CS571 Project Report

Linear Predictive Coding

Rajesh R
S21005 - MS Scholar
23 November 2021

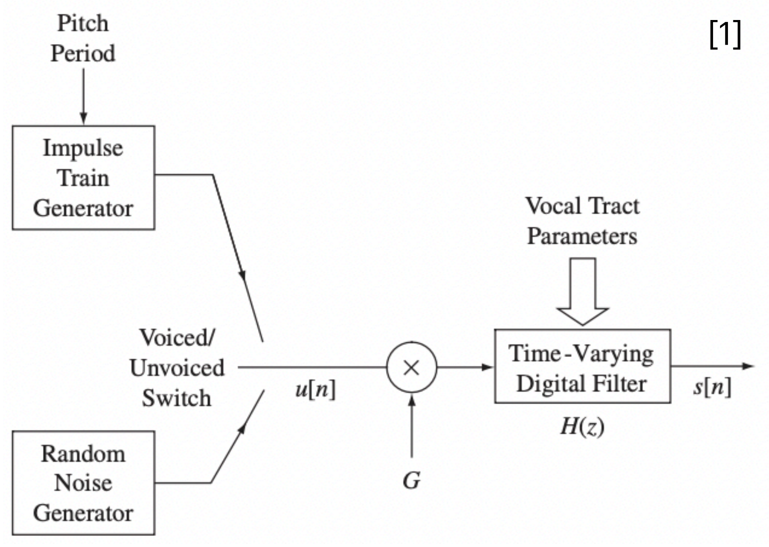
Linear Predictive Coding

Linear predictive analysis of speech signals is one of the most powerful speech analysis techniques. This method has become the predominant technique for estimating the parameters of the discrete-time model for speech production (i.e., pitch, formants, short-time spectra, vocal tract area functions) and is widely used for representing speech in low bit rate transmission or storage and for automatic speech and speaker recognition. The importance of this method lies both in its ability to provide accurate estimates of the speech parameters and in its relative ease of computation.[1] [1]

Basic Idea & Principle

LPA of speech samples can be approximated as a linear combinations of past speech signals. By minimising sum of squared differences over finite interval between the actual speech samples and linearly predicted ones, a unique set of predictor coefficients can be determined.

$$H(z) = \frac{S(z)}{U(z)} = \frac{G}{1 + \sum_{k=1}^{p} a_k z^{-k}}$$



[1]

[1] Text Extracted from Pearson. Rabiner, Lawrence R. Schafer, Ronald W - Theory and applications of digital speech processing-Pearson_Prentice Hall

Speech samples $s(n)$ are related to the excitation $u(n)$ by simple difference equation,

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + Gu(n)$$

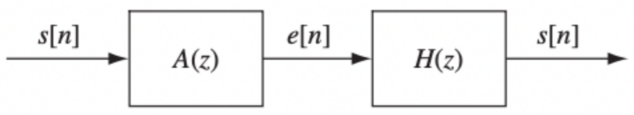
Between the pitch pulses $Gu(n)$ is zero. Therefore, $s(n)$ can be predicted from a linearly weighted summation of past samples. However if $Gu(n)$ is included then we can predict $s(n)$ only by approximation. If $Gu(n) = 0$ between the pulses, we get,

$$s(n) = a_1 s(n-1) + a_2 s(n-2) + \ldots\ldots + a_p s(n-p)$$

That is the nth sample can be viewed as a linear combination of p past samples.

Inverse Filtering

The aim is to determine the all zero filter which is the inverse of the vocal tract model. Inverse filtering of $s(n)$, to give the error signal, $e(n)$, and direct filtering of the error signal, $e(n)$, to reconstruct the original speech signal, $s(n)$.



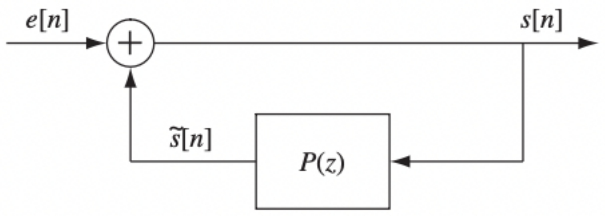
[1]

Energy = $e^2(n)$ and goal is to minimise the $e^2(n)$. Note that inverse of all pole filter is all zero filter.

Linear Prediction Model

$\hat{s}(n) = \sum_{k=1}^{p} \alpha_k s(n-k)$ where α_k is the predictor coefficients. The error between the actual signal $s(n)$ and predicted values $\hat{s}(n)$ is given by,

$$e(n) = s(n) - \hat{s}(n) = s(n) - \sum_{k=1}^{p} \alpha_k s(n-k)$$



[1]

The figure[2] shows the linear predictor model for reconstructing the original signal back. From the block diagram,

$$H(z) = \frac{S(z)}{U(z)} = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}} \text{ Vocal tract (IIR filter)}$$

$$\frac{E(z)}{S(z)} = 1 - \sum_{k=1}^{p} a_k z^{-k} \text{ is Linear Predictor (FIR)}$$

$$\frac{E(z)}{U(z)} = G \frac{1 - \sum_{k=1}^{p} a_k z^{-k}}{1 - \sum_{k=1}^{p} a_k z^{-k}} = G$$

$$\frac{E(z)}{U(z)} = G \rightarrow E(z) = GU(z)$$

Therefore, $e(n) = Gu(n)$. Ideallt we need a technique to produce the coefficients of the speech production model α_k. If we determine the correct coefficients, then the error signal $e(n) = Gu(n)$ and the linear predictor is called an **Inverse filter**.

The transfer function of the inverse filter is given by,

$$A(z) = \frac{E(z)}{S(z)} = 1 - \sum_{k=1}^{p} a_k z^{-k}$$

A by-product of the LPC analysis is the generation of the error signal $e(n)$ and it is a good approximation to the excitation source. It is expected that the prediction error $e(n)$ will be large for voiced speech at the beginning of each pitch period. Thus the pitch period can be determined by detecting the positions

[2] Figure took from Pearson. Rabiner, Lawrence R. Schafer, Ronald W - Theory and applications of digital speech processing-Pearson_Prentice Hall

of the samples of $e(n)$ which are large & defining the period as the difference between pairs of sample of $e(n)$ which exceed a reasonable threshold. Alternatively the pitch period can be determined by performing an autocorrelation analysis on $e(n)$ and detecting the largest peak in the appropriate range. Another way of interpreting why error signal is valuable for the pitch detection is the observation that the spectrum of the error signal is approximately flat, thus the effects of formants have been eliminated in the error signal.

In conclusion, we can say that except for a sample at the beginning of each pitch period, every sample of the voiced speech waveform can be predicted from the past p samples.

For voiced speech $e(n)$ would consist of a train of impulses. $e(n)$ would be most of the time except at the beginning of the pitch period. The prediction is not valid at instants of time where the input pulses occur. The prediction error $e(n) = Gu(n)$ would be scaled pulse train for voiced speech frames and scaled random noise for unvoiced speech frames. Because of the time varying nature of the speech the predictor coefficients should be estimated from short segments of speech signal.

Implementation of LPC

Representing spectral envelope of a digital signal of speech in compressed form and encoding good quality speech at low bit rate.

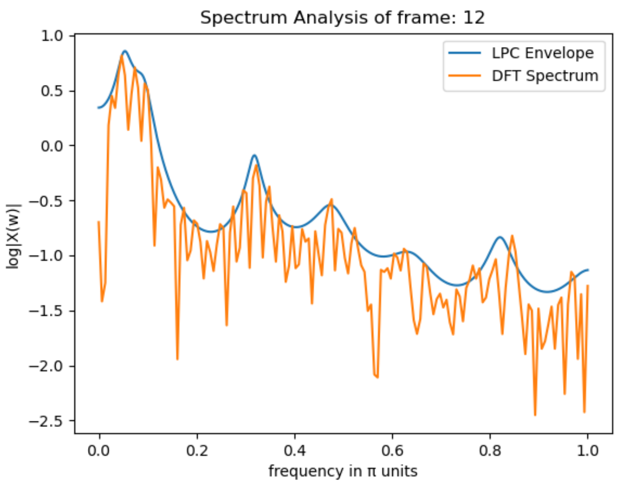
We have used short time processing of speech signals and segregated into frames and applied LPC analysis using the inbuilt python module librosa.lpc() and plotted its LPC Spectrum with its DFT spectrum. Later obtaining the Inverse filter. Using this inverse filter, we can get the residual signal $e(n)$. Using residual signal with LPC filter we get synthetic speech frames and finally we are getting compressed signal by overlap add method.

Algorithm

```
1.  Imports various useful packages
2.  input_speech, fs = soundFile.read(inputAudio.wav)
3.  Window = scipy.windows.hamming(fs, False)
4.  function frameblocks(signal, window, o=0.5):
        #creates input speech into frames
        returns frames
5.  For each frame:
        dft_frame = numpy.fft.fft(frame)
        plot numpy.log10(abs(dft_frame))
        lpc_coeff = librosa(frame, order) #Returns order+1 length vector
        w, h = scipy.freqz(lpc_coeff, 1)
        plot(w, numpy.log10(abs(h))
        poles = 1
        residual = signal.lfilter(coefficients, poles, frame) #Inverse filter
        synthetic = signal.lfilter([1], coefficients, residual)
        return all synthetic frames
6.  function add_frame_blocks(signal, windows o=0.5)
        #recreates the signal
        return output signal
7.  soundFile.write(output_signal, fs)
8.  end
```

Results & Analysis

(1) Tried with the **human voice** audio should.wav and here are the results:



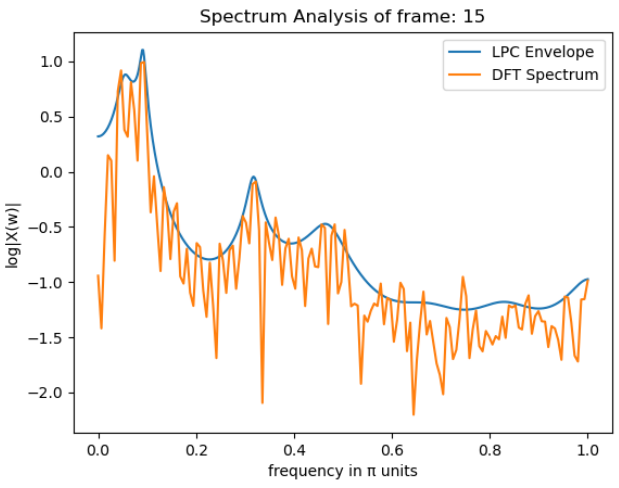
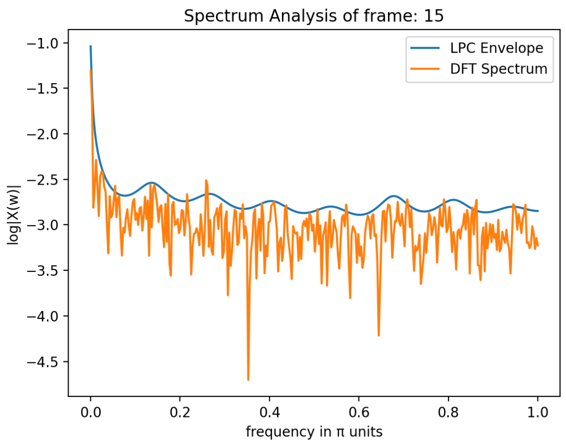


Fig (1) The LPC Envelope & DFT of frame 12. Fig (2) The LPC Envelope & DFT of frame 15

(2) Tried with **band music** audio CatinaBand3.wav file and the results are:



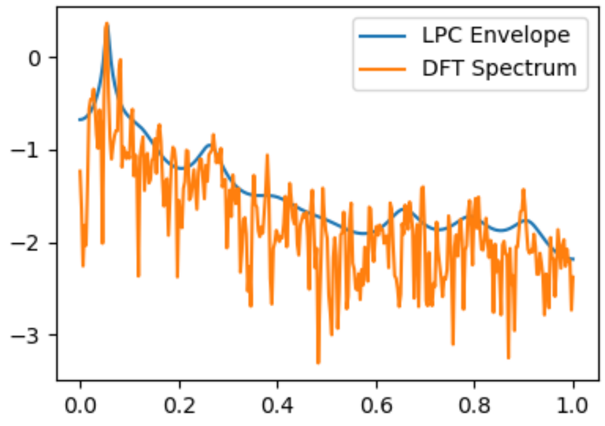
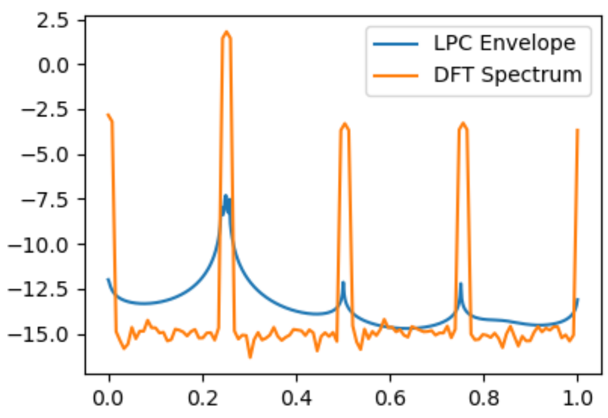


Fig (1) The LPC Envelope & DFT of frame 15. Fig (2) The LPC Envelope & DFT of frame 100

(2) Tried with **sinusoidal generated** audio file and the results are:



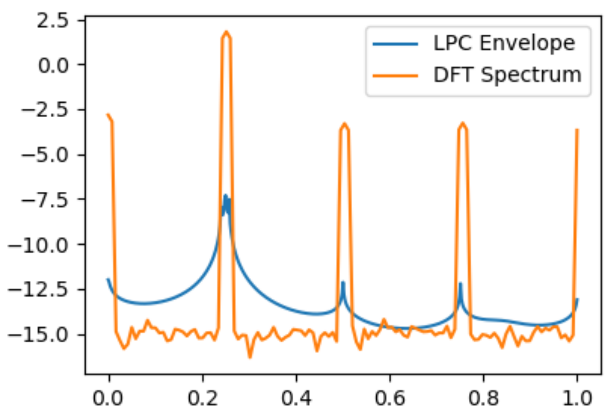


Fig (1) The LPC Envelope & DFT of frame 10. Fig (2) The LPC Envelope & DFT of frame 100

Conclusion

I have created a user friendly simple UI to perform all these tasks and the user manual is attached. The code works with all the different types of audio signals. I have learned to write code on myself to implement LPC and to perform LPA to estimate all kinds of its speech parameters. I have also managed to create and UI by this project.

Since I have used librosa module to estimate LPC coefficients, the module themselves have a flaw and they failed to process audio which resulted in the ill-conditioned system. So, this project also not supports that type of audio inputs. This limitations can be handled by using powerful computational techniques.

GitHub Link

Have a look at the **source code**, **user manual**, results obtained and various resources that used in the project

https://github.com/Rajesh-Smartino/Linear-Predictive-Coding-of-Speech-Signals

References

1. Pearson. Rabiner, Lawrence R. Schafer, W Ronald - “Theory and applications of digital speech processing” - *Pearson_Prentice Hall*