A DYNAMIC CAUSAL FILTER APPROACH TO SPEECH TRAJECTORY SEGMENTATION

G. Platt and M.D. Alder

Department of Mathematics
The University of Western Australia

ABSTRACT - We consider the possibility of segmenting speech into phonemic segments using an idea that stems from the concept of Linear Predictive Coding. Linear Trajectory Predictive Segmentation, or LTPS, involves performing linear predictive analysis on each dimension of the speech trajectory, and assuming that points in the trajectory where the error of our prediction is large correspond to points where phonemic transitions are occurring. LTPS has two basic parameters that alter the prediction made. The ability of this method to accurately segment speech into phonemes is analysed, for various combinations of these parameter values.

INTRODUCTION

The basic idea behind linear predictive analysis involves coding speech samples as linear combinations of past speech samples. Finite intervals of the speech sample are predicted with a set of coefficients that minimise the sum of the squared differences between the actual speech samples and the linearly predicted ones. The coefficients are simply the weightings for each previous sample used in the linear prediction.

We decided to perform linear predictive analysis on each frame, x_i, of a feature vector representation of our speech sample rather than the original speech sample $S(n)$. We are thus looking at a high dimensional speech trajectory, so we have to make predictions for each dimension of the space we are working in. If the feature vector representation is n-dimensional, then $x_i = (x_{1,i}, x_{2,i}, ..., x_{n,i})^T$, so we would get the following system of equations for a linear trajectory predictor of order p, memory m:

$$\begin{array}{llllll} x_{n-1,i} & = a_{1,i}x_{n-2,i} & +a_{2,i}x_{n-3,i} & +\ldots & +a_{p,i}x_{n-p-1,i} & \text{for } i = 1,2,\ldots,n \\ x_{n-2,i} & = a_{1,i}x_{n-3,i} & +a_{2,i}x_{n-4,i} & +\ldots & +a_{p,i}x_{n-p-2,i} & \text{for } i = 1,2,\ldots,n \\ & & \vdots & & & \\ x_{n-m,i} & = a_{1,i}x_{n-m-1,i} & +a_{2,i}x_{n-m-2,i} & +\ldots & +a_{p,i}x_{n-p-m,i} & \text{for } i = 1,2,\ldots,n \end{array} \quad (1)$$

We now have $n \times p$ unknown predictor coefficients to solve for, and $a_{i,j}$ corresponds to the i-th predictor coefficient in the j-th dimension.

A common view of the structure of speech is that it consists of regions of quasi-stationarity in the spectral envelope, interspersed with abrupt changes in behaviour. These abrupt movements in the spectral envelope correspond to short periods of unpredictable movement in the speech trajectory, and hence at these points we assume that our linear predictor will give high prediction values. So, if we make linear predictions of each frame of our speech trajectory, and graph the prediction error we hope that regions where our prediction error is high will correspond to phonemic transitions in the original speech. We thus investigated how well this method detects these phoneme transitions for various values of p and m.

COMPUTATION OF THE LINEAR TRAJECTORY PREDICTION METHOD

The best predictor coefficients to use in the system of equations in 1 are of course those that minimise the error. The system essentially defines m hyperplanes in p dimensional space, and the optimum predicting coefficients will correspond to a point in the space whose sum total squared Euclidean distance from all hyperplanes is a minimum.

Our prediction error for frame n, dimension i is thus:

$$
\begin{aligned}
E_{n,i} &= (x_{n-1,i} \quad -a_{1,i}x_{n-2,i} \quad +a_{2,i}x_{n-3,i} \quad +\dots \quad +a_{p,i}xn-p-1,i)^2 \\
&+ (x_{n-2,i} \quad -a_{1,i}x_{n-3,i} \quad +a_{2,i}x_{n-4,i} \quad +\dots \quad +a_{p,i}xn-p-2,i)^2 \\
&\qquad\qquad\qquad\qquad +\dots \\
&+ (x_{n-m,i} \quad -a_{1,i}x_{n-m-1,i} \quad +a_{2,i}x_{n-m-2,i} \quad +\dots \quad +a_{p,i}xn-m-p,i)^2
\end{aligned}
$$

We hence find our best predictor coefficients for frame n, dimension i by setting $\partial E_{n,j}/\partial a_{ji} = 0$, for $j = 1, 2, ..., p$, which gives us the following equation:

$$
\begin{bmatrix}
\sum_{j=1}^{m} x_{n-j-1,i}^2 & \sum_{j=1}^{m} x_{n-j-1,i}x_{n-j-2,i} & \cdots & \sum_{j=1}^{m} x_{n-j-1,i}x_{n-j-p,i} \\
\sum_{j=1}^{m} x_{n-j-2,i}x_{n-j-1,i} & \sum_{j=1}^{m} x_{n-j-2,i}^2 & \cdots & \sum_{j=1}^{m} x_{n-j-2,i}x_{n-j-p,i} \\
\vdots & \vdots & \ddots & \vdots \\
\sum_{j=1}^{m} x_{n-j-p,i}x_{n-j-1,i} & \sum_{j=1}^{m} x_{n-j-p,i}x_{n-j-2,i} & \cdots & \sum_{j=1}^{m} x_{n-j-p,i}^2
\end{bmatrix}
\begin{pmatrix} a_{1,i} \\ a_{2,i} \\ \vdots \\ a_{p,i} \end{pmatrix}
$$

$$
= \begin{pmatrix} \sum_{j=1}^{m} x_{n-j-1,i}x_{n-j,i} \\ \sum_{j=1}^{m} x_{n-j-2,i}x_{n-j,i} \\ \vdots \\ \sum_{j=1}^{m} x_{n-j-p,i}x_{n-j,i} \end{pmatrix}, \text{ for } i = 1, 2, \dots, n
$$

or put simply, $\Phi a = \psi$, where Φ is the symmetric $p \times p$ matrix and a is the vector of predictor coefficients. Non-singularity of the matrix Φ is almost always ensured if the memory of our predictor, m is equal to or greater than the order, p of the predictor. The only other way that this matrix could be singular is if some of the m hyperplanes defined in the system of equations are parallel, and of course when dealing with real data, this rarely occurs, and this situation never occured in our testing.

We now have a unique matrix of predictor coefficients for each frame in the speech trajectory. If we denote the matrix for frame t as $A(t) = [a_{ij}]_t$, we can use this matrix to make a prediction, $\hat{x}_t$ of what we think the frame x_t should be.

$$\hat{x}_t = X(t)A(t)$$

where

$$X(t) = \begin{bmatrix} x_{t-1} & x_{t-2} & \cdots & x_{t-p} \end{bmatrix}$$

We then define our total prediction error for each frame, E_t to be

$$E_t = |x_t - \hat{x}_t|^2 = \sum_{i=1}^{n} (x_{t,i} - \hat{x}_{t,i})^2$$

We now have a sequence of error values, that is, an error graph, which reflects the linear predictability of the speech sample at each frame. If there is a direct correlation between high total error values and points in the speech trajectory that correspond to transitions between phonemes, then this can be a useful way of segmenting our speech trajectories into phonemic elements. Two example error graphs with different (order, memory) pairs are displayed in figures 3 and 4.

We analysed the performance of the LTPS method with the following combinations of order and memory.

Order	Memory
2	2,3,4,5 and 6
3	3,4,5 and 6
4	4,5 and 6

All these types of LTP are compared with one another in an effort to find the combination of order and memory that achieves the best results.

ANALYSIS

We analysed the performance of our algorithms by producing error graphs from speech data that was obtained from the TIMIT data base produced by the National Institute of Standards and Technology (U.S.A.). The 12 (order, memory) combinations were all used to produce error graphs for 160 sentences. Four speakers, two male and two female, were taken from each of the eight dialect regions, and five sentences from each speaker were used. The sentences had been recorded in low noise conditions, sampled at 16 kHz and digitized to 16 bit resolution.

A fast Fourier transform with a 32 millisecond window and 29 millisecond overlap was then performed on the sample, and the average frequency energies were binned into 12 bandwidths covering the audio spectrum. We thus have the result that all speech samples were represented as a discrete trajectory in $\mathbb{R}^{12}$. The errorgraphs were then compared to the phoneme transition points that are provided by the TIMIT data base.

Ideally, we would want our graphs to have higher values at frames corresponding to phoneme transitions, and lower values at frames situated in the middle of phonemes, similar to what is shown in figure 1:

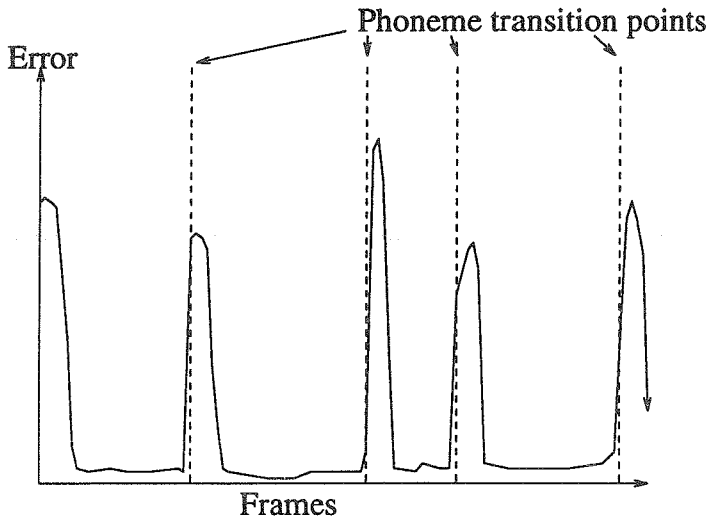

Figure 1: The ideal error graph

Note that the actual values associated with the graph are not important, as any change in scale can be accounted for by a similar change in scale to a threshold value that we use to decide where to segment trajectories.

For each LTP case, we should get high error values at and just after the point where there is a defined phoneme transition. This is because a p-th order linear predictor, with memory m, predicts the next frame from the previous $m + p$ frames, and so predictions of the first $m + p$ frames after a phoneme transition will have used some frames from two distinct phoneme states.

A performance measure function was devised to allow us to compare our error graphs quantitatively. Put simply, this function adds all the relative errors of frames at and just after each phoneme transition, and

subtracts all relative errors of all the other frames. The error values that are added, that is the error for the frames at and just after phonemic transitions, are weighted so that a constant error graph will give a performance rating of zero. These particular error values are also weighted by another factor that incorporates the distance of these - hopefully unpredictable - frames from the actual transition point. In other words the frame that is at the point of transition is given more weight than the frame next to it and so on.

The relative error for each frame is calculated by taking the actual error, dividing by the average error of all frames inside that particular phoneme and squaring the result. Using relative errors rather than the actual error values gives the performance function scale invariance, which is ideal for our purpose.

To assist in our description of our performance measure, let the frames of the speech trajectory that correspond to a phoneme transition point, as defined in the phonetic transcription information provided by the TIMIT database, be labelled S_x, where S_i corresponds to the i-th phoneme transcription in U, and let the number of phoneme transitions be φ. Also let $\check{S}_i = S_i + p + m + 3$ Each frame requires the previous $p + m$ frames to obtain our linear trajectory prediction error values, and so the frames in the regions $\{S_i, S_{i+1}, ..., \check{S}_i\}$ should have higher prediction errors. The extra three frames in both cases allows for inaccuracy of approximately ± 5 milliseconds in the original phonetic transcription alignment. The performance indicator, denoted $\Pi(E, S, \check{S})$ is defined as follows:

$$\Pi(E, S, \check{S}) = \sum_{i=1}^{\varphi} \sum_{j=S_i}^{S_{i+1}} f_i(x_j)$$

where

$$f_i(x_j) = \begin{cases} -(\frac{E_j}{\alpha_i})^2 & , \check{S}_i < j < S_{i+1} \\ X_i W_{j-S_i} (\frac{E_j}{\alpha_i})^2 & , S_i \leq j \leq \check{S}_i \end{cases}$$

where

$$\alpha_i = \frac{\sum_{j=S_i+1}^{S_{i+1}} E_j}{S_{i+1} - S_i}$$

$$X_i = \frac{S_{i+1} - \check{S}_i}{\check{S}_i - S_i}$$

and

$$W_n = W_0 - \frac{(\check{S}_i - S_i)(W_0 - 1)}{\sum_{i=1}^{\check{S}_i - S_i} i^2} n^2$$

where W_0 is the amount of weighting given to the total predictor error at the point of phoneme transition, which is

$$W_0 = \frac{1}{2} \left(\frac{(\check{S}_i - S_i)^3}{(\check{S}_i - S_i)^3 - \sum_{i=1}^{\check{S}_i - S_i} i^2} + 1 \right)$$

An error graph that has relatively high values at or just after phoneme transitions, and low values elsewhere will score a higher performance value than an error graph that doesn't do so. Hence error graphs that do the right thing will score well. Figure 2 is a graph of the average performance ratings that each of the LTP algorithms achieved over all dialect regions. The performance ratings displayed refer to the performance of the algorithm on each sentence, averaged over all sentences.

The last column labelled "MLR" corresponds to the performance of another method of segmenting the speech trajectory, involving Maximum Likelihood Methodologies, explained by Algazi, Brown, Ready, Irvine, Cadwell & Chung (1993). This method also produces a graph which aims to associate high values in the graph with points of sub-word transitions. These sub-words do not correspond directly to phonemes, and hence it is no surprise that the algorithm performed indifferently when analysed with our defined performance measure.

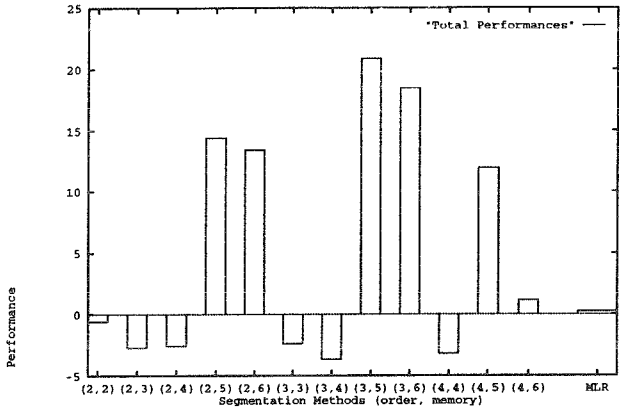

Figure 2: Average performance ratings over all Dialect Regions.

As figure 2 shows, the best overall performance was achieved by the third order linear predictor with memory five. However further analysis of this particular algorithm revealed that it only performed well during transitions involving fricatives, for example nasal-fricative transitions, fricative vowel transitions and so on, with the exception of stop-vowel and vowel-stop transitions, which also performed well. Unfortunately other transitions performed indifferently. It is thus evident that this algorithm cannot be used on its own as a reliable means of segmenting speech, however it shows some promise at detecting certain types of phoneme transitions.

POSSIBLE IMPROVEMENTS TO THE METHOD

The LTPS algorithm could be refined, in an effort to obtain better results, in the following two ways.

- The resultant error graphs could be smoothed to filter out random spikes in the graph.

-We could change this by replace our n equations for each frame with the one matrix equation, as shown below:

$$x_n = A_1 x_{n-1} + A_2 x_{n-2} + \cdots + A_p x_{n-p}$$

This draws all dimensions of the space into the one equation, which seems intuitively appealing, but increases the complexity of the problem by a factor of n.

REFERENCES

M.D. Alder, 'Principles of Pattern Classification: Statistical, Neural Net and Syntactic methods of getting robots to see and hear', (available by anonymous ftp from ciips.ee.uwa.edu.au in directory pub/syntactic/book) 1994.

V.R. Algazi, K.L. Brown, M.J. Ready, D.H. Irvine, C.L. Cadwell & S. Chung, 'Transform Representation of the Spectra of Acoustic Speech Segments with Applications-I: General Approach and Application to Speech Recognition', *IEEE Transactions on Speech and Audio Processing*, vol 1, No.2, 1993, 180-186.

National Institute of Standards and Technology, 'Getting Started With The DARPA TIMIT CD-ROM: An Acoustic Phonetic Continuous Speech Database', CD-ROM Documentation (Prototype Version), 1988.

L.R. Rabiner & R.W. Schafer, '*Digital Processing of Speech Signals*', Prentice Hall, Englewood Cliffs, New Jersey, 1978.

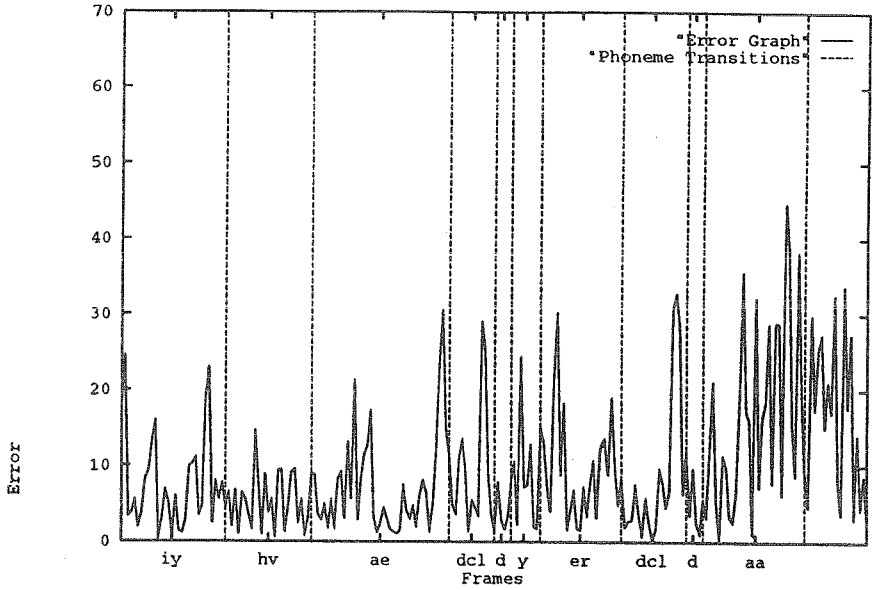

Figure 3: Example errorgraph for the LTP segmentation method (order=3, memory=3)

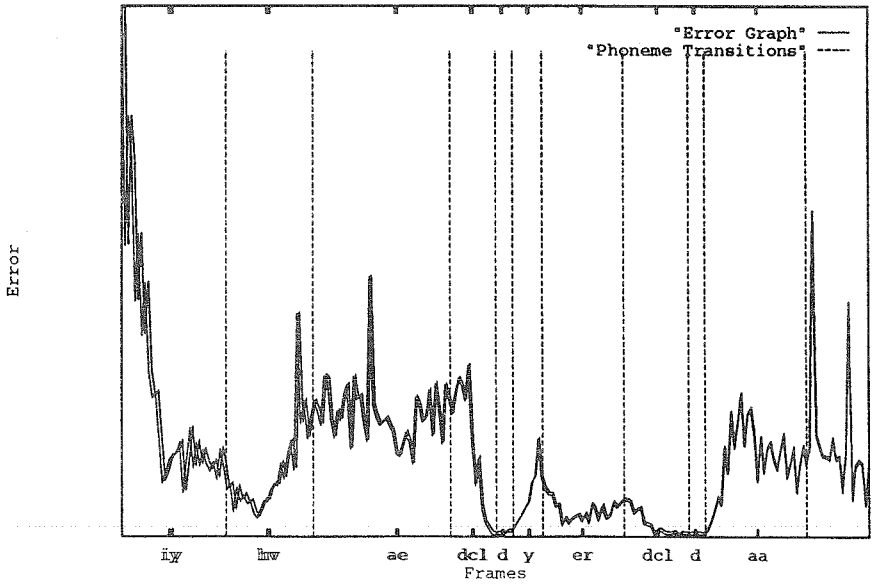

Figure 4: Example errorgraph for the LTP segmentation method (order=3, memory=5)