homework #5

Linear Prediction Analysis

EE 8993: Fundamentals of Speech Recognition

December 6, 1998

submitted to:

Dr. Joseph Picone

submitted by:

Suresh Balakrishnama

Institute for Signal and Information Processing
Department of Electrical and Computer Engineering
Mississippi State University
MS 39762, USA
Email: balakris@isip.msstate.edu

1. INTRODUCTION

Linear Prediction Analysis has been among the most popular methods for extracting spectral information from speech. Linear Prediction analysis is an important method for finding the shape of a spectrum. In linear prediction the signal is modeled as a linear combination of the its past values and present and past values of a hypothetical input to a system whose output is the given signal. Each continuous-time signal $s(t)$ is sampled to obtain a discrete-time signal $s(nT)$,also known as time-series, where n is an integer variable and T is the sampling interval.

2. Problem Description

Implement a capability to plot a signal's FFT spectrum, and the gain-matched spectrum produced by a linear prediction model. The tool must read speech from a binary file (assume 16 bit linear sampling), and allow the user to select the following:

- sample frequency of the signal
- preemphasis constant
- window duration in secs
- center time for the window is secs
- a rectangular or hamming window
- the linear prediction order

You can approach this problem one of two ways

- implement the signal processing in matlab and figure out how to manipulate binary files into matlab
- implement everything in C++ (preferred)

In the latter case, the interface should be something like this:

```
my_prog 8000.0 0.95 0.03 28.7 1 10 foo.raw | xmgr -source stdin
```

The net result should be a plot of the signal spectrum computed using the following parameters:

- fs = 8 kHz
- preemphasis = argv[1]
- window_duration = argv[2]
- center time of the window = argv[3]
- hamming window = yes
- lp_order = argv[4]

and plotted on a log amplitude vs. linear frequency scale. The spectra of the corresponding linear prediction model should be plotted as well. Xmgr accepts multiple sets of data, so simply print your xy points for both plots to stdout, with the second set separated by a newline, and xmgr will take care of the rest.

You can use a DFT to compute the spectrum of the signal, or a zero-stuffed fft. The important thing is to only use window_duration number of samples of real data (note that window_duration is specified in secs).

For example,

```
my_prog.exe 8000.0 0.95 0.03 3.0 12 | xmgr -source stdin
```

should produce a signal and lp model spectrum for a 30 msec window of the signal centered at 3 secs. The lp analysis will be of order 12. A preemphasis filter 1 - 0.95z** is applied to the data. For most of you, this should be a useful tool to have around. Feel free to pull the LP analysis software of the net. The main thing is to get the visualization component working - and to understand gain matching of the two spectra.

The resulting plots will typically have about a 60 dB dynamic range for studio quality data.

3. Description of Algorithms

One of the most powerful models currently in use is that where a signal s_n is considered to be the output of some system with some unknown input u_n such that the following relation holds:

$$s_n = -\sum_{k=1}^{p} a_k s_{n-k} + G\sum_{l=0}^{q} b_l u_{n-1} \qquad (1)$$

where $b_0 = 1$ and $a_k,\ 1 \le k \le p,\ b_l,\ 1 \le l \le q$, and the gain G are the parameters of the hypothesized system. The output from equation (1) s_n is a linear function of past outputs and present and past inputs. The signal s_n is predictable from linear combinations of past outputs and inputs. This is the reason for this system to be called linear prediction. The predicted value is a linear combination of previous values in the signal. Linear prediction error is an important term and the parameters chosen in LP analysis to determine prediction coefficients should be such as to minimize linear prediction error. For a speech signal s_n, predicted values is given by

$$\tilde{s}(n) = \sum_{k=1}^{p} \alpha_k s(n-k) \qquad (2)$$

and the prediction error is given by

$$e(n) = s(n) - \tilde{s}(n) = s(n) - \sum_{k=1}^{p} \alpha_k s(n-k) \qquad (3)$$

According to Parseval’s theorem, if error is small in time domain error is small in frequency domain also and this error should be minimized to the least. The error can be minimized by finding the best or optimal value of α_k. To explain the computation involved for α_k let us consider a short-time prediction error:

$$E = \sum_n e^2(n) \tag{4}$$

$$= \sum_n \left(s(n) - \sum_{k=1}^{p} \alpha_k s(n-k) \right)^2$$

$$= \sum_n s^2(n) - \sum_n \left(2s(n) \sum_{k=1}^{p} \alpha_k s(n-k) \right) + \sum_n \left(\sum_{k=1}^{p} \alpha_k s(n-k) \right)^2 \tag{5}$$

$$= \sum_n s^2(n) - 2 \sum_{k=1}^{p} \alpha_k \sum_n s(n)s(n-k) + \sum_n \left(\sum_{k=1}^{p} \alpha_k s(n-k) \right)^2$$

The error can be minimized with respect to α_l for each $1 \le l \le p$ by differentiating E and setting the result equal to zero.

$$\frac{\partial E}{\partial \alpha_l} = 0 = -2\sum_n s(n)s(n-l) + 2\sum_n \left(\sum_{k=1}^{p} \alpha_k s(n-k) \right) s(n-l) \tag{6}$$

Rearranging terms we get,

$$\sum_n s(n)s(n-l) = \sum_{k=1}^{p} \alpha_k \left(\sum_n s(n-k)s(n-l) \right) \tag{7}$$

Equation (7) is known as linear prediction equation and α_k are known as linear prediction coefficients or predictor coefficients.

Levinson-Durbin’s Recursion Method

The L-D recursion is a recursive-in-model-order solution for the autocorrelation equations. The solution for desired order-M model is successively built from lower-order models, beginning with the 0th order predictor which is no predictor coefficient at all. This method uses autocorrelation coefficients to determine the prediction coefficients and reflection coefficients. The prediction coefficients can be computed using the following equations:

$$
\begin{gathered}
E_0 = r(0) \\
k_i = \left(R(i) - \sum_{j=1}^{i-1} \alpha_j^{(i-1)} R(i-j) \right) \Big/ E^{(i-1)} \\
\alpha_i^{(i)} = k_i \\
\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)} \\
E^{(i)} = (1 - k_i^2) E^{(i-1)}
\end{gathered}
\tag{8}
$$

with $1 \le i \le p$ and $1 \le j \le i-1$ where i indicates the current iteration, $i-1$ indicates the previous iteration, j is the total number of iterations, and p is the order of the prediction. E is the error term, R is the autocorrelation coefficient, k is the reflection coefficient, and α is the predictor coefficient.

DFT Method

Like the previous case, there are many algorithms to calculate the DFT coefficients. However, here since we focus on the linear prediction task and not the DFT, we did not use any fast implementation of the DFT calculation but just straight implementation from the DFT equation, Equation (9).

$$
\beta(i) = \sqrt{\left(\sum_{j=0}^{N-1} x(j) \cos\left(\frac{2\pi i j}{p} \right) \right)^2 - \left(\sum_{j=0}^{N-1} x(j) \sin\left(\frac{2\pi i j}{p} \right) \right)^2} \tag{9}
$$

with $1 \le i \le p$ where i indicates the current iteration, N is the total number of iterations, and p is the order of the DFT.

4. Results

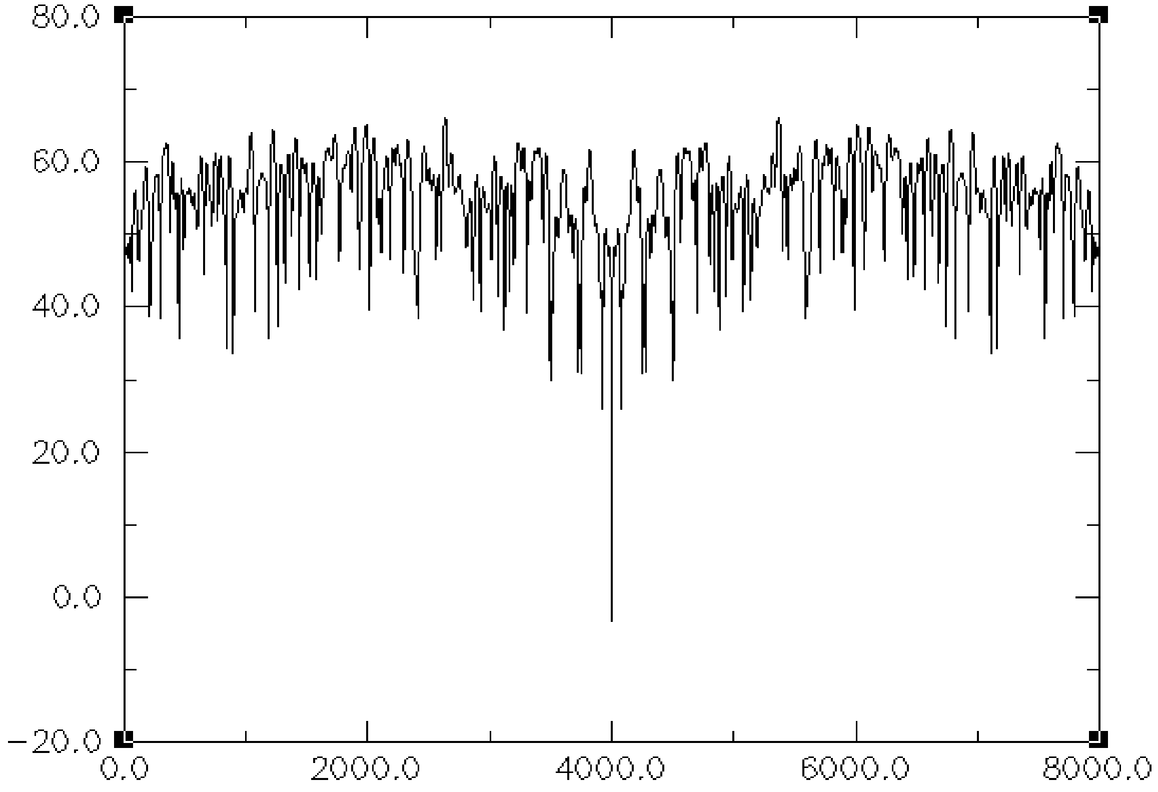

Figure 1. Plot showing DFT spectrum of speech file

5. Conclusions

The DFT spectrum was obtainable but the computation of LP derived spectrum became difficult and its plot could not be obtained demonstrating the effect of LP derived spectrum over DFT spectrum. The error between the LP-derived and DFT spectrum could not be analyzed. But based on theory, the error becomes smaller as the LP model order gets higher and higher.

6. References

[1] J.Makhoul, “Linear Prediction: A Tutorial Review”, *Proceedings of the IEEE*, Vol. 63, April 1975.

[2] J.Picone, “ECE 8993: Fundamentals of Speech Recognition Lecture Notes”, Mississippi State University, May 1998.