# DIGIT SPEECH RECOGNITION

*Project Submitted for the Course*

**EE-322M: Signal Processing (Spring, 2015)**

Instructors:
Prof. Prithwijith Guha
Prof. Rohit Sinha

*by*

VINEETH S. BHASKARA [Roll# 120121007]

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

**April, 2015**

## Introduction

This project Recognizes the Digit Speech using the fairly recent state-of-the-art technologies of very efficient Feature Extraction and Pattern Sequence Matching. Although our project mentions 'Digit Speech Recognition', but the same framework may be applied even for 'Character/Word Speech Recognition'. These are very important for fast and accurate Automatic Speech Recognition (ASR) technology.

The Voice Signal is, practically, a signal of infinite dimension for computing purposes. A direct analysis and synthesizing the complex voice signal is very cumbersome due to too much information contained in the signal. Moreover, it is very hard for the computer to distinguish between 'Speech' and 'Noise' parts of the Signal, and therefore any Learning Algorithm may learn Noise instead of actual Speech part and hence may result in a very bad Recognizing system.

Therefore, it becomes extremely important to precisely represent the voice signal features so that we learn not the Noise but only the Speech. For this purpose, the Mel-Frequency Cepstral Coefficient Features. They were introduced by **Davis and Mermelstein [1]** in the 1980's, and have been state-of-the-art ever since. For Feature Matching, we use another revolutionary technique to find distance between Time Sequences of Multi-dimensional Vectors, called Dynamic Time Warping (DTW) introduced by **Sakoe and Chiba [2] in IEEE Transactions** having around 3000 citations now.

## Feature Extraction: MFCCs

The first step in any automatic speech recognition system is to extract features i.e. identify the components of the audio signal that are good for identifying the linguistic content and discarding all the other stuff which carries information like background noise, emotion etc.

The main point to understand about speech is that the sounds generated by a human are filtered by the shape of the vocal tract including tongue, teeth etc. This shape determines what sound comes out. If we can determine the shape accurately, this should give us an accurate representation of the phoneme being produced. The shape of the vocal tract manifests itself in the envelope of the short time power spectrum, and the job of MFCCs is to accurately represent this envelope.

## Motivation behind the MFCCs steps:

An audio signal is constantly changing, so to simplify things we assume that on short time scales the audio signal doesn't change much (when we say it doesn't change, we mean statistically i.e. statistically stationary, obviously the samples are constantly changing on even short time scales). This is why we frame the signal into 20-40ms frames. If the frame is much shorter we don't have enough samples to get a reliable spectral estimate, if it is longer the signal changes too much throughout the frame.

The next step is to calculate the power spectrum of each frame. This is motivated by the human cochlea (an organ in the ear) which vibrates at different spots depending on the frequency of the incoming sounds. Depending on the location in the cochlea that vibrates (which wobbles small hairs), different nerves fire informing the brain that certain frequencies are present. Our periodogram estimate performs a similar job for us, identifying which frequencies are present in the frame.

The periodogram spectral estimate still contains a lot of information not required for Automatic Speech Recognition (ASR). In particular the cochlea can not discern the difference between two closely spaced frequencies. This effect becomes more pronounced as the frequencies increase. For this reason we take clumps of periodogram bins and sum them up to get an idea of how much energy exists in various frequency regions. This is performed by our Mel filterbank: the first filter is very narrow and gives an indication of how much energy exists near 0 Hertz. As the frequencies get higher our filters get wider as we become less concerned about variations.

Once we have the filterbank energies, we take the logarithm of them. This is also motivated by human hearing: we don't hear loudness on a linear scale. Generally to double the percieved volume of a sound we need to put 8 times as much energy into it. This means that large variations in energy may not sound all that different if the sound is loud to begin with. This compression operation makes our features match more closely what humans actually hear.

The final step is to compute the DCT of the log filterbank energies. There are 2 main reasons this is performed. Because our filterbanks are all overlapping, the filterbank energies are quite correlated with each other. The DCT decorrelates the energies which means diagonal covariance matrices can be used to model the features in e.g. a DTW classifier.

## What is Mel-Scale?

The Mel scale relates perceived frequency, or pitch, of a pure tone to its actual measured frequency. Humans are much better at discerning small changes in pitch at low frequencies than they are at high frequencies. Incorporating this scale makes our features match more closely what humans hear.

The formula for converting from frequency to Mel scale is:

$$M(f) = 1125 \ln(1 + f/700)$$

To go from Mels back to frequency:

$$M^{-1}(m) = 700(\exp(m/1125) - 1)$$

## MFCCs Implementation Steps

We have the Speech Signal of length 5 seconds recorded at a sampling rate of 8000Hz. So, this means we have 40,000 amplitude values for this time domain signal recorded. We now proceed with the following steps:

STEP 1: Divide the total Time Domain Speech Signal into FRAMES of width 25 ms (=200 samples) with an overlap between consecutive frames of about 10-15 ms (=120 samples). If speech doesn't divide into an integral value, pad extra zeros to the signal or cut down very few samples to the end of the signal.

In our project, we have 498 such frames. We finally would extract a set 13 coefficients per frame after the entire MFCC process.

STEP 2: Now, to each frame (total of 498) multiply with a smoothing function such as a Hamming Window, and take the Discrete Fourier Transform of it (for each frame). We have used fft() function of MATLAB for this.

Now, compute the Periodogram Estimate of the frame by taking the modulus squared of the components of the above taken Fourier Transform and divide it by the scalar N=200 (number of samples per frame).

STEP 3: Compute the Mel-spaced filterbank. This is a set of 10-30 (13 is standard) triangular filters that we apply to the periodogram power spectral estimate from step 2.

To get the filterbanks shown in the below figure we first have to choose a lower and upper frequency. Good values are 300Hz for the lower and 4000Hz for the upper frequency. Using the Frquency to Mel-scale formula, convert the upper and lower frequencies to Mels.

Now, take 13 additional points spaced linearly between the lower and upper Mel-scale values (as we need 13 filter banks). Convert these 15 Mel-linearly spaced points back to Frequencies using the Mel-to-Frequency conversion formula.

Now we create our filterbanks. The first filterbank will start at the first point, reach its peak at the second point, then return to zero at the 3rd point. The second filterbank will start at the 2nd point, reach its max at the 3rd, then be zero at the 4th etc. A formula for calculating these is shown in the paper by **Shen et al. [3] in EURASIP Journal (2012)** as follows:

$$H_m[k] = \begin{cases} 0 & k < f[m-1] \\ \dfrac{(k - f[m-1])}{(f[m] - f[m-1])} & f[m-1] \le k \le f[m] \\ \dfrac{(f[m+1] - k)}{(f[m+1] - f[m])} & f[m] \le k \le f[m+1] \\ 0 & k > f[m+1] \end{cases}$$

where $1 \le m \le M$ and the boundary points $f[m]$ are uniformly spaced in the Mel-scale

Where, f[m] is given by:

$$f[m] = \left(\frac{N}{F_s}\right) B^{-1}\left(B(f_l) + m\frac{B(f_h) - B(f_l)}{M+1}\right)$$

where $f_l$ and $f_h$ are the lowest and the highest frequencies of the filterbank, $F_s$ is the sampling rate, and the Mel-scale $B$ and its inverse $B^{-1}$ are given by

$$B(f) = 1125\ \ln(1 + f/700)$$

$$B^{-1}(b) = 700\left(e^{(b/1125)} - 1\right)$$

STEP 4: To calculate filterbank energies we multiply each filterbank with the power spectrum, then add up the coefficents. Once this is performed we are left with 13 numbers that give us an indication of how much energy was in each filterbank.

STEP 5: Take the log of each of the 26 energies from step 4. This leaves us with 26 log filterbank energies.

STEP 6: Take the Discrete Cosine Transform (DCT) of the 13 log filterbank energies to give 13 cepstral coefficents. Out of these, the first MFC Coefficient is just the sum of log signal energies, and therefore we remove it before processing to make our recognition Loudness-Variations Robust.

The resulting features (12 numbers for each frame) are called Mel Frequency Cepstral Coefficients.

## Pattern Matching- Dynamic Time Warping (The DTW Algorithm):

Now, once we got the MFC Coefficients of Speech Train Samples and Test Sample, we need an Algorithm for Classification designed specially for Speech Applications. The best match (lowest distance measure) is based upon dynamic programming. This is the DTW Algorithm.

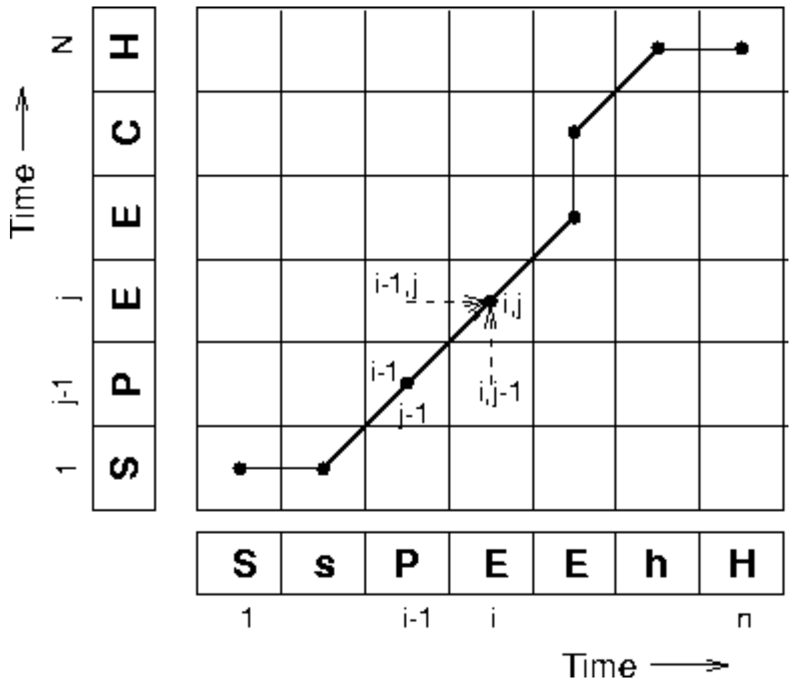
NOTE that here each speech train sample or test sample is associated with a set 12 coefficients per frame and the number of frames being = 498. So, we have 498 12-dimensional time sequence of vectors for each of Test and Trained samples, and we need to Match them! The distance between "Two Sequences of Vectors" is highly nontrivial and is achieved by DTW with extra advantages. The algorithm and description are mentioned below: We have not a single feature vector for each sample, but a 'set of feature vectors' that must be matched.

Since the feature vectors could possibly have multiple elements, a means of calculating the local distance is required. The distance measure between two feature vectors is calculated using the Euclidean distance metric. Therefore the local distance between feature vector x of signal 1 and feature vector y of signal 2 is given by,

$$d(x,y) = \sqrt{\sum_i (x_i - y_i)^2}$$

Speech is a time-dependent process. Hence the utterances of the same word will have different durations, and utterances of the same word with the same duration will differ in the middle, due to different parts of the words being spoken at different rates. To obtain a global distance between two speech patterns (represented as a sequence of vectors) a time alignment must be performed.

This problem is illustrated in below figure, in which a ''time-time'' matrix is used to visualize the alignment. As with all the time alignment examples the reference pattern (template) goes up the side and the input pattern goes along the bottom. In this illustration the input ''SsPEEhH'' is a 'noisy' version of the template “SPEECH”. The idea is that ‘h’ is a closer match to ‘H’ compared with anything else in the template. The input “SsPEEhH” will be matched against all templates in the system’s repository. The best matching template is the one for which there is the lowest distance path aligning the input pattern to the template. A simple global distance score for a path is simply the sum of local distances that go to make up the path.



Above figure shows the time alignment between the test and the training pattern.

To make the algorithm and reduce excessive computation we apply certain restriction on the direction of propagation. The constraints are given below.

- Matching paths cannot go backwards in time.
- Every frame in the input must be used in a matching path.

- Local distance scores are combined by adding to give a global distance.

This algorithm is known as Dynamic Programming (DP). When applied to template-based speech recognition, it is often referred to as Dynamic Time Warping (DTW). DP is guaranteed to find the lowest distance path through the matrix, while minimizing the amount of computation. The DP algorithm operates in a time-synchronous manner: each column of the time-time matrix is considered in succession (equivalent to processing the input frame-by-frame) so that, for a template of length N, the maximum number of paths being considered at any time is N.

If D(i,j) is the global distance up to (i,j) and the local distance at (i,j) is given by d(i,j):

$$D(i,j) = \min[D(i-1,j-1), D(i-1,j), D(i,j-1)] + d(i,j)$$

Given that D(1,1) = d(1,1) (this is the initial condition), we have the basis for an efficient recursive algorithm for computing D(i,j). The final global distance D(n,N) gives us the overall matching score of the template with the input. The input word is then recognized as the word corresponding to the template with the lowest matching score.

We have implemented the above DTW algorithm using recursion in MATLAB.

## Results of our MATLAB Project

We have Trained our program with a total of **67 Train Speech Samples** of Digits 1, 2 and 3.

We Tested it on a total of **106 Test Speech Samples** of Digits 1, 2 and 3, by recording the voices of various people. The results have been as follows:

| **Digit Class** | **Accuracy** |
|---|---|
| 1 | 87.5 % |
| 2 | 90 % |
| 3 | 96.1538 % |
| **Overall Accuracy:** | **90.566 %** |

Therefore, our results are satisfying.

## Conclusion

MFCCs provide a very powerful way to extract efficient Feature Vectors for performing Speech/Voice Recognition. Then a special method of DTW is applied to perform Feature Matching. These state-of-the-art methods have been discussed above.

We are very thankful to Prof. Prithwijith Guha and Prof. Rohit Sinha for their constant encouragement and support in building this project for submission in the EE322M Signal Processing Course at IIT Guwahati.

## References

[1] S. Davis and P. Mermelstein, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. **28** – 4, p. 357-366 (1980)

[2] H. Sakoe and S. Chiba, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. Assp-**26** (1978)

[3] L. Shen, C. Yeh and S. Hwang, EURASIP Journal on Audio, Speech and Music Processing 2012, **2012**:28

[4] L. Munda, M. Begam and I. Elamvazuthi, Journal of Computing, Vol. **2**, Issue 3 (2010)

[5] L. Rabiner and B. Juang, Fundamentals of Speech Recognition, First Edition, ISBN-13: 978-0130151575