Lossless Data Compression Using Adaptive Filters

N. Magotra†, W. McCoy†, F. Livingston†, S. Stearns‡

† Dept. of EECE, University of New Mexico, Albuquerque, NM 87131
‡ Dept. 9311, Sandia National Laboratory, Albuquerque, NM 87185

Ph.:505-277-0808/email:magotra@houdini.eece.unm.edu

Abstract

This paper describes the application of adaptive filters in a two stage lossless data compression algorithm. The term lossless implies that the original data can be recovered exactly. The first stage of the scheme consists of a lossless adaptive predictor while the second stage performs arithmetic coding. The unique aspects of this paper are (a) defining the concept of a reversible filter as opposed to an invertible filter (b) performing lossless data compression using primarily floating-point operations (c) designing lossless adaptive predictors (d) using a modified arithmetic coding algorithm that can readily handle inputs consisting of more than 14 bits.

1. INTRODUCTION

Lossless data compression has applications in the area of archiving research data bases, high fidelity audio compression, medical and radar image data compression and other areas where fidelity of the signal is critical. Because there is less data involved with a one-dimensional signal, the algorithms for waveform data compression can be much more complex than their two dimensional counterparts. In addition, waveforms such as speech tend to arise from more complicated models than image data. In this paper we focus on lossless speech and seismic waveform data compression.

The first stage of the lossless data compression algorithm described in this paper is an adaptive predictor. However, in general, any reversible filter can be used in this stage. The idea of reversible filtering is very closely related to predictive coding as it was presented by Atal in his work at Bell Labs [1]. Predictive coding was developed for the purpose of bit rate reduction in telephone calls. It was discovered that by quantizing and transmitting the residue sequence, the bit rate could be substantially reduced while an acceptable level of fidelity could be maintained at the receiver. More recently, a modified predictive coder for a digital, lossless system has been proposed [2], as shown in Figure 1. This paper is a generalization of the aforementioned technique using adaptive filtering techniques and floating point arithmetic.

The second stage of the lossless data compression algorithm consists of an encoding stage. In the present version we use a modified form of arithmetic coding [3, 4].

We have shown [5] that various predictor structures can be used in the first stage for lossless data compression with only a few critical restrictions. The predictor structures may be linear or nonlinear, fixed or adaptive, and can be implemented using floating point arithmetic. These algorithms are incrementally processed as opposed to block processed.

The discussion in this paper is limited to purely digital systems. In order to be considered a reversible filter, a filter must have a reverse which exactly restores its output to the bit-for-bit exact original input as shown in Figure 2. The question is not simply one of structure, but also of implementation. If computers were not limited in precision, transform operations such as the FFT would be reversible; causal, invertible filters would be reversible, but they are not, in general reversible because of finite precision effects.

2. PREDICTION STAGE

The following discussion describes the lossless prediction stage in detail. Figure 2 shows a block diagram of a predictor which can be implemented by the following equation,

$$e_n = x_n - H(\mathbf{X}_n^-, \theta_n^-) \tag{1}$$

where x_n is the current integer sample value and $H(\mathbf{X}_n^-, \theta_n^-)$ is the predicted value. $\mathbf{X}_n^-$ is a sequence of vectors representing the past samples of the signal source and possibly past values of the error sequence. θ_n^- is the sequence of weight vectors and e_n is the difference between the current sample value and the predicted value. The receiver processor recovers the current sample using the following equation.

$$x_n = e_n + H(\mathbf{X}_n^-, \theta_n^-) \tag{2}$$

In order to implement Equation (2), the receiver must have access to e_n and must also be able to reconstruct $H(\bullet)$. e_n will be decoded from the transmitted bit stream at the receiver. The receiver processor will be able to reconstruct $H(\bullet)$ under two conditions. First, the receiver processor must have access to $\mathbf{X}_n^-$ and θ_n^-. Therefore, a keyword may need to be transmitted prior to decompression which contains information such as a signal model, or the number of transmitted bits per symbol. Second, it must also be able to generate, bit for bit, the exact same $H(\bullet)$ as the compression processor. The goal of the first

0-7803-2431-5/95 $4.00

stage of the algorithm is to losslessly decorrelate the input data.

We have shown that, it is possible to use floating point arithmetic on the past values and still preserve the data integrity. This makes the implementation of adaptive techniques such as the normalized least mean squared (NLMS) algorithm very straightforward. The only restrictions being that the prediction filter output is rounded and the compression and recovery algorithms have to be implemented on the same architecture. The following equations summarize the NLMS algorithm.

$$e(n) = u(n) - \text{ROUND}\left[\mathbf{w}_M^T(n)\mathbf{u}_M(n-1)\right]$$
$$\mathbf{w}_M^T(n) = \left[w_1(n)\, w_2(n) \cdots w_M(n)\right] \quad (3)$$
$$\mathbf{u}_M^T(n-1) = \left[u(n-1)\, u(n-2) \cdots u(n-M)\right]$$

$$\mathbf{w}_M(n+1) = \mathbf{w}_M(n) + \mu(n)e(n)\mathbf{u}_M(n-1) \quad (4)$$

$$\hat{\sigma}_{uu}^2(n) = \beta\hat{\sigma}_{uu}^2(n-1) + (1-\beta)e_{i-1}^2(n-1) \quad (5)$$

$$\mu(n) = \frac{\mu}{\hat{\sigma}_{uu}^2(n)} \quad (6)$$

where $\mathbf{w}_M$ is the weight vector, $\mathbf{u}_M$ is the input data vector, μ is a convergence parameter, β is a smoothing parameter, and $\hat{\sigma}_{uu}$ is an input power estimate. The algorithm can be losslessly reversed as follows.

$$\mathbf{w}_M(n+1) = \mathbf{w}_M(n) + \mu(n)e(n)\mathbf{u}_M(n-1) \quad (7)$$

$$u(n) = e(n) + \text{ROUND}\left[\mathbf{w}_M^T(n)\mathbf{u}_M(n-1)\right] \quad (8)$$

The power estimate is updated according to equation (6).

In searching for a good adaptive algorithm, the primary goal was fast convergence because it translates into lower mean squared error and higher compression ratios. Lattice filters are known for their fast convergence properties, therefore, they were investigated next. The structure for a lattice filter is different than an adaptive transversal filter in that the current sample undergoes M additions where M is the order of the filter as can be seen in Figure 3. The implementation is a bit more difficult for a lattice filter. It is necessary for the lattice predictor to have an a priori algorithm, that is, all coefficient updates must be a function of only past values of the error. In order to compress the data the operands must all be of the same precision as the input data which typically would necessitate a rounding operation prior to each addition. This produced large numerical errors in practice and it was necessary to reconfigure the computations as given below.

We found the recursive least squares lattice (RLSL) adaptive filtering technique to be optimal for this application. Recursive least squares algorithms are designed to lock onto the optimal weight vector at each iteration of the algorithm and because of this were able to perform better than the gradient adaptive algorithm.

The algorithm we used is the a priori RLSL as described by Haykin [6]. The key equations are summarized below.

RLSL structures minimize the energy functions given by Equations (9) and (10) at each time instant, n.

$$F_m(n) = \sum_{i=0}^{n} \lambda^{n-i}\eta_m^2(i) \quad (9)$$

$$B_m(n) = \sum_{i=0}^{n} \lambda^{n-i}\psi_m^2(i) \quad (10)$$

$\eta_m(n)$ and $\psi_m(n)$ are the forward and backward prediction errors respectively at time n and of order m. The order update recursions are given by

$$\Delta_{m-1}(n) = \lambda\Delta_{m-1}(n-1) + \gamma_{m-1}(n-1)\psi_{m-1}(n-1)\eta_{m-1}(n) \quad (11)$$

where $\Delta_{m-1}(n)$ is a lattice coefficient for stage $m-1$ which has not been normalized, $\gamma_{m-1}(n)$ is as defined below in Equation (16), and λ is the recursive least squares "forgetting factor". The $\Delta_{m-1}(n)$ coefficient is the same for both forward and backward prediction coefficients of the same order.

$$\Gamma_{f,m}(n) = -\frac{\Delta_{m-1}(n)}{B_{m-1}(n-1)} \quad (12a)$$

$$\Gamma_{b,m}(n) = -\frac{\Delta_{m-1}(n)}{F_{m-1}(n)} \quad (12b)$$

$\Gamma_{f,m}(n)$ and $\Gamma_{b,m}(n)$ are the normalized forward and backward lattice coefficients. The basic equations that actually implement the lattice structure are given below

$$\eta_m(n) = \eta_{m-1}(n) + \Gamma_{f,m}(n-1)\psi_{m-1}(n-1) \quad (13)$$

$$\psi_m(n) = \psi_{m-1}(n-1) + \Gamma_{b,m}(n-1)\eta_{m-1}(n) \quad (14)$$

The following two equations represent updated energy functions for the forward and backward predictors.

$$F_{m-1}(n) = \lambda F_{m-1}(n-1) + \gamma_{m-1}(n-1)\eta_{m-1}^2(n) \quad (15a)$$

$$B_{m-1}(n) = \lambda B_{m-1}(n-1) + \gamma_{m-1}(n-1)\psi_{m-1}^2(n) \quad (15b)$$

$$\gamma_m(n) = \gamma_{m-1}(n) - \frac{\gamma_{m-1}^2(n)\psi_{m-1}^2(n)}{B_{m-1}(n)} \quad (16)$$

$\gamma_m(n)$ is a conversion factor which allows the lattice filter to quickly adapt to sudden changes in the input data.

To ensure lossless operation of the algorithm, the final stage lattice output is computed as a single summation equation to prevent excessive round off errors. As long as the same lattice equations are implemented at the time of uncompressing the data (on a computer of similar architecture as the one used for compression) the scheme is lossless.

3. ARITHMETIC CODING

The residue (error) sequence generated by the prediction stage is encoded in the second stage. We use an arithmetic coding algorithm as described below. Arithmetic coding is able to incrementally compress data at near optimal rates and is also able to adapt to changing statistics [3]. The arithmetic coding algorithm includes two independent parts, an adaptive probability model, and an incremental transmission implementation.

The probability model consists of an array of integer frequency counts which is adapted with the arrival of each new symbol. The higher the frequency of a particular symbol, the larger its bin in the probability model. The total frequency count in the array must not exceed a predetermined value in order to prevent overflow. Each frequency count must be at least one.

The basic concept behind arithmetic coding is to map a sequence of integers from a random source onto a real number with precision related to the sequence length. Upon the arrival of each new symbol a new range is computed which is a function of the current range and the limits of the bin in the probability model associated with that symbol. Key equations [3] for encoding are given below.

$$h(n) = l(n-1) + bin\_hi(n) * pcn \quad (17)$$

$$l(n) = l(n-1) + bin\_low(n) * pcn \quad (18)$$

where h, l are the high and low of the current range respectively, $bin\_hi(n)$, $bin\_low(n)$ are the current limits of the probability model corresponding to the current symbol, and pcn is a number inversely related to the precision of the current range limits.

After computing $h(n)$ and $l(n)$ the arithmetic coder transmits all common leading bits between $h(n)$ and $l(n)$ and recomputes the range as follows [3].

if ($l \geq Half$)

  Transmit 1

  $h = (h - Half)*2$

  $l = (l - Half)*2$

  $pcn = pcn*2$

if ($h < Half$)

  Transmit 0

  $h = h*2$

  $l = l*2$

  $pcn = pcn*2$

In this pseudo-code, *Half* represents the midpoint of the range of long integer values on the computer architecture. Conditions of underflow must be avoided and are explained in detail in [3].

The coding scheme described above has been shown to give near optimal compression for white gaussian residue sequences with dynamic ranges less than 14 bits [3]. At ranges larger than 14 bits, overflow can occur, and the symbol set becomes large and cumbersome. Improvements have been made recently in the arithmetic coding algorithm which solve this problem [4]. This paper presents results incorporating these improvements. The results should give an idea of the kind of compression ratios to be expected from this algorithm.

Table 1 presents the compression results obtained by using the RLSL algorithm (20 stages, $\lambda = 0.996$) and modified arithmetic code. As seen from the results the average compression ratio is 2.5. Note that the compression ratio for each event was obtained by assuming that the input data were represented in the minimum number of bits needed to represent the maximum absolute value.

4. RESULTS AND CONCLUSIONS

Adaptive algorithms have been demonstrated in application to the lossless data compression problem. The RLSL algorithm has been demonstrated to perform the best for a seismic data base. At present various other processing techniques are being investigated for the first stage and the entire algorithm is undergoing trials using speech and image (Synthetic Aperture Radar (SAR)) data. A real time version of the algorithm for seismic event data compression is currently being implemented using the Texas Instruments TMS320C3x floating-point digital signal processing chip.

References

[1] Atal, B.S. and Schroeder, M.R., "Adaptive Predictive Coding of Speech Signals", The Bell System Technical Journal, Oct., 1970, pp. 1973-1986.

[2] Stearns, S. D. , Tan, L., and Magotra, N., "Lossless Compression of Waveform Data for Efficient Storage and Transmission", IEEE Transactions on Geoscience and Remote Sensing, May 1993.

[3] Witten, Neal, and Cleary, "Arithmetic Coding for Data Compression", Communications of the ACM, pp. 520-540, June 1987.

[4] Stearns, S.D., "Arithmetic Coding in Lossless Waveform Data Compression", Submitted for publication, August 1994.

[5] McCoy, J.W., Design and Analysis of Predictors for Lossless Data Compression, Ph.D. Dissertation, Dept. of Computer and Elect. Engr., April 1995.

[6] Haykin, S. Adaptive Filter Theory, Prentice Hall, 2nd edition, 1991.

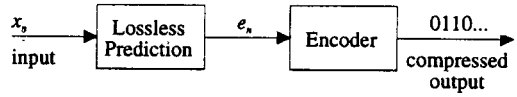

Figure 1. Algorithm Block Diagram

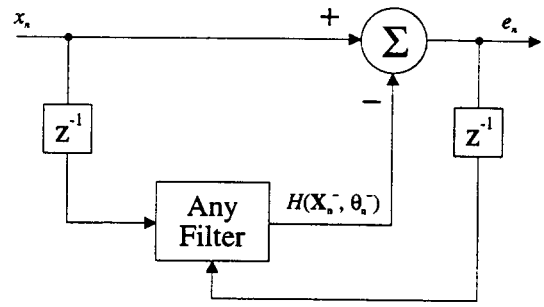

Figure 2. Block Diagram of a Lossless Predictor

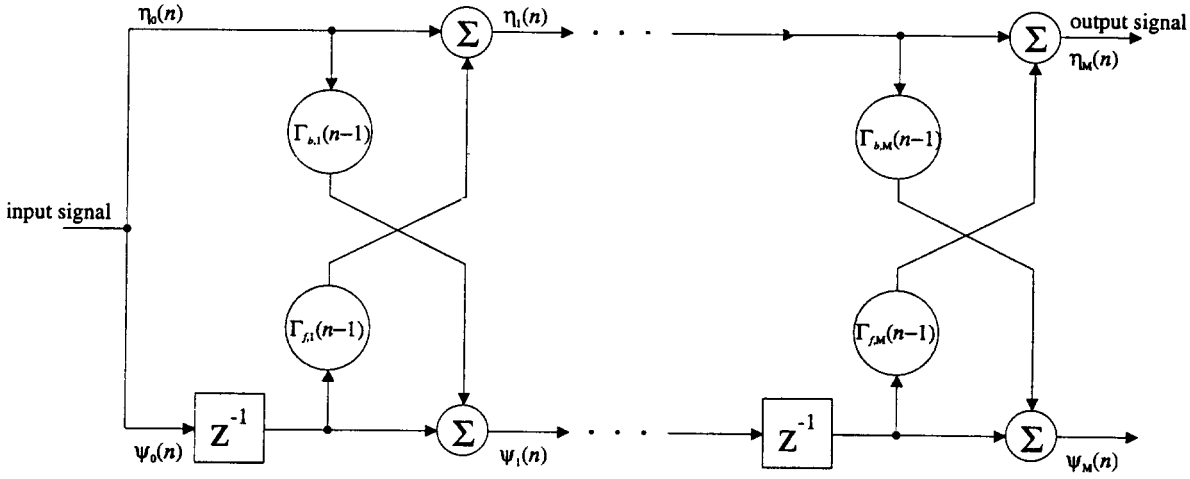

Figure 3. M-Stage Lattice Predictor

Event ID	Final Compression Ratio
anmbhz89	4.79
anmbhz92	3.19
anmehz92	2.12
anmlhz92	1.46
kipehz13	2.60
kipehz20	2.03
rarbhz92	2.42
rarlhz92	1.30

Table 1. RLSL Based Result Summary