GRADIENT-DESCENT BASED WINDOW OPTIMIZATION FOR LINEAR PREDICTION ANALYSIS

Wai C. Chu

DoCoMo USA Labs – Mobile Media Laboratory
181 Metro Drive, Suite 300, San Jose, CA 95110, U.S.A.
wai@docomolabs-usa.com

ABSTRACT

The autocorrelation method of linear prediction (LP) analysis relies on a window for data extraction; we propose an approach to optimize the window based on gradient-descent. It is shown that the optimized window has improved performance with respect to popular windows, such as Hamming. The technique has potential in quality improvement for many LP-based speech coders.

1. INTRODUCTION

The autocorrelation method of LP analysis [1] is widely adopted by many modern speech coding algorithms. The basic procedure consists of signal windowing, autocorrelation calculation, and solving the normal equation leading to the LP coefficients. The role of the window is in the isolation of a frame of the input signal so that the resultant LP coefficients reflect the properties of the particular frame. There are many types of windows utilized by various speech coding standards, many of them rely on the following expression for autocorrelation computation:

$$R[l] = \sum_{k=l}^{N-1} w[k]s[k]w[k-l]s[k-l]\,. \qquad (1)$$

In the above equation, $R[l]$ represents the autocorrelation values, with l = 0, 1, 2, …, being the time-lag of interest. The sequence $w[n]$, n = 0 to N-1 is the window of length N; while the input sequence is denoted by $s[n]$. It is assumed that $w[n]$ = 0 outside the interval $n \in [0, N\text{-}1]$.

Most windows adopted by coding standards have a tapered-end appearance in time domain: beginning and end of the window have low amplitudes, with a peak located in between. These windows are described by simple formulas, and their selections are inspired by the smooth appeal in time domain, frequently linked to spectrum estimation applications. We propose in the present study an optimization procedure that can lead to windows with better performance. The technique is based on gradient-descent where the gradient is derived from the Levinson-Durbin algorithm.

2. OPTIMAL WINDOW

Input signal samples located within the *analysis interval* are processed to obtain the LP coefficients, these are used for actual synthesis or prediction inside the *synthesis interval*. The two intervals might not be the same in practice. Several metrics are defined to quantify the performance of a given window. The prediction-error energy at the synthesis interval $n \in [n_1, n_2]$ is given by

$$\begin{aligned} J &= \sum_{n=n_1}^{n_2} \big(e[n]\big)^2 = \sum_{n=n_1}^{n_2} \big(s[n]-\hat{s}[n]\big)^2 \\ &= \sum_{n=n_1}^{n_2} \left(s[n]+ \sum_{i=1}^{M} a_i\, s[n-i] \right)^2 \end{aligned} \qquad (2)$$

note that $e[n]$ denotes the prediction error; input speech and predicted speech are denoted by $s[n]$ and $\hat{s}[n]$, respectively; a_i, i = 1 to M are the LP coefficients, with M being the prediction order. Alternative performance quantifier is the prediction gain, given by

$$PG = 10\log_{10}\left(\sum_{n=n_1}^{n_2} \big(s[n]\big)^2 \Big/ \sum_{n=n_1}^{n_2} \big(e[n]\big)^2 \right), \qquad (3)$$

that is, it is the ratio (in dB) between the signal energy and prediction-error energy.

Given a large amount of speech data, the optimal window is the one capable of producing the lowest prediction-error energy, or highest prediction gain, averaged over all signal frames. This definition makes sense for coding applications, since most of them rely on some form of quantization of the excitation signal to the synthesis filter. Thus, the lower the magnitude of the

excitation signal, the lower is the error of the synthetic speech.

In the subsequent sections we will measure the performance of the window using prediction-error power (PEP) – the time-averaged prediction-error energy, and segmental prediction gain (SPG) – the frame-averaged prediction gain in decibel domain.

3. GRADIENT-DESCENT OPTIMIZATION

The prediction-error energy (2) can be considered as a function of the N samples of the window. By finding the gradient of J with respect to the window sequence

$$\nabla J = \left[\frac{\partial J}{\partial w[0]} \quad \frac{\partial J}{\partial w[1]} \quad \cdots \quad \frac{\partial J}{\partial w[N-1]} \right]^T \tag{4}$$

it is possible to adjust it in the direction negative to the gradient so as to reduce the prediction-error energy; this is the principle of gradient-descent and is widely used to deal with a variety of practical problems [2]. Derivative of (2) with respect to the window sequence $w[n]$ is found as

$$\begin{aligned} \frac{\partial J}{\partial w[n]} &= \sum_{k=n_1}^{n_2} 2e[k] \frac{\partial e[k]}{\partial w[n]} \\ &= \sum_{k=n_1}^{n_2} 2e[k] \left(\sum_{i=1}^{M} s[k-i] \frac{\partial a_i}{\partial w[n]} \right). \end{aligned} \tag{5}$$

Derivative of the LP coefficients can be calculated from the Levinson-Durbin algorithm [4]; that is, start from a zero-order predictor and increase the order one at a time. Detail procedure is given below.

- Initialization, $l = 0$:

$$\frac{\partial J_0}{\partial w[n]} = \frac{\partial R[0]}{\partial w[n]}; \; n = 0, \ldots, N\text{-}1. \tag{6}$$

- Recursion. For $l = 1, 2, \ldots, M$:

$$\begin{aligned} \frac{\partial k_l}{\partial w[n]} = \frac{1}{J_{l-1}} \Bigg(& \frac{\partial R[l]}{\partial w[n]} - \frac{R[l]}{J_{l-1}} \frac{\partial J_{l-1}}{\partial w[n]} + \\ & \sum_{i=1}^{l-1} a_i^{(l-1)} \frac{\partial R[l-i]}{\partial w[n]} + R[l-i] \frac{\partial a_i^{(l-1)}}{\partial w[n]} - \frac{a_i^{(l-1)} R[l-i]}{J_{l-1}} \frac{\partial J_{l-1}}{\partial w[n]} \Bigg), \end{aligned} \tag{7}$$

$$\frac{\partial a_l^{(l)}}{\partial w[n]} = -\frac{\partial k_l}{\partial w[n]}, \tag{8}$$

and for $i = 1, 2, \ldots, l\text{-}1$:

$$\frac{\partial a_i^{(l)}}{\partial w[n]} = \frac{\partial a_i^{(l-1)}}{\partial w[n]} - a_{l-i}^{(l-1)} \frac{\partial k_l}{\partial w[n]} - k_l \frac{\partial a_{l-i}^{(l-1)}}{\partial w[n]} \tag{9}$$

stop if $l = M$. Otherwise

$$\frac{\partial J_l}{\partial w[n]} = \left(1 - k_l^2\right) \frac{\partial J_{l-1}}{\partial w[n]} - 2k_l J_{l-1} \frac{\partial k_l}{\partial w[n]}. \tag{10}$$

Note that $a_i = a_i^{(M)}$ and the k_ls are the reflection coefficients. The sought derivatives are found in (8) and (9). In actual implementation, steps (6) to (10) are embedded within the framework of the Levinson-Durbin algorithm, where the LP coefficients are found from the known autocorrelation values. Derivative of (1) is given by

$$\frac{\partial R[l]}{\partial w[n]} = \begin{cases} w[n+l]s[n+l]s[n]; 0 \le n < l \\ w[n-l]s[n-l]s[n]; \; N-l \le n < N \\ s[n]\left(w[n-l]s[n-l] + \right. \\ \left. w[n+l]s[n+l] \right) \quad ; \text{otherwise} \end{cases} \tag{11}$$

In summary, we compute the gradient (4) by finding the autocorrelation values using the window and the input signal (1), their derivatives are found from (11); the LP coefficients are solved using the Levinson-Durbin algorithm with the autocorrelation values as inputs, their derivatives are found using (6) to (10); in addition, the prediction-error sequence is calculated using the input signal and the LP coefficients. The results are plugged into (5) to obtain the desired quantities.

Given a set of speech data $\{s_k[n], k = 0, 1, \ldots, N_t\text{-}1\}$ known as the training data set of size N_t, where each $s_k[n]$ is an array containing the speech samples. We seek to find a window in such a way that the PEP averaged over the entire training data set is minimized. The steps to follow are:

1. Initialization. Use an initial window w_0 and compute the PEP of the whole training set, denote result as PEP_0. Initial window can be chosen arbitrarily. Set $m \leftarrow 1$.
2. Gradient-descent. Set $w_m \leftarrow w_{m-1}$. For $k \leftarrow 0$ to $N_t - 1$:
   - Calculate error gradient ∇J with respect to w_m and s_k.
   - Update window:

$$w_m[n] \leftarrow w_m[n] - m \cdot \frac{\partial J}{\partial w_m[n]}. \tag{12}$$

3. Decision to stop training. Find PEP using w_m for the whole training set, denote result as PEP_m. If PEP_m has not decreased substantially with respect to PEP_{m-1}, stop; otherwise set $m \leftarrow m+1$ and go back to Step 2.

In order to find the *PEP* of a certain window with respect to the training data set, the whole set is LP analyzed frame-by-frame using the given window with the power of prediction error found at the end, which represents a

performance measure. Also note in (12) that the step size parameter m determines the adaptation speed, it is constant and can be selected experimentally for a given situation. One complete pass through the training data set with window update is referred to as one epoch (Step 2).

4. EXPERIMENTAL RESULTS

All experiments share the same training data set, created using 54 files from the TIMIT database [3] (downsampled to 8kHz), total duration is approximately three minutes. To evaluate the generalization capability of the optimized window for signals outside the training data set, a testing data set is formed using 6 files not included in the training data set; total duration is roughly 8.4 second. Prediction order is always equal to ten.

The first experiment consists on applying the window optimization procedure for four window lengths: 120, 160, 200, and 240 samples. Total number of training epochs is equal to 100, with $m = 10^{-9}$ and an initial rectangular window. In addition, the analysis interval is equal to the synthesis interval with lengths equal to that of the window. Figure 1 shows the SPG results, where we can appreciate the growth as training progresses, which tend to saturate after roughly 20 epochs. Performance gain is usually high at the beginning of the training cycles, with gradual lowering and eventual arrival to a local optimum. Longer windows tend to have lower SPG, which is expected since the same prediction order is applied for all cases, and a lower number of samples are better modeled by the same amount of LP coefficients.

Figure 2 shows the initial and final windows for $N = 240$. All windows upon convergence developed a similar tapered-end appearance, with the middle samples slightly elevated. The experiment was repeated for an initial Hamming window, with the results summarized in Figure 3. As we can see, performances of the optimized windows regardless of the initial condition are quite similar; in fact, their final shapes are alike, implying that sensitivity of the solution toward initial window is low. Moreover, enhancement is consistent for both training and testing data set, hence optimization gain can be generalized for data outside the training set. For initial rectangular window, PEP dropped an average of 6.3% in training, and 8.2% in testing. For initial Hamming window, PEP dropped an average of 1.2% in training, and 1.3% in testing. Percentage drop in PEP for the case of Hamming window is not as much as for the initial rectangular window, since the Hamming window with its tapered-end shape represents a superior choice for the problem at hand.

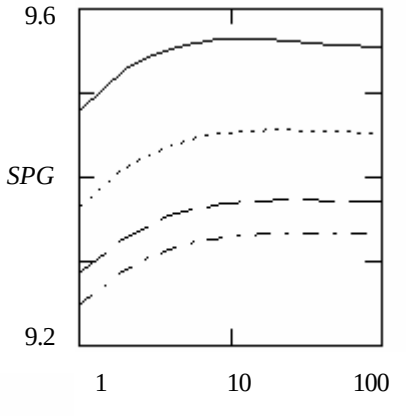

Figure 1. Segmental prediction gain (*SPG*) as a function of the training epoch (m) in an experiment. Four window lengths are considered, starting from top: 120, 160, 200, and 240.

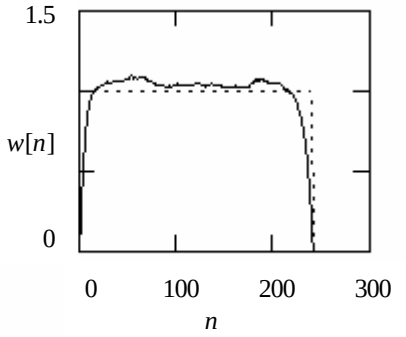

Figure 2. Initial and final (solid trace) windows for a length of 240 samples in an experiment.

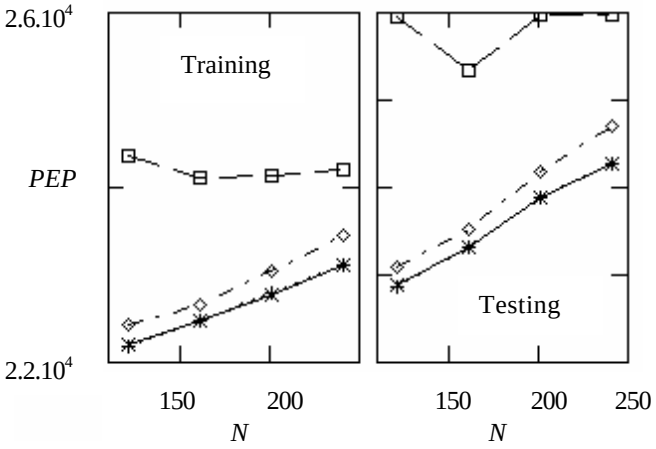

Figure 3 Performance comparisons (*PEP*) between four windows as a function of the length (N): rectangular (square), Hamming (diamond), optimized from rectangular (x), and optimized from Hamming (+).

In the second experiment we consider the 240-sample analysis interval with coordinate $n \in [0, 239]$. Five synthesis intervals are defined: $\boldsymbol{I}_1 = [0, 59]$, $\boldsymbol{I}_2 = [60, 119]$, $\boldsymbol{I}_3 = [120, 179]$, $\boldsymbol{I}_4 = [180, 239]$, and $\boldsymbol{I}_5 = [240, 259]$; that is, the first four intervals are located inside the analysis interval, while the last one is located outside; a 240-sample initial rectangular window is used. Optimization is performed for 1000 epochs with $m = 10^{-9}$. The final windows are plotted in Figure 4; as expected, they take on a shape that reflects the underlying position of the synthesis interval. Performance gain for $\boldsymbol{I}_1$ to $\boldsymbol{I}_4$ is due to suppression of signals outside the region of interest; while for $\boldsymbol{I}_5$, putting most of the weights near the end of the analysis interval plays an important role. The PEP dropped an average of 10.1% in training and 13.6% in testing.

5. CONCLUSION

A window optimization procedure for the autocorrelation method of LP analysis is described, which is based on the principle of gradient-descent. Given a large amount of training data, the window is tuned so as to reduce the prediction-error power. In essence, the best shape of the window is found, maintaining the rest of the parameters (length, prediction order, etc.) fixed. Based on the gradient, the window is changed toward the optimal shape, defined as the one providing the lowest prediction-error power. It is observed that the optimized window can be generalized in the sense that performance gain applies equally well to data outside the training data set. The described approach was inspired by back-propagation training in neural networks [5]. We are currently extending the method to the schemes adopted by standardized coders.

6. REFERENCES

[1] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*, Prentice Hall, Englewood Cliffs, NJ, 1978.

[2] S. S. Rao, *Engineering Optimization – Theory and Practice*, 3rd edition, John Wiley & Sons, 1996.

[3] J. Garofolo et al, *DARPA TIMIT, Acoustic-Phonetic Continuous Speech Corpus CD-ROM*, National Institute of Standards and Technology, 1993.

[4] S. J. Orfanidis, *Optimum Signal Processing*, 2nd edition, McGraw-Hill, 1988.

[5] S. Haykin, *Neural Networks*, Macmillan, 1994.

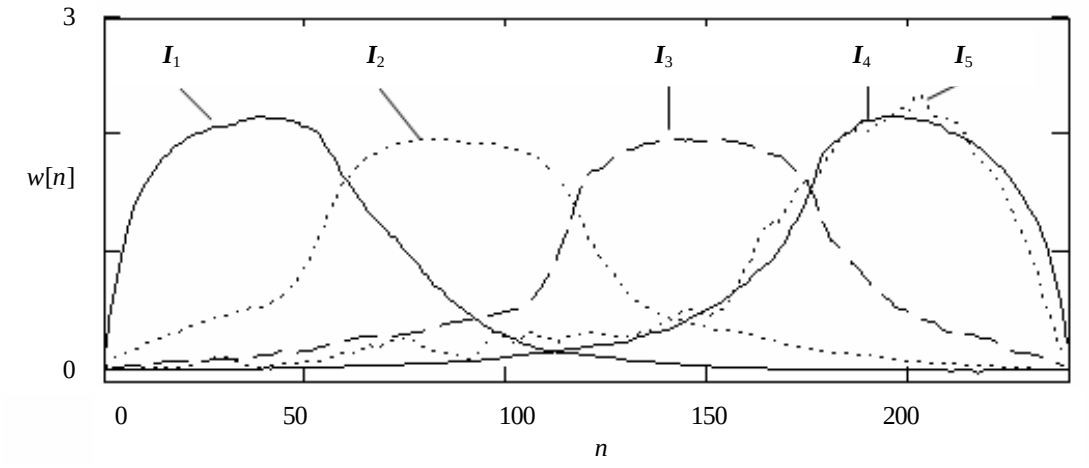

Figure 4. Optimized windows in an experiment, a 240-sample rectangular window is used to start.