Lossless Compression of High Fidelity Audio and Adaptive Linear Prediction

Andrew L. Adams
Harris RF Communications
Rochester, NY 14610

Steven W. McLaughlin
Electrical Engineering Department
Rochester Institute of Technology
Rochester, NY 14623

Abstract

We consider the lossless compression of high fidelity (e.g. 16-bit) digital audio using adaptive linear prediction. Both linear predictive coding (LPC) and least mean squares (LMS) predictors are considered. Preliminary results are presented for the compression of industry standard Sound Quality Assessment Material (SQAM) [1] samples from 16 bits to 1.5 - 3 bits. Previous results by others on the same audio source was in the 8-bit range.

Preliminaries

Conventional lossless compression algorithms (e.g. Lempel-Ziv) do not compress 16-bit digital audio very well largely because of the wide dynamic range (i.e. large alphabet) and extended redundancy of audio samples. We investigate methods which losslessly reduce the dynamic range and extract long term correlation. Linear prediction uses a linear combination of past (and possibly future) samples to predict the present. If the predictor is accurate, the difference between the prediction and the actual value will be small. A good entropy coder can be used to encode the error sequence. By saving only the error sequence and generating the prediction, the original source sequence can be reproduced perfectly.

1. Error Sequence Generation

The error sequence is generated by an adaptive finite impulse response (FIR) digital filter. The poles, or weights, of the filter are adapted using the LMS or LPC method. The predicted value $\chi(k)$ is:

$$\chi(k) = \sum_{i=1}^{N} w_i \, x(k-j) \qquad (1.1)$$

where w_i are filter weights, $x(k)$ is the source sequence. The error sequence $e(k)$ is:

$$e(k) = x(k) - \chi(k) \qquad (1.2)$$

Signal Reconstruction

The original signal is reconstructed by using the same FIR filter used at the encoder and adding the error to the decoder prediction. To guarantee lossless reproduction, the first N samples of the source $x(n)$ and the filter weights w_j must be transmitted with the encoded error sequence $\chi(n)$. This allows the Nth + 1 sample and beyond to be predicted using (1.1).

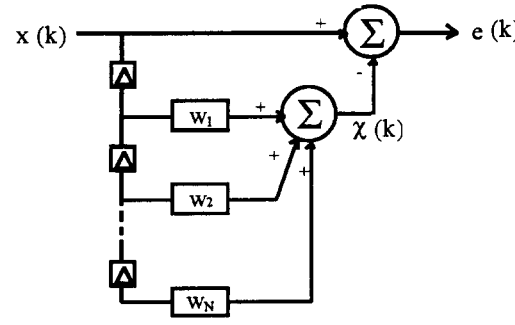

Figure 1.1: Error Sequence Generation

2. LMS Introduction

The LMS algorithm adapts the filter weights to minimize the average power of the error between the prediction and the actual sample. This method is popular because of its ease of implementation and relatively good performance [1,ch.6]. The following is an overview of of the LMS algorithm. For more details refer to reference [1,chs. 2,6]. Converting (1.1) to vector form yields:

$$\chi(k) = X_k^T W_k \qquad (2.1)$$

where:

$$X_k^T = [x(k-1)\ x(k-2)\ \ldots\ x(k-N)]$$

$$W_k = [\, w_k\ w_{k-1}\ \ldots\ w_{k-N}\,]$$

Substituting (2.1) into (1.2) yields:

$$e(k) = x(k) - X_k^T W_k \qquad (2.2)$$

0-7803-2431-5/95 $4.00

Gradient Estimation

By transforming (2.2) into matrix form, gradient methods can be used to solve for the optimum weight vector to produce the minimum mean square error. For LMS, an estimate of the gradient is used in a steepest descent algorithm to approach the actual $W_{k\ opt}$[1]. The instantaneous error power $e^2(k)$ is used as the estimate of the mean square error. By taking the gradient of (2.2) the LMS gradient estimate is:

$$\nabla_{Est} = 2e(k)X_k^T \qquad (2.3)$$

and the updated filter weights are

$$W_{k+1} = W_k + \mu\nabla_{Est} \qquad (2.4)$$

By substituting (2.3) into (2.4) the LMS algorithm results:

$$W_{k+1} = W_k + \mu 2e(k)X_k^T \qquad (2.5)$$

Equation (2.5) adapts the weights of the linear predictor where the gain constant μ controls the rate of adaptation. Because the squared instantaneous error is used as an estimate of the mean square error, the LMS algorithm is inherently noisy [1]. To obtain lossless reconstruction, the following information must be transmitted to the decoder:

- Number of weights used in the linear predictor to produce the error sequence
- The first N 16 bit samples of the audio file
- The N floating point filter weights used to produce the predicted value for the N+1 sample
- The encoded error sequence for samples N+1 to the length of the original input.

By storing the first N input samples and the N filter weights, the N+ 1 sample may be predicted using the same methodology as above. This predicted value is identical to the prediction at the encoder because it is produced from the same input values, the same filter weights and the same algorithm. By truncating the predicted value (as is done when the error sequence is generated) and adding the error for the N+1 sample, the original N+1 sample results.

LMS Entropy Calculation

We consider the first order entropy of the error sequences. A histogram of every possible error value is generated. The error is discrete and limited to a 16 bit signed value (-32,768 to 32,767). From the histogram the probability $p(n)$ of each error value is easily calculated giving the first order entropy:

$$H = -\sum_{n=-32768}^{32786} p(n)\ log_2\ p(n) \qquad (2.6)$$

For a better estimate of the entropy (i.e. entropy rate) one would use joint distributions of the error sequence[4].

LPC Introduction

The LPC (linear predictive coding) method of linear prediction attempts to model the audio samples as an autoregressive process. An autoregressive process is an all pole filter whose description matches (1.1). The weights of the filter are determined by minimizing the square of the error between the actual and predicted value [2].

Due to the non-stationarity of the audio input, the autoregressive model is only valid for a short time. To account for this, the LPC algorithm produces an autoregressive model for small blocks of input [3,ch. 12]. For greater detail on LPC coding refer to [2].

The optimum weights w_k for the autoregressive model are derived by minimizing the error between the actual value and the predicted value using the least squares criteria [2]. Squaring (1.2), differentiating with respect to w_k and setting equal to 0 to calculate the optimum weights yields:

$$0 = -2x(k)\sum_{i=1}^{N} x(k-j) - 2\sum_{i=1}^{N} w_j\ x(k-j) \qquad (3.1)$$

Using the definition of the autocorrelation function $r(k)$ (3.1) reduces to (the Yule-Walker equation):

$$r(k) = \sum_{i=1}^{N} w_j\ r(k-j) \qquad (3.2)$$

Eq. (3.2) can be put in matrix form. This autocorrelation matrix is Toeplitz and positive semi-definite, which may be solved efficiently using the Levinson-Durbin algorithm. The controlling parameters for the LPC implementation are the block size M and the number of filter weights N for the autoregressive model.

Autocorrelation Estimate

An estimate of the autocorrelation values must be calculated for the sample block. The number of autocorrelation values needed is N + 1. The autocorrelation values $r_{Est}[k]$ must be floating point values to maintain the accuracy of the calculated weights w_k. It was found that truncating the autocorrelation values induced error into the calculated weights when the Yule-Walker equations were solved. This error would cause the entropy of the resulting output to begin to rise as the number of weights was increased passed a threshold value.

Error Sequence Generation

Once the weights for the input block have been calculated, the predicted values can be calculated. For each input in the block, a predicted value is calculated using (1.1). The resulting prediction is a floating point value because the filter weights are floating point. The prediction is truncated and the error between the actual input and the truncated prediction is calculated using (1.2). The error value is saved for use in reconstruction. This process is repeated for every input in the block, using the same filter weights for every prediction. When all the error values for the block have been calculated, a new block is read in and the process repeats, starting with the estimate of the autocorrelation values of the new block.

Reconstruction

The information needed to losslessly reconstruct an audio sample sequence using the LPC prediction method presented above is broken into two classes: file information and block information. The file information consists of the following:

- The number of filter weights N used to produce the block error sequence (a 16 bit value).
- The block size (a 16 bit value).

It is natural to assume a fixed block size and fixed number of filter weights throughout. The block information consists of the following:

- The first N audio samples of the block not encoded (16 bit values)
- The N floating point filter weights used for the block prediction (32 bit floating point values)
- The encoded error sequence for samples N+1 to the length of the block.

Entropy Calculation

Only the first order entropy of the error for each block produced by the LPC algorithm is calculated. A histogram of every possible error value is generated.The first order block entropy can be calculated using the histogram probabilities:

$$H(k) = -\sum_{n=-32768}^{32768} p(n)\ log_2\ p(n) \qquad (3.3)$$

$$k=1,2,...,M$$

The block entropy is averaged to get the first order entropy estimate

$$H_{LPC} = \frac{\sum_{k=1}^{M} H(k)}{M} \qquad (3.4)$$

4. Results

For the audio samples, Sound Quality Assessment Material (SQAM) [5] recordings for subjective tests were used. The following three samples from SQAM were used:

Track	Sample	Duration (sec)
66	Stravinsky	0:18
69	ABBA	0:33
70	Eddie Rabbitt	0:21

The previously known results on these music samples are given in [6] where a binomial coefficient predictor achieved a compression ratio of about 2:1,namely 16-to-8 bits (the results in [6] are "compression ratio vs. time" and not entropy as in our case).

LMS Results. The following tables show the results of the LMS implementation for SQAM audio samples. Each row is a specific learning constant and each column is a specific number of filter weights. Each table entry is the calculated entropy for the given learning constant and number of filter weights.

SQAM66

	3	5	10	20
$1x10^{-9}$	2.6	1.8	1.6	1.5
$5x10^{-10}$	3.1	2.2	1.7	1.51

SQAM69

	3	5	10	20
$1x10^{-9}$	2.6	1.9	1.6	1.5
$5x10^{-10}$	3.1	2.1	1.8	1.5

SQAM70

	3	5	10	20
$1x10^{-9}$	2.6	1.9	1.6	1.5
$5x10^{-10}$	3.0	2.0	1.7	1.5

LPC Results

The following tables show the results of the LPC implementation for SQAM audio samples and other music samples. Each row is a specific block size and each column is a specific number of filter weights. Each table entry is the calculated entropy for the given block size and number of filter weights.

SQAM66

	3	5	10	20
128	6.3	6.0	5.7	5.5
1024	5.5	4.7	3.9	3.6
4096	3.8	3.2	2.6	2.4
8192	3.2	2.7	2.2	2.0

SQAM69

	3	5	10	20
128	6.3	6.0	5.7	5.5
1024	5.5	4.7	3.9	3.6
4096	3.8	3.2	2.6	2.4
8192	3.2	2.6	2.2	2.0

SQAM70

	3	5	10	20
128	6.3	6.0	5.7	5.5
1024	5.5	4.7	3.9	3.6
4096	3.9	3.2	2.7	2.4
8192	3.4	2.6	2.3	2.1

5 Conclusions

A compression technique that uses the well known LMS and LPC predictors has been used to losslessy compress high fidelity audio. The procedure improves the compression ratioes (in the 8 bit range) to around 1.5-3 bits.

References

[1] Widrow, B., and Stearns, S. D. (1985), *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, N.J.

[2] Kay, S. M. (1988), *Modern Spectral Estimation, Theory & Application*, Prentice-Hall, Englewood Cliffs, N.J.

[3]Proakis, J. G. (1988), *Introduction To Digital Signal Processing*, MacMillan, New York.

[4] Cover, T. M., and Thomas, J. A. (1991), *Elements of Information Theory*, John Wiley & Sons, New York.

[5]European Broadcast Union Sound Quality Assessment Material Compact Disk, Catalog N.422.204.2, 1988.

[6]Cellier, C., Chênes, P., and Rossi, M. (1993) "Lossless Audio Data Compression For Real Time Applications", Audio Engineering Society, Presented at 95th Convention, New York.