPC coefficient is removed because it is always **1(one)**. If there are \( N \) numbers of frames in the speech sample, we get \( N \times 12 \) numbers of coefficients to form the feature vector \([\text{lpc}]\) of the speech signal.

For example, if we take the speech of Assamese word ‘• răng’ (/ram/) of a speaker, then we get the output result, depicted below, from the Matlab built-in function **lpc()**. Here, only first ten frames of the speech sample are considered.
The 12 order LPC coefficients ignoring the first coefficient value for three utterances of the word “অসমীয়া” uttered by a female speaker are presented in Figure 6.29.

Figure 6.29: 12th order LPC coefficients for 3 utterances of word “অসমীয়া (/ɔʃmija/).
Figure 6.30: 12th order LPC coefficients of speech “ම” (/ma/) uttered by three different speakers.

Again, the 12 order LPC coefficients are plotted in the Figure 6.30 ignoring the first coefficient value for Assamese word “ම” (/ma/) uttered by three different speakers. We can observe from the Figures (6.29 and 6.30) that the coefficients peaks tend to occur at the same spots. When applied properly, the LPC coefficients highlight the locations of the formants in the frequency spectrum which are greatly influenced by the glottal shape and vocal cords of the speakers indicating the speaker’s uniqueness and speeches.

In the next step, these lpc feature vectors [lpc] of different speeches from our database are compressed/clustered to make the different sizes of features into same size before passing these features through the neural network for further processing in our recognition system.

6.4.4 Mel Frequency Cepstral Coefficients (MFCCs)

MFCC is one of the most popular feature extraction techniques used in automatic speech or speaker recognition system using the Mel scale which is based on the human ear scale. It is based on the non linear human perception of the frequency of sounds. These coefficients
represent audio based on perception. They are derived from the Mel frequency cepstrum. The spectral information can after that be converted to MFCC by passing the signals through band pass filters where higher frequencies are artificially boosted, and then applying an inverse Fast Fourier Transform (FFT) on it [99]. It combines the advantages of the cepstrum analysis with a perceptual frequency scale based on critical bands.

As a result, the higher frequencies are being more prominent. Since the Mel frequency cepstrum can represent a listener’s response system clearly, therefore MFCC is always considered to be the best available approximation of human ear. The block diagram of MFCC computation can be depicted by the Figure 6.31.

![Block diagram of MFCC.](image)

Figure 6.31: Block diagram of MFCC.

It can be shown in more clearly as depicted by the following Figure 6.32.

![Flow diagram of MFCC computation.](image)

Figure 6.32: Flow diagram of MFCC computation.
To obtain the MFCCs of a speech signal, the signal is first subjected to pre-emphasis filtering procedure. After that, the speech is processed frame-by-frame basis which is called framing. In frame blocking, the continuous speech signal is divided into frames of N samples with adjacent frames being separated by M (M<N). The first frame consists of the first N samples. The second frame begins M samples after the first frame, and overlaps it by N-M samples and so on. The next step in the processing is to window each individual frame so as to minimize the signal discontinuities at the beginning and end of each frame. Hamming window is used for this computation. Windowing means multiplying the window function $w(n)$ with the framed speech signal $s(n)$ to obtain the windowed speech signal $s_w(n)$ like given below.

$$s_w(n) = s(n).w(n) \quad \quad (6.29)$$

The discrete Fourier transform (DFT) of the windowed speech signal is then computed by the following Equation (6.30).

$$\hat{S}_w(k) = \sum_{n=0}^{N-1} s_w(n)e^{-j2\pi kn/N} \quad \quad (6.30)$$

The Mel-filter bank is a triangular band pass filter which is equally spaced around the Mel-Scale. A Mel is a unit of perceived pitch or frequency of a tone. The mapping between real frequency (Hz) and Mel-frequency is given by the following Equation (6.31).

$$f_{mel} = 2595\log(1 + \frac{f}{700}) \quad \quad (6.31)$$

The power spectrum from the DFT step is then binned by correlating it with each triangular filter in order to reflect the frequency resolution of the human ear. Binning means multiplying the power spectrum coefficients with the triangular filter gain or coefficients and summing the resultant values to obtain the Mel-spectral coefficients as in Equation (6.32).

$$G(k) = \sum_{n=0}^{N} \eta_{kn}.[\hat{S}_w(k)]^2 \quad \quad (6.32)$$
Where $\eta_{kn}$ is the triangular filter coefficients, $k=0, 1, 2, \ldots, k-1$, $n=0, 1, 2, \ldots, N/2$ and $G(k)$ is the Mel-spectral coefficients.

After that, the log of the Mel-spectral coefficients $G(k)$, is taken. This step is to level unwanted ripples in the spectrum and done the following Equation (6.33).

$$m_k = \log G(k) \quad \text{--- (6.33)}$$

Finally, DCT is applied to the log Mel-cepstrum $m_k$ as in Equation (6.34) to obtain the Mel-frequency Cepstral Coefficients (MFCC) $c_i$ of the $i^{th}$ frame.

$$c_i = \sqrt{\frac{2}{N}} \sum_{k=1}^{N} m_k \cos \left( \frac{\pi i}{N} (k - 0.5) \right) \quad \text{--- (6.34)}$$

The relation between Hertz scale and Mel scale can be depicted by the Figure 6.33.

![Relation between Hertz scale and Mel scale](image)

Figure 6.33: Hz-Mel Scale.

In this thesis, 24 triangular filters are used to map the powers of the spectrum onto the Mel scale. A picture showing the 24 triangular filters depicted in Figure 6.34.
The crosses (‘x’) indicate the centre frequency of each triangular filter.

The MFCC is done by the tool voicebox where there is a function called melcepst() to calculate the Mel cepstrum. The preprocessing steps like framing, windowing are present in the melcepst() function itself. The MFCC is computed of a speech signal using built in function melcepst() as follows.

\[
\text{MFCC=melcepst ( EmphasisSignal, 16000, ‘M’, 12, 24, 256, 128 ) ;}
\]

Where, 16,000 is the sampling frequency of the speech signal, ‘M’ means window type Hamming, 12 is the number of cepstral coefficients, 24 is the number of triangular filter banks, 256 is the frame length and 128 is the frame overlap which are already discussed.
Figure 6.35: 12 MFCC computation of 10\textsuperscript{th} frame of speech “ক” (/kitap/) uttered by a speaker.

Figure 6.36: 12 MFCC computation of 15\textsuperscript{th} frame of speech “ম” (/ma/) uttered by three speakers.
The output from the MFCC computation function for the first ten frames of the speech signal is depicted below.

The output MFCC is a $N \times 12$ matrix of 12 MFCCs per each $N$ frame. The numbers of frames $N$ will naturally differ for each speech signal. These MFCC are passed to the neural network. But a neural network cannot have a variable number of $N$ of inputs, so we need to compress the feature vector using clustering algorithm.

### 6.5 FEATURE VECTOR COMPRESSION

In the above sections each of the features, such as **Zero Crossing Rate**, **Short Time Energy**, **LPC** and **MFCC** are computed for each frame of the speech signal. But the problem is the strong randomness nature of the speech signal which leads different signals may have different sizes of feature vectors. **For example**, the utterance of the word ‘*া/ma/* will have different durations, and thus different number of frames every time. This will create the non-static feature structure of the extracted features vectors. Moreover, the features vectors will be the inputs to the artificial neural network, but the basic artificial neural networks cannot handle these time varying characteristics and the non-static information. To overcome from these problems, it is necessary
to merge some of the elements in the feature vector. Therefore, a method of compressing the different sizes of features into the same size is needed.

The **K-means** clustering algorithm is used as a compressing method in this thesis. Clustering is an essential task in data mining process which is used for the purpose to make groups or clusters of the given data set based on the similarity between them. A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters. Clustering has been a widely studied problem in a variety of application domains including neural networks. Cluster analysis divides data into meaningful or useful groups or clusters.

### 6.5.1 K-means

The k-means is one of the simplest and most popular clustering techniques which is used here to cluster the feature vector. This is an unsupervised learning algorithm. This algorithm clusters the feature vector depending on attributes into K partitions. It uses the k means of data generated from Gaussian distributions to cluster the vectors.

<table>
<thead>
<tr>
<th>K-means Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong></td>
</tr>
<tr>
<td>K: the number of clusters.</td>
</tr>
<tr>
<td>D: a data set containing n objects.</td>
</tr>
<tr>
<td><strong>Output:</strong></td>
</tr>
<tr>
<td>A set of K clusters.</td>
</tr>
<tr>
<td><strong>Method:</strong></td>
</tr>
<tr>
<td>1. Arbitrarily choose K objects from D as the initial cluster centers.</td>
</tr>
<tr>
<td>2. <strong>Repeat</strong></td>
</tr>
<tr>
<td>3. (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster.</td>
</tr>
<tr>
<td>4. Update the cluster means, i.e., calculate the mean value of the objects for each cluster.</td>
</tr>
<tr>
<td>5. <strong>Until</strong> no change</td>
</tr>
</tbody>
</table>

The algorithm can be described by the following steps:
• Define \( k \)-centroids, one for each cluster. These centroids should be positioned in a cunning way because different locations cause different result. The better selection is to place them as far away as possible from each other.

• Take every point belonging to a given data set and associate it to the nearest centroid. When no point is waiting, the first step is completed and an early group age is done. At this point we need to re-calculate \( k \) new centroids as barycenters of the clusters resulting from the previous step. After generating these \( k \) new centroids, a new binding has to be done between the same data set points and the nearest new centroids. A loop has been generated resulting the \( k \) centroids change their locations step by step until no more changes are done.

• Repeat the previous step until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be computed.

In this step the algorithm aims at minimizing an objective function, in this case considering a squared error function. The objective function is calculated as follows:

\[
J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^j - c_j \right\|^2
\]

--- (6.35)

Where, \( \left\| x_i^j - c_j \right\|^2 \) is a chosen distance measure between a data point and the cluster center. The result is an indicator of the distance of the \( n \) data points from their respective cluster centers.
Figure 6.37: Clusters using K-means algorithm.

One example of **K-means cluster** algorithm is depicted in the Figure 6.37. In this case, we splitted the data in 2 clusters, the blue points have been assigned to the first and the red ones to the second. The crosses are the centers of the clusters.

K-means algorithm has been optimized for speed, and is suitable for very large data sets. The K-means algorithm implemented as a built in function in Voicebox which is used for computation with **slight modifications** in the algorithm.

The K-means is called as follows:

```
features = kmeans(feature_vector, num_of_clusters);
```

Where, feature_vector is any one from the features ZCR, STE, LPC and MFCC.

The **num_of_clusters** is a variable what is tuned. If we take the feature vector MFCC, then the result from the `kmeans()` function is a “ **num_of_cluster x 12** ” instead of the previous “**N x 12**” matrix, where `num_of_clusters` is a known, chosen value.
The output matrix is again reshaped by the following programming code into a “1 by num_of_cluster x 12 vector”, which will be the input to the neural network.

```plaintext
features = features(:);
```

We can describe the whole clustering concept in this thesis by the Figure 6.38 depicted as follows.

![Figure 6.38: Clustering N-phoneme word. (KMC: K-means clustering algorithm).](image)

The block diagram of working of the recognition system can be depicted by the Figure 6.39. The architectures are same for both speech and speaker recognitions systems.
Each compressed feature vector ZCR, STE, LPC and MFCC which are computed frame by frame in the speech signal are made altogether to represent the resultant feature vector frame by frame of the speech signal. Now these resultant feature vectors are passed to the neural network.